Određivanje glikoproteina iz humane plazme

Rapčan, Borna

Borna Rapčan

Određivanje glikoproteina iz humane plazme

DIPLOMSKI RAD

Predan Sveučilištu u Zagrebu Farmaceutsko-biokemijskom fakultetu

Zagreb, 2019.

Ovaj diplomski rad je prijavljen na Zavodu za biokemiju i molekularnu biologiju i rađen je pod stručnim vodstvom prof. dr.sc. Gordana Lauca u suradnji sa Genadijem Razdorovom, mag. med. biochem.

Zahvaljujem se mentoru prof. dr. sc. Gordanu Laucu na ukazanom povjerenju i uloženom trudu prilikom izrade ovog diplomskog rada.

Zahvaljujem se mag. med. biochem. Genadiju Razdorovu na pomoći pri izradi programa te na strpljenju i svim savjetima bez kojih ovaj rad ne bi bio moguć.

SADRŽAJ

1. UVOD

1.1. GLIKANI GLIKOPROTEINA

Međunarodna unija za čistu i primijenjenu kemiju (IUPAC) definira glikane kao sinonime s polisaharidima. Prema toj definiciji glikani su složene organske molekule koje se sastoje od velikog broja (obično više od 10) monosaharida međusobno povezanih glikozidnim vezama (https://goldbook.iupac.org/html/G/G02645.html). Naziv glikani se također često koristi za ugljikohidratne komponente glikokonjugata (glikoproteina, glikolipida i proteoglikana), čak i kada su te komponente oligosaharidi i monosaharidi. Zbog toga se može reći da je definicija glikana sinonimna definiciji saharida.

Glikani su posebni zbog svoje kompleksnosti u odnosu na druge složene molekule. Zbog prirode veza koje glikani tvore, njihova raznolikost i kompleksnost je mnogo veća nego u nukleinskih kiselina i peptida. Ta raznolikost proizlazi iz glikozidnih veza (ovisno na kojem mjestu veza nastane, posjeduju više od 1 mjesta vezanja po molekuli te izvedbe koje mogu biti u α i β konfguracijama), monosaharidnog sastava (koji je bogatiji od nukleinskih kiselina kojih je 4, ali manje raznolik od aminokiselina kojih ima 20), strukture glikana (razgranatost glikanskih, odnosno nerazgranatost peptidnih i nukleinskih struktura) i duljine glikana. Anomerizacija je još jedan proces koji povećava kompleksnost glikana. Anomeri su dijastereoizomeri nastali kao posljedica povezivanja monosaharida u cikličku strukturu. Zbog tog svojstva monosaharidne komponente se mogu razlikovati samo po prvom asimetričnom ugljikovom atomu koji prilikom izgradnje poluacetala može imati hidroksilnu skupinu ispod ili iznad osnovne ravnine. Pojam α-konfiguracija se koristi kada struktura odgovara referentnoj strukturi u Fischerovoj projekciji, a pojmom β-konfiguracija označava se suprotna konfiguracija. To svojstvo povećava raznolikost dva puta linearnih i razgranatih glikana. Koliko su te strukture kompleksne govori podatak da se sastavljanjem tetrasaharida korištenjem samo jednog šećera u jednoj formi prstena (npr. glukopiranoze) mogu sastaviti 1792 različite strukture (Rudd i sur., 2017).

Glikani imaju široki spektar uloga u živim organizmima. Kao polisaharidi u stanicama se koriste kao izvori energije u obliku škroba i glikogena. Osim toga, glikani mogu poprimiti strukturne i zaštitne uloge. Primjer takvih uloga su hitinski oklopi kukaca i celuloza biljaka. Pektini pomažu biljkama u prilagođavanju promjenama tijekom rasta (Voragen i sur., 2009).

Osim uloga u obliku polisaharida glikani stvaranjem glikokonjugata proširuju spektar svojih uloga. Prva skupina glikokonjugata koji će biti opisani su glikoproteini. Oni pokazuju veliki raspon uloga u živim organizmima, od strukturnih (kolagen), lubrikativnih/protektivnih (mucini), transportnih (transferin), imunoloških (imunoglobulini), hormonskih (tireostimulirajući hormon), enzimskih (alkalna fosfataza), receptorskih, pomaganja pri smatanju proteina (kalneksin) do interakcije s drugim glikanima (Cummings i Etzler, 2009). U obliku glikozaminoglikana (GAG) sudjeluju u interakcijama faktora rasta, inhibiciji proteaza, kontroli glikozilacijskih procesa, vezanja proteina virusnih kapsula i u drugim procesima. Glikani mogu stvarati i glikolipide te preko njih sudjelovati u staničnoj regulaciji rasta i apoptozi (Schnaar, 2004). U imunosnom odogovoru putem selektina koji se vežu na njih pomažu leukocitima da se usmjere na područje zahvaćeno upalom. Osim toga, virusi ih koriste za prepoznavanje stanica domaćina (Lopez, 2006), a također imaju bitnu ulogu u stvaranju krvnih grupa i drugim biološkim interakcijama.

1.2. GLIKOZILACIJA PROTEINA

Glikozilacija proteina je vrsta posttranslacijskih modifikacija, iako bi se mogla svrstati i u skupinu kotranslacijskih modifikacija budući da započinje u endoplazmatskom retikulumu (ER) paralelno s translacijom. Procjenjuje se da je približno petina svih proteina glikozilirana (Khoury i sur., 2011). Kemijski gledano, glikozilacija je reakcija u kojoj nukleotidni šećer (glikozil donor) biva prihvaćen na hidroksilnu ili neku drugu funkcionalnu skupinu druge molekule (glikozil akceptor) (Cummings i Etzler, 2009). Glikozilacija ima veliku važnost jer sudjeluje u raznim biološkim procesima poput pravilnog smatanja proteina, otpornosti proteina na toplinu i proteaze, staničnom prepoznavanju, prihvaćanju stanica za izvanstanični matriks (npr. kod leukocita receptori za navođenje), topljivosti proteina i mnogim drugim (Francisco i sur., 2019). U slučaju proteina, glikani kao donori se najčešće prihvaćaju na bočne lance aminokiselina u O- i N- glikozilaciji (Khoury i sur., 2011). Osim tih modifikacija, moguće su C-vezane glikozilacije, vezanje glikozilfosfatidilinozitolnih (GPI) sidra i fosfoglikozilacije. Osim proteina, glikozilirati se mogu lipidi i proteoglikani. Otprilike polovina svih tipično eksprimiranih proteina u stanicama prolazi kroz proces glikozilacije (Kobs, 2015).

Većina topljivih i za membranu vezanih proteina je glikozilirano (Chandler i Costello, 2016). Glikozilacija se odvija u svim granama živog svijeta - bakterijama, arhejama i eukariotima. Za stvaranje tako velikih i složenih struktura potreban je velik broj enzima i koraka u sintezi. Sinteza glikoproteina odvija se u tri koraka: povezivanje monosaharida prijenosom monosaharida s jednog supstrata na drugi (djelovanjem glikozil transferaza), hidroliza glikozidne veze i odvajanje monosaharida (uz pomoć glikozidaza, npr. manozidaze). Budući da glikozilacija ne radi po predlošku (poput translacije i transkripcije), svi navedeni događaji se ne moraju dogoditi pa se stanica mora osloniti na veliki broj enzima potrebnih za odvijanje glikozilacije. Zbog toga glikozilacija ovisi o aktivnosti enzima. Mehanizmi glikozilacije se izvode postupno reakcijama koje su ovisne o završetku prethodne enzimske reakcije (Rudd i sur., 2017). Budući da sama sinteza toliko ovisi o enzimima, njihova različita aktivnost u različitim odjeljcima i stanicama može znatno promijeniti glikozilaciju glikoproteina (Goettig, 2016). Reakcije također ovise i o dostupnosti supstrata (Rudd i sur., 2017).

Najčešći oblik glikozilacije proteina je N-glikozilacija (Khoury i sur., 2011). U N-glikozilaciji akceptori su asparaginski (Asn), odnosno argininiski (Arg) bočni lanci. Događaju se na sekrecijskim ili membranskim proteinima kod arheja i eukariota, dok kod većine bakterija to nije zapaženo. Kod eukariota N-glikozilacija počinje kao translacijski događaj u ER-u. Prvi korak je sinteza prekursora, koja počinje formacijom dolikol pirofosfatno vezanog N-acetilglukozamina (Dol-P-P-GlcNAc). Sinteza se odvija u dvije faze. Faza I događa se na citoplazmatskoj, a faza II na luminalnoj strani ER-a. U fazi I se dvije uridin difosfat N-acetilglukozaminske (UDP-GlcNAc) molekule vežu na dolikol te se potom na tu strukturu dodaje pet manoza (Man) pri čemu nastaje Dol-$GlcNAc_2$-Man_5. U drugoj fazi se rastući glikan premjesti na luminalnu stranu i dodaju mu se četiri Man i tri glukoze (Glc). Potom se prethodno složeni blokovi četrnaest šećera (dva GlcNAc, devet Man i tri Glc) dodaju na početni polipeptidni lanac. Nakon kidanja tri Glc i jednog Man ostatka, protein se prebacuje na Golgijevu aparatu (GA) gdje glikani gube različit broj Man ostataka i dobivaju kompleksnije strukture procesom terminalne glikozilacije (Breitling i Aebi, 2013). Razlikujemo tri tipa zrelih N-glikana: visoko manozni (ne prolaze kroz proces terminalne glikozilacije), hibridni (u kojima su samo Man ostaci povezani s šest granom jezgre, a kompleksni nastavci su na tri grani) i kompleksni (s različitim kombinacijama ostataka Man, GlcNAc, N-acetilgalaktozamina (GalNAc), fukoze (Fuc) i sijalinske kiseline (N-

acetilneuraminska kiselina, NeuAc)). Koncenzusna aminokiselinska sekvenca N-glikozilacije je Asn-Xaa-Ser/Thr, gdje je Xaa bilo koja aminokiselina osim prolina (Pro), a treonin (Thr) je češći od serina (Ser). Rjeđa je sekvenca sa cisteinom (Cys) Asn-Xaa-Cys (Rudd i sur., 2017).

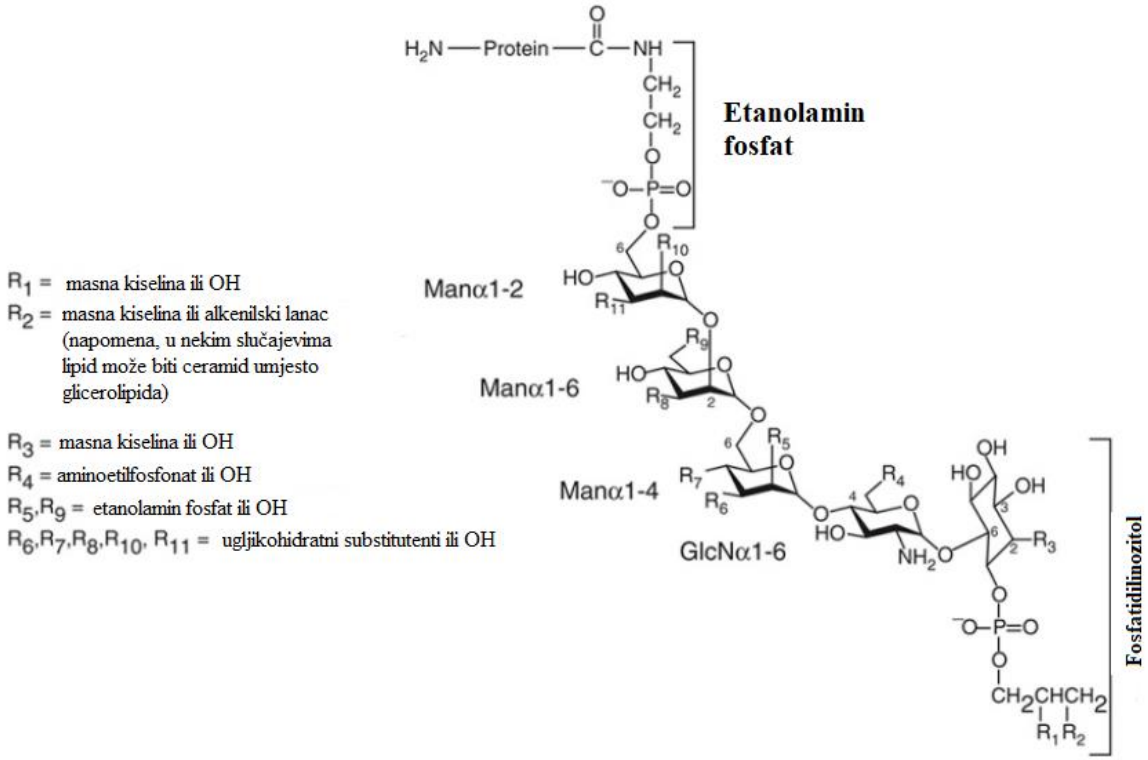
U drugom najčešćem obliku glikozilacije, O-glikozilaciji, akceptor je hidroksil kisik koji se može nalaziti na serinu, treoninu, tirozinu (Tyr), hidroksilizinu ili hidroksiprolinu (Gemmill i Trimble, 1999). O-glikozilacija sekrecijskih i membranskih proteina je posttranslacijska modifikacija koja se događa u cis-Golgijevom odjeljku nakon N-glikozilacije i smatanja proteina. Važna je za lokalizaciju i prometovanje proteina, njihovu topljivost i antigenost te za međustanične interakcije. Gradi se postupnim dodavanjem šećera. Najčešći tip O-glikozilacije sekrecijskih i membranskih proteina sisavaca je dodavanje terminalnog GalNAc, tzv. mucinski tip glikana. On se dalje može produljiti galaktozom (Gal), GlcNAc ili s oboje dajući osam učestalih struktura koje se često dalje proširuju dodavanjem tri sijalinske kiseline. Osim mucinskog tipa O-vezanih glikana, proteini sisavaca također mogu imati Man, Fuc, Glc, Gal i ksilozne (Xyl) terminalne veze. Neki jezgreni proteini imaju jednostavne O-vezane glikane u kojima se može naći jedna molekula N-GlcNAc ostatka vezanog za Ser ili Thr. Ta modifikacija ima važnu ulogu podešavanja biološke aktivnosti intracelularnih proteina i može biti predmet natjecanja glikozilacije s fosforilacijom (Rudd i sur., 2017).

Osim O- i N-glikozilacije koje predstavljaju većinu takvih reakcija, postoje i druge klase reakcija koje su nešto rjeđe. C-vezani glikani (manoze) su kovalentno vezani za ugljikov atom triptofanskog (Trp) bočnog lanca proteina. Predložena su dva signala manozilacije Trp-Xaa-Xaa-Trp (gdje su jedan ili oba Trp manozilirana) i Trp-Ser/Thr-Xaa-Cys.

S-vezani glikani se vežu preko bočnog lanca Cys. Takvo vezanje je rijetko, a prvo je zapaženo u ljudskim i bakterijskim peptidima. Glikacija je neenzimsko dodavanje reducirajućih šećera na N-krajeve proteina (Maillardova reakcija). Ti se šećeri dalje modificiraju i postaju krajnji produkti napredne glikacije (engl. *advanced glycation endproducts*, AGE) i javljaju se u raznim stanjima poput dijabetesa tipa dva, raka, ateroskleroze, Parkinsonovoj i Alzheimerovoj bolesti (Vistoli i sur., 2013.).

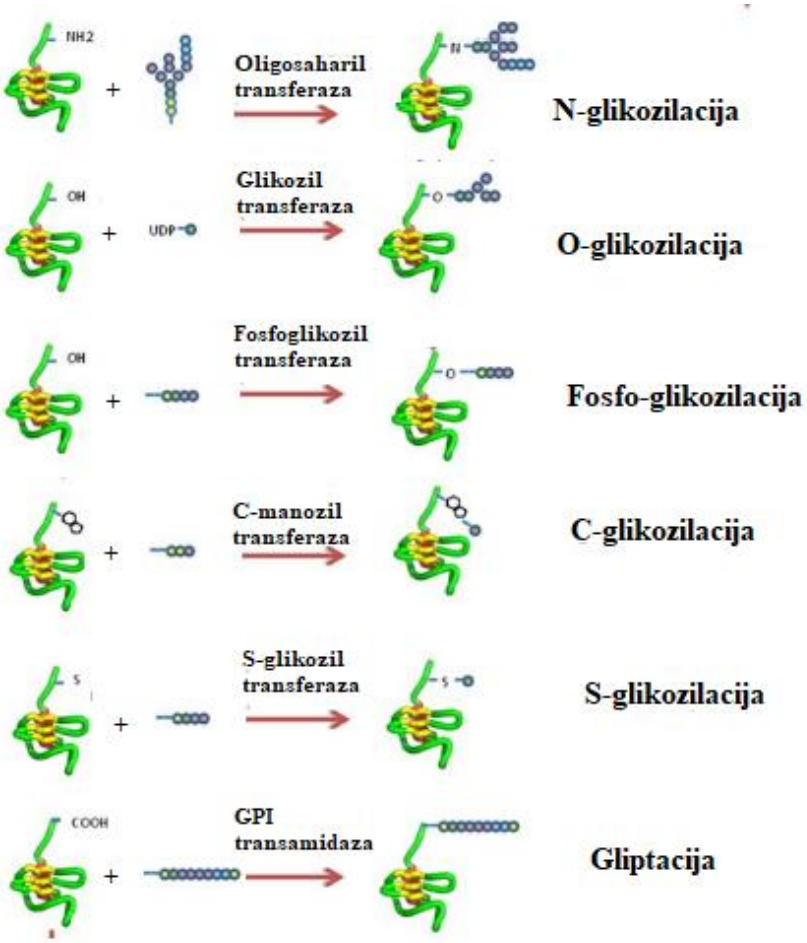
GPI sidra vežu proteine na lipide pomoću glikozidnih veza. Većinom se nalaze na vanjskom sloju lipidnog dvosloja okrenuti prema izvanstaničnom okolišu. Podstruktura Manα1-4GlcNα1–6mio-inozitol-1-P-lipid je univerzalni znak GPI sidra (Slika 1). Interakcije

GPI sidra i proteina su jedinstvene veze ugljikohidrata s proteinima jer protein nije vezan na reducirajući kraj šećera GPI oligosaharida, nego je terminalni glukozaminski (GlcN) ostatak vezan α1-6 vezom s fosfatnom grupom fosfatidil inozitola (Rudd i sur., 2017). Distalni nereducirajući Man ostatak je vezan za protein putem etanolamin fosfata između C6 hidroksilne grupe Man i α-karboksilne grupe karboksi terminalne aminokiseline. Izvan zajedničke jezgre GPI sidra su vrlo različita (Slika 1).



Slika 1. Prikaz opće strukture GPI sidra (preuzeto i prilagođeno s https://www.ncbi.nlm.nih.gov/books/NBK453072/figure/ch12.f1/?report=objectonly)

Fosfoglikani su uglavnom zapaženi u parazita (poput lišmanija i tripanosoma) (Sacks i sur., 2000). Karakterizirani su vezama između glikana i serina ili treonina fosfodiesterskim vezama. U nekih vrsta poput lišmanija ovo je najčešći oblik glikozilacije (Ilg, 2000). Takav oblik glikozilacije je od velike važnosti tim organizmima jer im omogućuje zaštitu od komplementa i promovira agregaciju parazita na domaćina. Također uključuje prebacivanje prethodno napravljenog fosfoglikana s membranske molekule putem fosfoglikoziltransferaze (Slika 2).



Slika 2. Vrste glikozilacije svrstane u specifične skupine prema prirodi ugljikohidrat-peptid veze

1.3. GLIKOMIKA I GLIKOPROTEOMIKA

Glikomika je grana znanosti koja se bavi definiranjem kompletnog repertoara glikana koje stanica ili tkivo proizvede pod određenim uvjetima u nekom vremenu, lokaciji i okolišu, dok glikoproteomika opisuje kako se glikom pojavljuje na staničnom proteomu (Rudd i sur., 2017). Glikani uključuju slobodne nevezane glikane (kao hijaluronan) i glikokonjugate. Glikokonjugati koje kralježnjaci sintetiziraju uključuju N- i O-glikane, glikozaminoglikane, GPI sidra vezana za proteine i glikane vezane na lipide. Naziv se sastoji iz naziva za šećere (gliko-) i omika (od već uspostavljenog nazivlja područja poput genomika, proteomika).

Koliko je velik broj glikoproteina govori činjenica da sačinjavaju većinu membranskih i izlučenih proteina, gotovo sve nuklearne i proteine koji vežu deoksiribonukleinsku kiselinu

(DNA), citoplazmatske enzime uključene u metaboličku regulaciju i neke mitohondrijske proteine. Glikokonjugati kralježnjaka su građeni od devet šećera: Glc, GlcNAc, Gal, GalNAc, Man, Fuc, D-glukuronske kiseline (GlcA), Xyl i NeuAc, te kao deseti šećer L-iduronska kiselina (IdoA) koja se stvara unutar prethodno sintetiziranih glikozaminoglikana. Kod razmatranja glikoproteina važno je naglasiti da je trenutno poznato devet aminokiselina koje mogu biti glikozilirane u prirodi (Asn, Arg, Ser, Thr, Tyr, Trp, Cys, hidroksilizin i hidroksiprolin) (Cummings i Pierce, 2014). No to ne predstavlja jedini problem u glikomici. Kompleksnost biosintetskih putova glikana i glikanskih struktura (u smislu njihove razgranatosti i mogućnosti modifikacije, za razliku od linearne prirode DNA i proteina) te njihova velika dinamičnost su također problemi pri radu s glikanima.

Velika raznolikost glikana u višim organizmima je popraćena velikim spektrom važnih uloga koje ti glikani vrše. Tako prezentiran glikom varira ovisno u kojoj stanici je eksprimiran i njenom stanju, ali i vrstama organizama (glikomi biljaka i prokariota su znatno drukčiji od glikoma sisavaca). Glikokonjugati vrše svoje biološke funkcije kompleksnim molekularnim mehanizmima koji uključuju mnogobrojne procese. Njihovi mehanizmi reguliraju ekspresiju i konformaciju glikokonjugata kroz direktno glikansko prepoznavanje (preko glikan vezujućih proteina, GBP) i indirektne doprinose glikana. Direktni doprinosi uključuju adheziju stanica, interakcije staničnog matriksa, staničnu signalizaciju, glikoproteinsko smatanje i označavanje za specifične organele. Indirektni mogu uključivati konformaciju glikoproteina, stabilnost i oligomerizaciju (Cummings i Pierce, 2014).

Uz prethodno navedeno, glikani se kao glikoproteini nalaze na površini stanica i igraju važnu ulogu u bakterijskom i viralnom prepoznavanju, inhibiciji proliferacije stanica, regulaciji cirkulacije i invaziji stanica (važno kod patofizioloških mehanizama u karcinomu). Također, povezani su s nekim nasljednim bolestima, imaju ulogu u prirođenoj imunosti te gotovo svim biološkim regulatornim putovima. Na površini stanica su toliko izraženi da one mogu sadržavati preko deset milijuna glikana vezanih na Asn i Ser/Thr bočne lance s koncentracijama terminalnih šećera koje se mogu približiti koncentracijama od stotinu milimola po litri (Cummings i Pierce, 2014). Unatoč velikoj zastupljenosti glikana u sisavcima, njihove uloge i organizacija su još uvijek većinom nepoznate, slično kao i njihove interakcije s proteinima (u usporedbi s proteomikom gdje su interakcije protein-protein i protein-nukleinska kiselina procijenjene u stotinama tisuća, s preko 70% proteina enkodiranih s barem jednom detektiranom domenom koja može stupiti u kontakt) (Liu i sur., 2012).

Glikomika još nije dovoljno napredovala da procijeni koliki broj protein-glikan interakcija ima (veličinu protein-glikan interkatoma) (Cummings i Pierce, 2014). One se i dalje svakodnevno otkrivaju te su potrebni mehanicistički uvidi kako bi se definirali proteinski motivi koji upravljaju glikanskim prepoznavanjem. Glikomika kao znanost ima veliki izazov zbog kompleksne i dinamične prirode glikoma (počevši već od same strukture šećera koja ima veliku mogućnost grananja u odnosu na proteine i nukleinske kiseline, zbog čega je mogući broj kombinacija iznimno velik). Za razumijevanje njihovih funkcija i doprinos višeg reda potrebne su fiziološke studije i identifikacija anaboličkih i kataboličkih puteva u organizmu. Za primjer možemo uzeti situaciju u kojoj dolazi do izostanka ekspresije jedne glikozil transferaze, zbog koje se određeni šećer ne može prenijeti na glikoprotein te zbog toga promjena ekspresije samo jedne glikozil transferaze dovodi do promjene glikozilacije niza glikoproteina. Osim toga, glikom je osjetljiv na razine egzogenih nutrijenata i metaboličke cikluse. Stoga, kombinacije studija u jednostaničnim sustavima ne mogu predvidjeti viši red funkcija. Nemogućnost predviđanja struktura koje se mogu očekivati na pojedinim aminokiselinama svakog glikoproteina dodatno otežava razumijevanje i poznavanje glikoma.

1.4. ALATI ZA DETEKCIJU I KARAKTERIZACIJU GLIKOPROTEINA

Kako raste broj informacija i znanje o važnosti glikana i glikoproteina u različitim bolestima i stanjima tako se pojavljuje potreba za razvijanjem novih i boljih metoda i analitičkih strategija za njihovo otkrivanje. To je potrebno kako bi se bolje razumjela njihova raznolika struktura i biokemija. Potrebne su tehnike koje mogu otkriti i proteinski i oligosahardini dio strukture. Glikani mogu biti istraživani na različitim razinama koje se mogu podijeliti u tri različite klase.

Glikoprofiliranje (još se naziva *fingerprinting* ili *patterning*) je jednodimenzionalno odvajanje kompleksnih glikanskih smjesa kako bi se stvorio otisak koji je specifičan za određeni segment glikoma. Tehnike koje se koriste u ovim analizama su tekućinska kromatografija visoke djelotvornosti (HPLC), kapilarna elektroforeza (CE) i spektrometrija masa (MS) (Rudd i sur., 2017).

Glikanska klasna karakterizacija koristi tehnologije za razdvajanje glikanskih smjesa prema tipu glikana. Primjer takve analize je MS odvajanje di-, mono- i neglikoziliranih

imunoglobulina G ili slabe anionske tekućinske kromatografije (LC) glikana u neutralne, mono-, di-, tri- i tetrasijalizirane strukturne tipove. Ovaj pristup je dobar za pokazivanje kritičnih osobina i prikaz kvantitativnih odnosa različitih glikanskih klasa (Rudd i sur., 2017).

Treća i posljednja klasa glikomske analize je detaljna strukturalna analiza. Ona zahtijeva određivanje monosaharidnih sekvenci i modifikacija, anomerizacije i veze glikana unutar genoma. U ovoj detaljnoj analizi obično su potrebne ortogonalne tehnologije za dodjeljivanje preliminarne strukture i za kasniju konačnu potvrdu. Primjer takve analize je separacija anionske izmjene u različito nabijenih glikanskih klasa koja mora biti nadopunjena tekućinskom kromatografijom hidrofilnih interakcija za odvajanje svake pojedine klase (Rudd i sur., 2017).

Niti jedna tehnika sama po sebi nije dovoljna da odredi sve aspekte glikoma ili glikoproteina. Postoji nekoliko pristupa analizi, a najčešće se koriste u kombinaciji. Korištenje tih tehnika omogućuje sastavljanje strukture glikoma *botom up* (od repertoara individualne stanice) ili *top down* (od analize globalne ekspresije u tkivima) smjerom (Smith i Kelleher, 2013.). Unatoč tome, istraživanje glikoma je otežano zbog heterogenosti njihovih veznih mjesta i njihove velike strukturne različitosti što je velika razlika u usporedbi s analizom genoma gdje se vrlo brzo može analizirati veliki broj molekula.

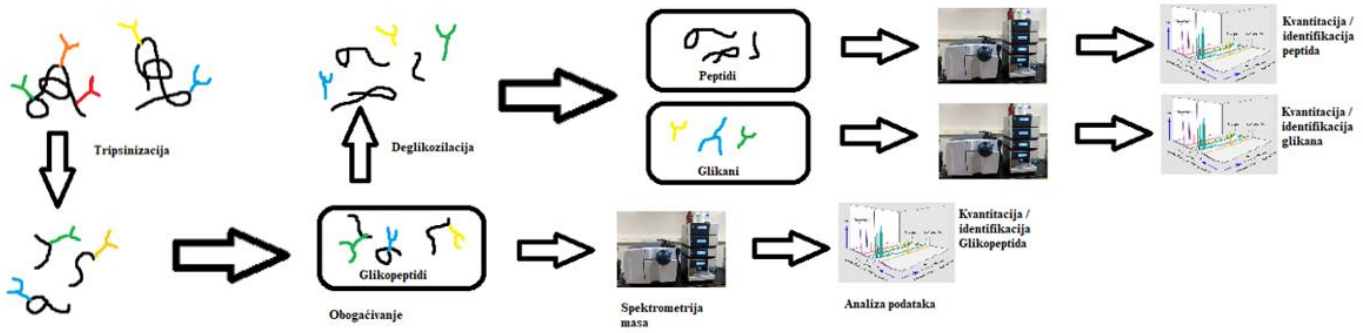
Za detekciju glikoproteina mogu se koristiti lektini, molekulske vrste s vrlo visokim i specifičnim afinitetom za glikanske strukture na proteinima. Nalazimo ih često u organizmima kao dio ER-a gdje pomažu pri smatanju proteina i u međustaničnim interakcijama te patogen-stanica interakcijama. Iako antiglikan antitijela također mogu vezati šećere, lektini se češće koriste zbog niže cijene, bolje karakterizacije i veće stabilnosti od antitijela (Rudd i sur., 2017). Kao i antitijela mogu biti konjugirani s raznim obilježavajućim molekulama. Lektini se koriste za identifikaciju glikoproteina, njihovu purifikaciju, karakterizaciju glikokonjugata na površini stanica te analizu djelovanja glikotransferaza i glikozidaza.

Glavna tehnika koja se danas koristi za analizu individualnih glikana u malim količinama je MS. U praksi se uzorak obogaćen glikoproteinima priprema iz staničnog lizata. U ovom slučaju se primjerice mogu specifično enzimski ili kemijski otpustiti N- ili O-glikani te enzimatski ili kemijski razdvojiti putem HPLC-a i sekvencirati. Važna prednost korištenja metode MS je mogućnost paralelne analize više glikana. Unatoč tome, korištenje MS-a može

dovesti do propuštanja drugih bitnih modifikacija poput sulfatacije ili O- acetilacije, ovisno o primijenjenim tehnikama (Pan i sur., 2011).

Jedan od pristupa analize glikoproteina putem MS-a uključuje četiri koraka. Obogaćivanje glikoproteina ili glikopeptida, multidimenzionalnu separaciju tekućinskom kromatografijom, tandem MS (MS/MS) te detaljnu bioinformatičku analizu dobivenih podataka. Slika 3. prikazuje korake glikoproteomske analize pri čemu u prvom koraku tripsinizacijom nastaju glikopeptidi koji se mogu analizirati direktno kombinacijom LC i MS/MS. Glikopeptidi se također mogu dodatnim korakom deglikozilacije rastaviti na peptide i glikane koji se nakon procesa obogaćivanja analiziraju metodom LC-MS/MS. Analizom dobivenih podataka moguće je kvantificirati tj. identificirati analizirane molekule (Pan i sur., 2011).

Ovisno o vrsti eksperimenta, glikoproteini se mogu odvajati za MS s ili bez kidanja glikana s molekule djelovanjem Endo-β-N-acetilglukozaminidaze H (endoglikanaze H ili endo H) ili Peptid-N(4)-(N-acetil-beta-glukozaminil) asparagin amidaza (PNGaze F). Kvantitativna komparativna analiza glikoproteoma se može izvesti diferencijalnim bilježenjem sa stabilnim izotopima aminokiselina u reagensima staničnih kultura (SILAC). Nadalje, MS selektiranim praćenjem reakcija (SRM) može biti izvedena na ciljnim glikoproteinima koristeći izotopno obilježene teške referentne peptide (Hossain i sur., 2011).



Slika 3. Slijed glikoproteomske analize

Cjelokupna strukturalna analiza glikana može naići na problem s molekulama koje imaju sličnu masu, a različitu strukturu. Takve strukture koeluiraju i mogu zahtijevati detaljnu anotaciju MS/MS spektra. Stoga se glikanima koji se analiziraju daju preliminarne strukture

koje moraju biti potvrđene barem jednom ortogonalnom tehnologijom. Mnogi bioinformatički alati se razvijaju kako bi se razriješio ovaj problem (Rudd i sur., 2017).

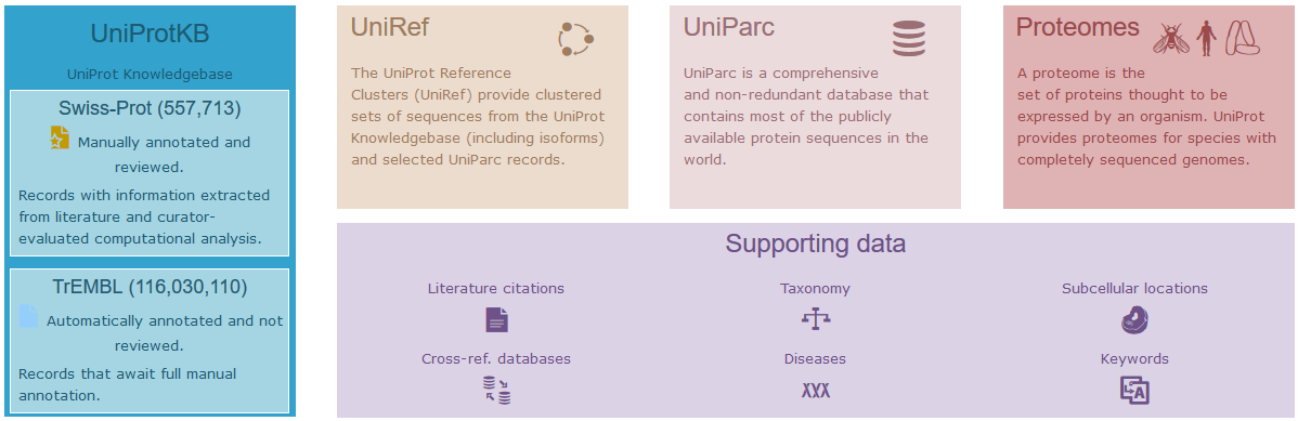
Zbog velike količine dobivenih podataka, istraživanja bioloških uloga glikana nužno ovise o dostupnosti bioinformatičkih alata i baza podataka. Takve baze podatka moraju biti arhivirane, organizirane, imati mogućnost pretraživanja i anotacija te biti povezane s drugim bitnim informacijama o genomu i proteomu. Osim baza podataka, postoje alati koji pomažu pri interpretaciji podataka dobivenih MS analizama kao što su Glycomod, Glycoworkbench, Glycanminertool i GRITS.

Baza podataka je organizirana skupina podataka koja je pohranjena i kojoj se pristupa elektronički. One se intenzivno primjenjuju u informacijskim znanostima pa sličnu važnost imaju i u istraživanjima živih organizama. Baze podataka su glavni način pohranjivanja velike količine informacija koje znanstvenici skupljaju u svojim istraživanjima. Njihova važnost raste kako raste broj podataka i znanja o pojedinim sekvencama nukleinskih kiselina, proteina i znanstvenih radova.

Prema tome, baze podataka možemo podijeliti prema predmetu interesa. Primjeri takve podjele uključuju baze proteina, aminokiselina, znanstvenih radova, i sl. Kada govorimo o bazama podataka koje sadrže informacije o biološkim molekulama (nukleinskih kiselina, proteina), možemo ih podijeliti na primarne i sekundarne. Primarne baze podataka sadrže originalan sadržaj koji je dodan od strane istraživača, dok se sekundarne izvode iz primarnih baza te se u njima često vrši proces organizacije i selektiranja podataka. Primjeri primarnih baza koje se koriste u bioinformatici su ENA (engl. *European Nucleotide Archive*), GenBank, GlyTouCan, dok su od sekundarnih baza poznatije RefSeq, InterPro, UniProt, Cazy i UniCarbKB.

UniProt je visoko kvalitetna, stabilna, opsežna baza proteinskih sekvenci s mnogobrojnim referencama i sučeljima za pretraživanje podataka sadržanih u bazi. Dio je UniProt baze podataka koja se sastoji od SwissProt-a (ručno bilježene i recenzirane baze podataka) te od TrEMBL-a (automatski bilježene i nerecenzirane baze). SwissProt i TrEMBL zajedno čine UniProt bazu znanja (engl. *UniProt Knowledgebase*, UniProtKB) koja sadrži sve zabilježene proteinske sekvence i dio je UniProt arhive (UniParc) koja sadržava sve dostupne proteinske sekvence. UniProt također sadržava i UniRef, bazu podataka koja uklanja redundantne podatke za postizanje potpunog pokrića sekvenci na tri rezolucije preklapanja

(preklapanja na 100%, 90% i 50%) i time daje veće brzine pretraživanja te smanjuje pristranost određenim sekvencama (Slika 4).



Slika 4. Organizacija UniProt baze podataka (preuzeto s https://www.uniprot.org/)

SwissProt osnovan je 1986. godine u suradnji Švicarskog instituta za bioinformatiku (SIB) i Europskog instituta za bioinformatiku (EBI) kao dokumentirana, referencirana, neredundantna bilježena baza podataka. Godine 1996. dodan je TrEMBL, računalno automatski bilježeni unosi izvedeni translacijom kodirajućih sekevenci EMBL (engl. *European Molecular Biology Laboratory*) baze podataka ne dodajući one koje su prethodno već postojale u SwissProt-u. Proteini iz TrEMBL-a mogu postati dio SwissProt-a nakon prolaska kroz ručno bilježenje i detaljnu dokumentaciju podataka i znanstvenih istraživanja proteina.

Nakon što su baze stvorene nužno je napraviti alate koji će ih moći uspješno analizirati i pomagati s interpretacijom. Takvi alati se mogu razvijati u raznim programskim jezicima, a trenutno je najpopularniji Python. Python je interpretirani programski jezik visoke razine. Stvorio ga je Guido van Rossum 1990. godine (prva verzija puštena je 1991.), a dobio je ime po Britanskoj televizijskoj seriji *Monty Python's Flying Circus*. Python je vrlo fleksibilan programski jezik. Dopušta razne stilove programiranja poput objektno orijentiranog, strukturnog i aspektno orijentiranog, dok su logičko programiranje i dizajn ugovorom (*Design by contract*, DbC) podržani putem ekstenzija.

Python je interpretirani, objektno orijentirani programski jezik. On inkorporira module, iznimke, dinamičko tipiziranje, vrlo visoku razinu podataka dinamičkoga tipa i klase. Python

se ističe među sličnim jezicima zbog vrlo jasne i konzistentne sintakse jezika (https://docs.python.org/3.6/faq/general.html). Python dolazi s velikom standardnom bibliotekom koja pokriva razna područja (https://docs.python.org/3.6/faq/general.html). Python je popularan u raznim granama industrije i znanosti, a to duguje jednostavnosti, fleksibilnosti i održivosti svoga koda. Osim toga njegova velika biblioteka, dinamičko tipiziranje omogućuju brzi razvoj novih aplikacija Pythona (Akeret i sur., 2014). Filozofija jezika se može opisati sa sljedećim aforizmima: „Lijepo je bolje nego ružno; eksplicitno je bolje nego implicitno; jednostavno je bolje nego kompleksno; kompleksno je bolje nego komplicirano; linearno je bolje nego upetljano; čitljivost je bitna; posebni slučajevi nisu dovoljno posebni da prekrše pravila o praktičnosti“. (https://www.python.org/dev/peps/pep-0020/). Do sada je Python prošao kroz tri verzije, Python 2.0 je izašao 2000. godine, a 3.0 2008. godine. Trenutna verzija je 3.7.2.

Objektno orijentirano programiranje je programska paradigma bazirana na konceptu objekta koji sadržava podatke u formi atributa, a kod u formi metoda. To omogućuje metodama da pristupe objektu i modificiraju atribute s kojima su na neki programski definirani način povezani. Glavna karakteristika takvog programiranja je da su objekti međusobno povezani i mogu međusobno dijeliti atribute i metode (Kindler i Krivy, 2011). Jezici koji podržavaju objektno orijentirano programiranje obično koriste sustav nasljeđivanja koda u obliku klasa ili prototipova. Programi koji koriste klase imaju dva glavna koncepta: klase (skup definicija oblika podataka i dostupnih procedura za dani tip klase ili objekta) i objekte (instance klasa).

Za izradu dobrog programa nužno je dobro aplikacijsko programsko sučelje (engl. *aplication programing in terface*, API). U računalnom programiranju API je set subrutina, definicija, komunikacijskih protokola i alata za izgradnju programa. Generalno gledano to je set jasno definiranih metoda komunikacije između različitih komponenti programa. Ukoliko je API dobro dizajniran, program je lako razvijati i održavati. Dobar API treba posjedovati sve temelje koji su nužni za rad programa. API se može pisati za razne sustave, bazirane na internetu, operacijske sustave, sustave baza podataka, računalni hardware ili za programske biblioteke. Uz API je nužno i stvoriti potrebnu dokumentaciju koja objašnjava čemu program služi i kako ga treba koristiti.

2. OBRAZLOŽENJE TEME

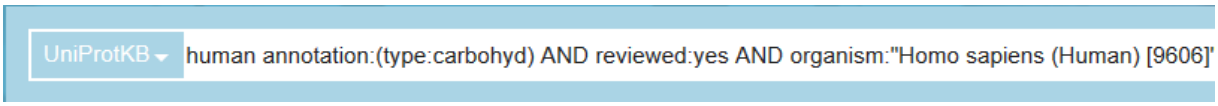
Pretraživanje proteina putem UniProt web sučelja se može činiti naporno i sporo te dolazak do informacija otežan zbog njihove količine. Glavni razlog nedostatnosti korisničkog sučelja baze je fokus UniProt baze na pojedinačne proteine, dok je pospremanje skupina većih od 400 proteina nemoguće. To stvara veliki problem jer je broj proteina u bazi podataka iznimno velik (za ljudski proteom je opisano 20 417, od kojih su 4 622 glikoproteini). Zbog toga je cilj ovog diplomskog rada napraviti alat koji će taj proces pojednostaviti i omogućiti sistematičniji pristup željenim informacijama. Za pretraživanje podataka koristi se Swissprot baza podataka. Informacije su ponuđene u više vrsta prikaza, a jedan od njih je i XML (engl. *Extensible Markup Language*) format korišten u ovom radu. Takvi podaci se mogu obrađivati programima, pretraživati i selektirati. Cilj ovoga rada je napraviti alat koji brzo i jednostavno nalazi jedan ili više proteina prema definiranim kriterijima (u ovom slučaju prema jedinstvenom pristupnom broju proteina, engl. *accession number*) i nalazi informacije od interesa istraživača poput imena, sekvence, poznatih izoformi, glikoproteina, mase, nusprodukata enzimske obrade (primarno u programu razvijenom za ovaj rad, tripsina), vrsta i mjesta glikozilacije sekvence i potencijalna glikozilacijska mjesta izoformi. Potrebno je prikazati izabrane informacije bilo u računalno čitljivom obliku za daljnju obradu ili u tekstualnom obliku čitljivom čovjeku.

3. MATERIJALI I METODE

3.1. PRISTUP PODACIMA BAZE PODATAKA I OSNOVNA STRUKTURA PROGRAMA

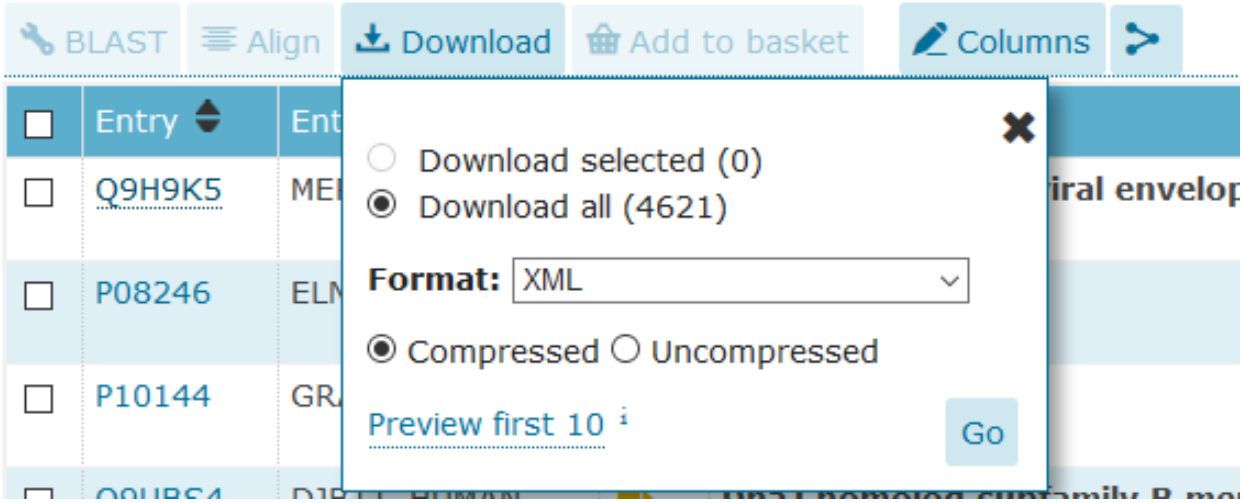
Za izradu ovog diplomskog rada korištena je UniProt baza podataka dostupna na https://www.uniprot.org/ i programski jezik Python.

Prvi korak u izradi programa je pronalazak odgovarajućeg seta proteina baze podataka. Za potrebe ovoga rada odabrana je UniProtova baza humanog glikoproteoma. Pri pretraživanju baze podataka potrebno je filtrirati bazu samo za anotirane proteine koji su dio Swiss-Prot-a, organizma Homo sapiens i da imaju anotaciju za posttranslacijske modifikacije glikozilacije (točan kod prikazan na Slici 5).



Slika 5. Usmjerena pretraga UniProtove baze podataka za Humane glikoproteine unutar Swiss-Prot-a

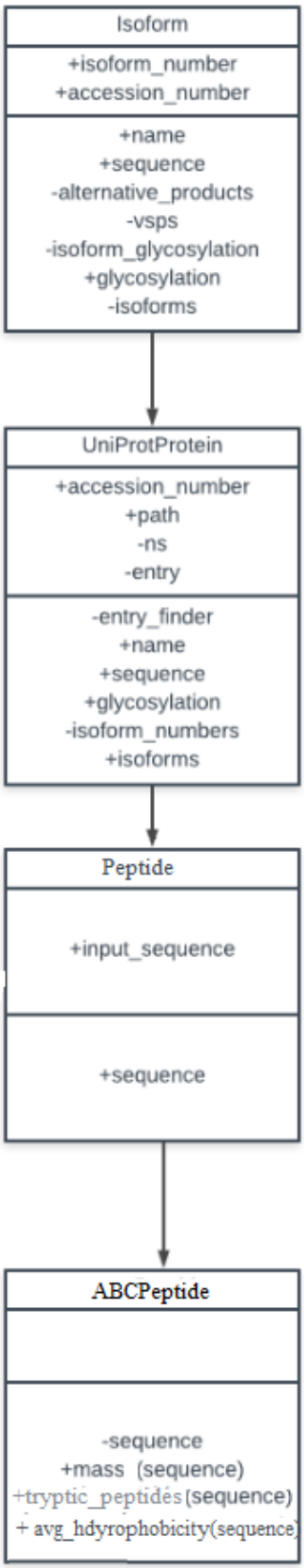
Tako pronađen set proteina je dostupan za preuzimanje u nekoliko oblika, a za ovaj rad baza je preuzeta u XML obliku (Slika 6). Taj XML dokument se dalje obrađuje Python programskim jezikom.



Slika 6. Preuzimanje pronađenih proteina u XML obliku na UniProt stranici

API se bazira na čitanju skinute XML datoteke i raščlambi podataka korištenjem XML *ElementTree* bibilioteke što će biti kasnije opisano.

API je organiziran u tri zasebne klase koje su definirane prema biološkom odnosu proteina i peptida te vrsti podataka poznatima o određenoj molekuli. *ABCPeptide*, *Protein*, *UniProtProtein* i *Isoform* podaci te zasebna funkcija *protein* pomažu stvaranju zbira podataka povezanih sa specifičnim proteinom. Navedene klase su u međusobno hijerarhijskom odnosu prema dijagramu klasa prikazanog na Slici 7. Apstraktna klasa (*ABCPeptide*) je najviša po hijerarhiji. Nakon nje slijedi klasa *Protein*, nakon koje je složena nešto detaljnija klasa *UniProtProtein*, te konačno *Isoform*. API također sadrži i funkciju koja služi kao tvornica za formiranje proteina.



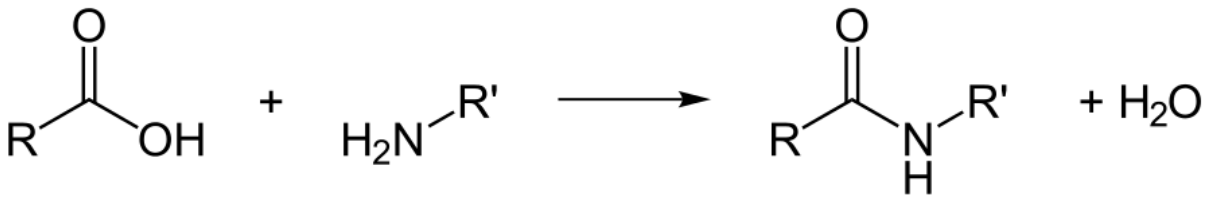
Slika 7. Dijagram klasa API diplomskog rada

3.2. ABCPEPTIDE

ABCPeptide je najviša klasa i definirana je kao apstraktna klasa i kao takva nasljeđuje metode ABC (engl. *abstract base class*) biblioteke.

ABC biblioteka pruža potrebnu infrastrukturu za definiranje apstraktne klase unutar Pythona. Zbog toga je namijenjena za držanje osnovne infrastrukture zajedničke svim klasama (ili u slučaju ovog rada peptidima) koje nasljeđuju njezine informacije, dok je metoda *sequence* definirana kao apstraktna metoda. Sekvenca peptida je opisana kao apstraktna jer je metoda *ABCPeptide* najviša klasa i predstavlja neki općeniti slučaj te nije namijenjena za implementaciju. Ta metoda poprima vrijednost tek kada se primijeni za poseban slučaj peptida i služi za uspostavljanje hijerarhije. U primjeru ovoga rada je to protein. Stvaranjem instance proteina se uspostavljanjem ovakve strukture uspostavlja kao i peptid. Ono što je namijenjeno za korištenje su dvije metode koje se nalaze unutar klase.

Prvo svojstvo nazvano je *mass*, a računa mase peptida na temelju njihovog aminokiselinskog sastava. To radi na način da preuzima podatke o atomskom sastavu aminokiselina iz rječnika *amino_atoms*. Podaci rječnika su pohranjeni na sljedeći način (npr. 'A': (3, 5, 1, 1, 0)). Python rječnik svoje podatke sprema u obliku ključ: vrijednost. U primjeru ovog rada ključ predstavlja slovo koje predstavlja aminokiselinu i vrijednosti broja pojedinih atoma točno određenim redoslijedom ugljik (C), vodik (H) dušik (N), kisik (O), sumpor (S) . Broj atoma odgovara aminokiselinskom ostatku radi jednostavnosti računanja, dok je za krajeve proteina/peptida dodana jedna molekula vode na njihovim krajevima (-H i -OH). To je napravljeno kako bi se zaračunalo otpuštanje molekule vode pri nastanku amidne veze (Slika 8). Ova metoda prolazi kroz niz slova ili *string* sekvence te zbraja za svaku aminokiselinu broj atoma koje sadrži. Za slučaj da se metodi zada aminokiselina koja ne postoji, ona će izbaciti *ValueError* koji će javiti da joj je zadana kriva aminokiselina. Do konačnog rezultata ukupne mase proteina metoda dolazi računanjem zbroja umnožaka ukupnog broja pojedinih atoma aminokiselina s njihovim atomskim brojem (pronađenim na http://ciaaw.org/atomic-weights.htm, Slika 9).



Slika 8. Nastajanje amidne veze

(preuzeto s https://bs.wikipedia.org/wiki/Peptidna_veza#/media/File:Amidbildung.svg)

```
@property
def mass(self):

        amino_atoms = {
                'A': (3, 5, 1, 1, 0, 0),
                'C': (3, 5, 1, 1, 1, 0),
                'D': (4, 5, 3, 1, 0, 0),
                'E': (5, 7, 3, 1, 0, 0),
                'F': (9, 9, 1, 1, 0, 0),
                'G': (2, 3, 1, 1, 0, 0),
                'H': (6, 7, 1, 3, 0, 0),
                'I': (6, 11, 1, 1, 0, 0),
                'K': (6, 12, 1, 2, 0, 0),
                'L': (6, 11, 1, 1, 0, 0),
                'M': (5, 9, 1, 1, 1, 0),
                'N': (4, 6, 2, 2, 0, 0),
                'P': (5, 7, 1, 1, 0, 0),
                'Q': (5, 8, 2, 2, 0, 0),
                'R': (6, 12, 1, 4, 0, 0),
                'S': (3, 5, 2, 1, 0, 0),
                'T': (4, 7, 2, 1, 0, 0),
                'V': (5, 9, 1, 1, 0, 0),
                'W': (11, 10, 1, 2, 0, 0),
                'Y': (9, 9, 2, 1, 0, 0)
                'U':  (3, 5, 1, 1, 0, 1)
        }

        carbons = 0
        hydrogens = 2
        oxygens = 1
        nitrogens = 0
        sulfurs = 0
        seleniums = 0

        sequence = self.sequence.upper()

        for letter in sequence:
                carbon, hydrogen, oxygen, nitrogen, sulfur, selenium = amino_atoms[letter]
                carbons += carbon
                hydrogens += hydrogen
                oxygens += oxygen
                nitrogens += nitrogen
                sulfurs += sulfur
                seleniums += selenium

        total_mass = carbons * 12.0116 + oxygens * 15.99977 + nitrogens * 14.00643 + sulfurs * 32.059 + hydrogens *
1.00784 + seleniums*78.971

        return total_mass
```

Slika 9. Računanje molekulske mase peptida

Drugo svojstvo koje se nalazi unutar *ABCPeptide* klase zove se *tryptic_peptides*. To svojstvo oponaša djelovanje serinske proteaze tripsina, a metoda kao i *mass* korištenjem

sekvence čita njezine aminokiseline i reže tako dane *stringove* proteina nakon lizina i arginina, osim kad su njihovi C-krajevi vezani za prolin. Ta provjera se vrši putem *regular expression (regex)* biblioteke Pythona. Nakon što pronađe mjesto reza, izračunava početnu poziciju reza i završnu poziciju reza molekula i vraća te informacije u obliku klase peptide (Slika 10).

```
@property
def tryptic_peptides(self):
        """This function makes trypsine cuts in a protein for all relevant cuts ( longer than 2 amino acids).

        Parameters
        ----------
        sequence: str

        Returns
        -------
        total_peptide: dict
                Returns dictionary of a order number of a trypsine cut with the cut, start and end position of a cut a
nd the
                mass of the cut.

        """
        tryptic_peptides = []
        pattern = re.compile('([KR]?[^P].*?[KR](?!P))')
        tryptic_list = pattern.split(self.sequence)
        k = 1
        for cut in tryptic_list:
                cut_start = k
                cut_end = k + len(cut) - 1
                if len(cut) > 0:
                        peptide_cut = Peptide(cut)
                        peptide_cut.start, peptide_cut.end = cut_start, cut_end
                        tryptic_peptides.append(peptide_cut)
                        k += len(cut)


        return tryptic_peptides
```

Slika 10. Računanje tripsinskih rezova peptida

Posljednje svojstvo koje ABC peptide ima je *avg_hydrophobicity* koje računa prosječnu hidrofobnost proteina kao što ime i sugreria. Koristi skalu koju su napravili Wolfenden R.V. i sur., a koristi vrijednosti hidracijskog potencijala u kcal/mol pri 25°C. Skala nema definiranu vrijednost za selenocistein (Sec) pa je njemu dodijeljena vrijednost od 0.000 kako ne bi utjecao na rezultat. Algoritam uzima cijelu sekvencu u obzir i računa prosječnu hidrofobnost za cijelu sekvencu te ne uzima smatanje proteina u obzir (Slika 11)

```
@property
def avg_hydrophobicity (self):
        # Amino acid scale: Hydration potential (kcal/mole) at 25°C
        # Author(s): Wolfenden R.V., Andersson L., Cullis P.M., Southgate C.C.F.
        # authors haven't provided hydrophobicity for Sec/U so it was designated with a 0.000
        amino_atoms = {
                'A': (1.940), 'C': (-1.240), 'D': (-10.950), 'E': (-10.200),
                'F': (-0.760), 'G': (2.390), 'H': (-10.270), 'I': (2.150),
                'K': (-9.520), 'L': (2.280), 'M': (-1.480), 'N': (-9.680),
                'P': (0.000), 'Q': (-9.380), 'R': (-19.920), 'S': (-5.060),
                'T': (-4.880), 'V': (1.990), 'W': (-5.880), 'Y': (-6.110),
                'U': (0.000)
        }

        hydrophobicity = 0.000
        sequence = self.sequence.upper()

        for letter in sequence:
                hydrophobicity += amino_atoms[letter]

        return hydrophobicity
```

Slika 11. Računanje hidrofobnosti peptidnih struktura

3.3. PEPTIDE I UNIPROTPROTEIN

Sljedeća klasa opisana ovim programom je klasa *Peptide*. *Peptide* definira sekvencu kao minimalnu informaciju o proteinu. Instance te klase se iniciraju kanonskom sekvencom (najčešćom sekvencom) proteina i nudi opcije obrade te sekvence opisane u *ABCPeptide* klasi. Može se koristiti kada je korisniku poznata sekvenca, ali ne i pristupni broj proteina s UniProta. Kada je korištena može vratiti *tryptic_peptides, mass, sequence* informacije o proteinu od interesa.

Nakon toga slijedi klasa *UniProtProtein* koja nasljeđuje od *Protein* klase. *UniProtProtein* je klasa koja predstavlja anotirani protein UniProt baze podataka. Za svoj rad zahtijeva unos lokacije XML dokumenta koji sadrži bazu podataka i jedinstveni pristupni broj. Ova klasa uzima dani jedinstveni pristupni broj i pretražuje bazu za njega korištenjem ElementTree modula. Taj modul koristi hijerarhijsku organizaciju XML dokumenta da nađe čvorove od interesa. Specifično u slučaju ovog rada traži čvor s nazivom *entry* (koji sadrži jedinstveni unos proteina u bazu podataka) i pretražuje čvor *accession* za dani pristupni broj.

Jednom kada nađe taj pristupni broj, sprema unos o proteinu u varijablu koja je nazvana *entry* (Slika 12).

```
def __init__(self, path, accession_number):
                self.path = os.path.abspath(os.path.join(os.path.dirname(os.path.realpath(__file__)), path))
                self.ns = {"uniprot": "http://uniprot.org/uniprot"}
                self.accession_number = accession_number
                self.entry = self._entry_finder()
                self.protein = None

        def _entry_finder(self):
                parse = ET.iterparse(self.path, events=("start", "end"))
                parse = iter(parse)
                event, root = next(parse)

                for event, elem in parse:
                        if event == "end" and elem.tag.endswith("}entry"):
                                accession_elem = elem.findall("{" + self.ns['uniprot'] + "}" + "accession")
                                if self.accession_number in [i.text for i in accession_elem]:
                                        return elem
                                        root.remove(elem)
```

Slika 12. Pretraživanje XML dokumenta UniPreotove baze podataka s ciljem nalaženja unosa s danim jedistvenim pristupnim brojem

Nakon što je pronađen željeni unos u bazu daljnjim korištenjem *ElementTree* biblioteke moguće je naći unesene podatke o pojedinom proteinu. Program ovog rada koristi *element tree* kako bi našao puno ime sekvence, sekvencu, glikozilacijska mjesta (ukoliko postoje, u suprotnom vraća prazan rječnik), podatke o izoformi te kreira instancu izoforme. Pronalaženje imena, sekvence, glikozilacijskih mjesta informacija o izoformama funkcioniraju na sličan način kao i traženje specifičnog unosa. Primjerice, ako tražimo sekvencu moramo naći čvor pod nazivom sekvenca i dodijeliti pronađeni niz znakova varijabli *sequence* te ju vratiti kada svojstvo *sequence* bude pozvano. Analogno, ukoliko tražimo puno ime proteina moramo naći čvor pod imenom *fullName* i podatak spremljen u njemu. Kod pretraživanja glikozilacijskih mjesta pretražuje se čvor *feature* koji na sebi ima atribut *glycosylation site*. Kada je takav čvor nađen, podaci o glikozilacijskim mjestima se spremaju u rječnik koji za ključ ima broj glikozilacijskog mjesta i informaciju o vrsti glikozilacijskog mjesta, na koji atom i koju aminokiselinu je takav glikan vezan. Ukoliko glikozilacijskih mjesta nema, protein će vratiti prazan rječnik (Slika 13).

```
@property
        def name(self):
                """Returns name of the protein.

                """
                protein_names = self.entry.find("uniprot:protein", self.ns)
                recommended_name = protein_names.find("uniprot:recommendedName", self.ns)
                full_name = recommended_name.find("uniprot:fullName", self.ns).text
                return full_name

        @property
        def sequence(self):
                """Returns canonical sequence of the protein.

                """

                sequence = self.entry.find("uniprot:sequence", self.ns).text
                sequence = ''.join(sequence.split())
                return sequence

        @property
        def glycosylation(self):
                """Returns dictionary of glycosilation sites and their types.

                """
                try:
                        glycosylation_dictionary = {}
                        for feature in self.entry.findall("uniprot:feature[@type='glycosylation site']", self.ns)
:
                                glycosylation_feature = feature.attrib
                                glycosylation_type = glycosylation_feature["description"]
                                location = feature.find("uniprot:location", self.ns)
                                glycosylation_site = location.find("uniprot:position", self.ns).attrib
                                glycosylation_site = int(glycosylation_site["position"])
                                glycosylation_dictionary[glycosylation_site] = glycosylation_type
                except AttributeError:
                        glycosylation_dictionary = {}
                return glycosylation_dictionary
```

Slika 13. Pretraživanje imena, sekvence i glikozilacijskih mjesta proteina unutar XML dokumenta UniProtove baze podataka

Kod stvaranja instance izoforme vrše se dva koraka. Prvi korak je pronalazak brojeva izoformi privatnom metodom *isoform_numbers* nalaženjem čvora *comment* s atributom *alternative products*. Unutar takvih komentara se nalaze čvorovi pod nazivom *isoform* unutar kojh pod čvorom *name* stoje brojevi izoformi poznatih za protein od interesa.

Drugi korak, ukoliko izoforma postoji, izvodi se putem *dunder* (*double under*) metode *__new__* nad klasom *Isoform*. Ta metoda omogućuje davanje kontrole nad kreiranjem nove instance. Ona se poziva prva i odgovorna je za vraćanje instance klase *Isoform*. Ona omogućuje kreiranje instance izoforme samo kada je to zaista potrebno i tako omogućuje brži rad programa. Pri kreiranju takve instance nužno je dati podatke koji su nužni za rad te metode, a to su *entry* koji je pronađen, broj izoforme o kojoj se radi i *namespace* XML dokumenta za pretraživanje čvorova. Kao konačan proizvod te metode se vraća rječnik koji kao ključ ima broj izoforme, a za svoju vrijednost nosi instancu jedne izoforme. Ukoliko izoforma ne postoji, program će vratiti grešku *AttributeError* s porukom *„No isoforms found“* (Slika 14).

```
def _isoform_numbers(self):

                isoform_numbers = []
                comment = self.entry.find("uniprot:comment[@type='alternative products']", self.ns)
                try:
                        for isoform in comment.findall("uniprot:isoform", self.ns):
                                isoform_number = isoform.find("uniprot:name", self.ns).text
                                if int(isoform_number) == 1:
                                        pass
                                else:
                                        isoform_numbers.append(int(isoform_number))
                except AttributeError:
                        isoform_numbers = []
                return isoform_numbers

        @property
        def isoforms(self):
                isoforms = {}
                if self._isoform_numbers() == []:
                        return AttributeError("No isoforms found")
                else:
                        for isoform_number in self._isoform_numbers():
                                isoform = Isoform.__new__(Isoform)
                                isoform.entry, isoform.isoform_number, isoform.ns = self.entry, isoform_numbe
r, self.ns
                                isoforms[isoform_number] = isoform
                        return isoforms
```

Slika 14. Pretraživanje imena, sekvence i glikozilacijskih mjesta proteina unutar XML dokumenta UniProtove baze podataka

3.4. ISOFOROMS

Zadnja klasa koja se nalazi u API-ju je *Isoform* klasa. Nasljeđuje od *UniProtProteina* i u njemu se inicijalizira kako je opisano gore. Kao svoje metode/svojstva sadrži *name*, *sequence*, privatne varijable *_vsps*, *_alternative_products*, *_isoform_glyosylation*,

glycosylation i *isoforms*. Klasa za svoj rad treba lokaciju dokumenta, pristupni broj i broj izoforme. Uz pomoć tih informacija može izračunati sve navedene informacije za jednu izoformu. Ime izoforme se računa iz imena odgovarajućeg UniProt proteina i broja izoforme i izgleda: ime proteina-broj izoforme (Slika 15).

```
class Isoform(UniProtProtein):

        def __init__(self, path, accession_number, isoform_number):
                super().__init__(path, accession_number)
                self.isoform_number = isoform_number

        @property
        def name(self):
            return super().name + "-" + str(self.isoform_number)

        def _alternative_products(self):...

        def _vsps(self):...

        @property
        def sequence(self):...

        def _isoform_glycosylation(self,isoform_number):...

        @property
        def glycosylation(self):...

        @property
        def isoforms(self):
```

Slika 15. Inicijalizacija i pretraživanje imena izoforme

Sekvenca se računa uz pomoć dvije privatne metode *_vsps* i *_alternative_products*. Metoda *_vsps* nalazi popis specifičnih ključeva koji se zovu *vsp* u UniProt XML dokumentu. Oni sadrže sve informacije o specifičnim promjenama koje se događaju imeđu izoformi i kanonske sekvence. Nalaze se korištenjem *ElementTree* biblioteke pod čvorom *feature* s atributom *splice variant*. Svaki *vsp* ključ sadrži informaciju o sekvenci varijacije, mjestu početka varijacije i mjestu završetka varijacije. *_vsps* nalazi te informacije i sprema ih u obliku rječnika s *vsp* kao ključem i pripadajućim setom informacija. Metoda *_alternative products* pretražuje UniProt unos da nađe sve specifične *vsp* ključeve i povezuje ih s

pojedinim izoformama. Nalazi se u *comment* čvoru i vraća rječnik u kojemu je broj izoforme ključ, a vrijednosti su setovi *vspova* (Slika 16). Takva dva rječnika se povezuju u metodi *sequence*. Ta metoda uz pomoć kanonske sekvence, pronađene kako je prethodno opisano, koristi informacije iz *_vsps* i *_alternative products* i od njih sastavlja sekvence izoformi (Slika 17). To radi na način da sortira promjene od zadnje prema prvoj te tim redom vrši promjene na *stringu* kanonske sekvence. Na taj način izbjegava problem različite dužine originalne i promjenjene sekvence. U slučaju da se metoda vrši s početka prema kraju, metoda bi morala uzimati u obzir promjene pozicija aminokiselina u odnosu na razliku dužina između promjenjenih dijelova kanonske sekvence i sekvence izoforme (npr. ako imamo deleciju četiri aminokiseline u izoformi, sve aminokiseline koje dolaze nakon te delecije su četiri mjesta ispred nego što zaista jesu u kanonskoj sekvenci) (Slika 17).

```
def _alternative_products(self):

        isoform_ids = {}
        comment = self.entry.find("uniprot:comment[@type='alternative products']", self.ns)
        try:
                for isoform in comment.findall("uniprot:isoform", self.ns):
                        isoform_number = int(isoform.find("uniprot:name", self.ns).text)
                        ref_id = isoform.find("uniprot:sequence", self.ns).attrib
                        if "ref" in ref_id:
                                isoform_ids[isoform_number]=(ref_id["ref"].split())
        except AttributeError:
                isoform_ids = {}
        return isoform_ids

def _vsps(self):

        ids_information = {}
        for feature in self.entry.findall("uniprot:feature[@type='splice variant']", self.ns):
                feat = feature.attrib

                id_ = feat["id"]

                variation = feature.find("uniprot:variation", self.ns)

                if variation is None:
                        variation = ""
                else:
                        variation = variation.text

                location = feature.find("uniprot:location", self.ns)

                try:
                        begin = int(location[0].get("position"))
                        end = int(location[1].get("position"))
                except IndexError:
                        position = int(location[0].get("position"))
                        begin = position
                        end = position

                ids_information[id_] = (variation, begin, end)
        return ids_information
```

Slika 16. Kreiranje sekvence izoforme

```
@property
def sequence(self):
        id_list = self._alternative_products()[self.isoform_number]
        isoform_information = [self._vsps()[key] for key in id_list]
        isoform_information.sort(key=lambda x: x[1], reverse=True)
        isoform_sequence = super().sequence

        for single_variation in isoform_information:
                variation, index_start, \
                index_end = single_variation
                isoform_sequence = isoform_sequence[:index_start - 1] + variation + isoform_sequence[in
dex_end:]

        return isoform_sequence```
```

Slika 17. Kreiranje sekvence izoforme

Za računanje i prikazivanje glikozilacije izoformi koriste se dvije metode *_isoform_glycosylation* i *glycosylation*. Privatna varijabla *_isoform glycosylation* računa glikozilacijska mjesta iz broja izoforme.

Prvo što *_isoform glycosylation* metoda napravi je lista *vspijeva* za određenu izoformu i sprema ju u varijablu *id_list*. Nakon toga iz nje izvuče sve promjene koje se događaju za željenu izoformu i sprema ih u *variation_information_collection*. Pri računanju joj pomažu *append_sites*, *index_end_last* i korekcijski faktor za poziciju. Nakon što su sve varijable postavljene, metoda prolazi za svaku varijaciju kroz sva glikozilacijska mjesta za tri slučaja: promjena se dogodila na početku prije varijacije, promjena se dogodila između dvije varijacije i promjena se dogodila na kraju. Ukoliko je glikozilacijsko mjesto već dodano, preskače se. Sve tako izračunate promjene koje nisu izbačene iz izoforme se spremaju u rječnik i ispisuju svojstvom *glyocsylation* po istoj logici kao i kod *UniProtProteina*. Za *isoforms* svojstvo je postavljeno da kada se pozove vrati *NotImplementedError* budući da je izoforma izvedena iz *UniProtProtein* unosa (Slika 18). Ako korisnik želi saznati informacije o drugim izoformama za neki protein, može to pronaći pretraživanjem tog proteina.

```
def _isoform_glycosylation(self,isoform_number):
                """Returns dictionary of a secific isoform glycosilation sites and their types.

                This method filters annotated glycosylation sites for a single isoform by checking 3 specific
cases.
                """
                isoform_glycosylation = {}
                id_list = self._alternative_products()[isoform_number]
                variation_information_collection = [self._vsps()[key] for key in id_list]
                appended_sites = []
                index_end_last = 0
                i = 0
                evariation, eindex_start, \
                eindex_end = variation_information_collection[-1]
                for single_variation in variation_information_collection:
                        variation, index_start,\
                        index_end = single_variation
                        i2 = i
                        i = index_end - index_start + 1 - len(variation) + i
                        for site in super().glycosylation.keys():
                                if site not in appended_sites:
                                        if index_end_last == 0:
                                                if site < index_start:
                                                        isoform_glycosylation[site] = super().glycosylation[si
te]
                                                        appended_sites.append(site)
                                        elif site > index_end_last:
                                                if site < index_start:
                                                        appended_sites.append(site)
                                                        sitem = site - i2
                                                        isoform_glycosylation[sitem] = super().glycosylation[s
ite]
                                        if index_end == eindex_end:
                                                if site > index_end:
                                                        appended_sites.append(site)
                                                        sitem = site - i
                                                        isoform_glycosylation[sitem] = super().glycosylation[s
ite]
                        index_end_last = index_end
                return isoform_glycosylation

        @property
        def glycosylation(self):
                return self._isoform_glycosylation(self.isoform_number)

        @property
        def isoforms(self):
                return NotImplementedError()
```

Slika 18. Računanje glikozilacijskih mjesta izoforme

Posljednja stvar koja se može naći u API je funkcija *protein*. To je tzv. tvornička funkcija koja pomaže kreiranju instanci proteina, izoforme, ili *UniProtproteina* ovisno o

unesenim podacima. Ukoliko korisnik unese sekvencu pri kreiranju instance proteina i zada sve ispravne aminokiseline, kreirat će se instanca *Protein* klase. Ukoliko korisnik zada *accession number* (može i *path* ukoliko to specificira, inače će biti zadan standardni *path* dokumenta UniProt baze podataka) kreirat će se instanca *UniProtprotein* klase. Ukoliko je zadan *isoform number* ali i *accession number* bit će kreirana instanca *Isoform* klase (Slika 19).

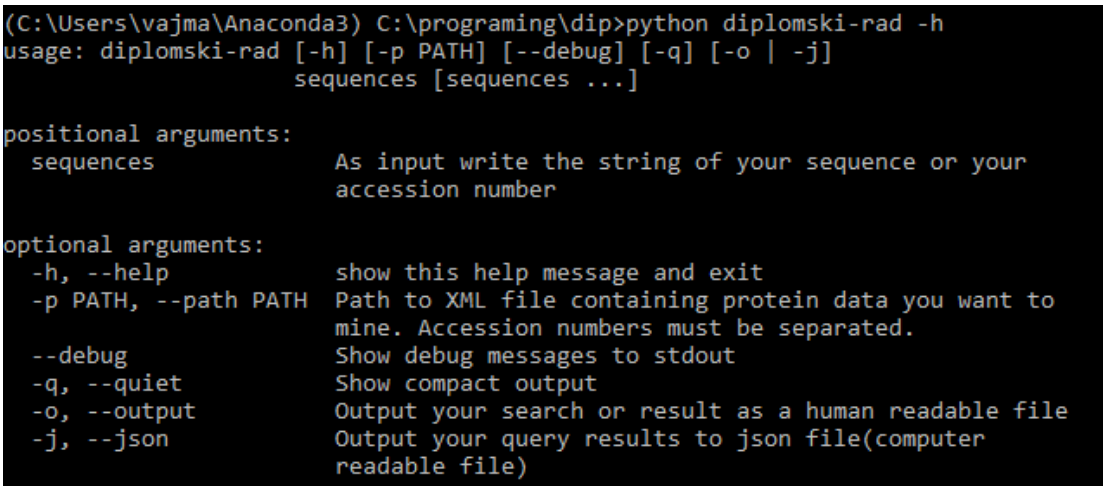
```
def protein(sequence=None, isoform_number=None,  acc=None, xmlfile=None):
        if not xmlfile:
                xmlfile ='gli.xml'
        if not isoform_number:
                if acc:
                        if re.match('[OPQ][0-9][A-Z0-9]{3}[0-9]|[A-NR-Z][0-9]([A-Z][A-Z0-9]{2}[0-9]){1,2}', acc):
                                return UniProtProtein(xmlfile, acc)
                        else:
                                raise ValueError('You have entered a non-valid accession number, please review your se
quence before '

                                                 'trying to process it again.')
                elif sequence:
                        if re.match("[ACDEFGHIKLMNPQRSTVWUY]+$", sequence):
                                return Protein(sequence)
                        else:
                                raise ValueError('You have entered a non-valid sequence, please review your sequence b
efore '

                                                                 'trying to process it again.')
                else:
                        raise ValueError('Sequence or accession number are required!')
        else:
                if acc:
                        return Isoform(xmlfile, acc, isoform_number)
                else:
                        return ValueError('Both isoform number and accession number are required!')
```

Slika 19. Završna funkcija za kreiranje instanci klasa API

3.5. KORISNIČKA SUČELJA

Korištenje programa se vrši putem jednog od dva ponuđena korisnička sučelja. Konzolno korisničko sučelje programa je napisano uz pomoć *Argparse* biblioteke Pythona. *Argparse* modul je odabran jer dolazi s već pripremljenim opcijama za korištenje programa. Jedna takva opcija je *-h* ili *--help* koja pokazuje sve mogućnosti sučelja.

Pri definiranju varijabli koje će se unositi pri izvršavanju programa moguće je definirati njihove kratke i duge verzije sa *-slovo* i *--naziv* za pozivanje funkcije i njihov opis koji će se pokazati pozivom opcije *-h* ili *--help* (Slika 20).



Slika 20. „Help*”* opcija korisničkog sučelja sa prikazanim opcijama programa.

Prema tome, korisničko sučelje ovog programa ima nekoliko opcija koje se mogu pozvati. Osim funkcija, zadana je i varijabla *sequences* koja prima sekvence proteina bilo da su one zadane kao specifični pristupni broj iz UniProta ili prave sekvence od slijeda aminokiselina koje su prikazane jednim slovom. *Sequences* prima jednu ili više sekvenci. Sučelje ima pet mogućih opcija uz *--help*. Definirani su *-p* ili *--path* koji daju mogućnost korisniku da sam odabere svoj UniProt dokument koji želi pretraživati. To može učiniti upisivanjem naredbe za *path* i upisivanjem lokacije dokumenta. Zatim postoji opcija *--debug* koja nije namijenjena za običnog korisnika,a koristi se za praćenje rada sučelja kada je nastao problem u programu koji se želi otkloniti. Postoji i opcija *-q* ili *--quiet* (Slika 22) koja se može koristiti kada korisnik u konzoli želi kompaktni ispis. Kompaktni ispis u sebi sadržava samo informaciju o imenu proteina i sekvenci, masi i broju izoformi, triptičkim rezovima te glikozilacijskim mjestima. Osim te opcije, postoje još dvije *-o* odnosno *--output* i *-j* odnosno *--json*. *Output* ili ispis opcija omogućuje korisniku da ispiše u čovjeku čitljivom *.txt* obliku, a *json* u računalu čitljivom obliku *.json* (iako je i on čitljiv ljudima) (Slika 21).

```
Name: Fibronectin
Mass: 262630.05697 Da
Sequence:
        MLRGPGPGLLLLAVQCLGTAVPSTGASKSKRQAQQMVQPQSPVAVSQSKPGCYDNGKHYQINQQWERTYLGNALVCTCYG
        GSRGFNCESKPEAEETCFDKYTGNTYRVGDTYERPKDSMIWDCTCIGAGRGRISCTIANRCHEGGQSYKIGDTWRRPHET
        GGYMLECVCLGNGKGEWTCKPIAEKCFDHAAGTSYVVGETWEKPYQGWMMVDCTCLGEGSGRITCTSRNRCNDQDTRTSY
        RIGDTWSKKDNRGNLLQCICTGNGRGEWKCERHTSVQTTSSGSGPFTDVRAAVYQPQPHPQPPPYGHCVTDSGVVYSVGM
        QWLKTQGNKQMLCTCLGNGVSCQETAVTQTYGGNSNGEPCVLPFTYNGRTFYSCTTEGRQDGHLWCSTTSNYEQDQKYSF
        CTDHTVLVQTRGGNSNGALCHFPFLYNNHNYTDCTSEGRRDNMKWCGTTQNYDADQKFGFCPMAAHEEICTTNEGVMYRI
        GDQWDKQHDMGHMMRCTCVGNGRGEWTCIAYSQLRDQCIVDDITYNVNDTFHKRHEEGHMLNCTCFGQGRGRWKCDPVDQ
        CQDSETGTFYQIGDSWEKYVHGVRYQCYCYGRGIGEWHCQPLQTYPSSSGPVEVFITETPSQPNSHPIQWNAPQPSHISK
        YILRWRPKNSVGRWKEATIPGHLNSYTIKGLKPGVVYEGQLISIQQYGHQEVTRFDFTTTSTSTPVTSNTVTGETTPFSP
        LVATSESVTEITASSFVVSWVSASDTVSGFRVEYELSEEGDEPQYLDLPSTATSVNIPDLLPGRKYIVNVYQISEDGEQS
        LILSTSQTTAPDAPPDTTVDQVDDTSIVVRWSRPQAPITGYRIVYSPSVEGSSTELNLPETANSVTLSDLQPGVQYNITI
        YAVEENQESTPVVIQQETTGTPRSDTVPSPRDLQFVEVTDVKVTIMWTPPESAVTGYRVDVIPVNLPGEHGQRLPISRNT
        FAEVTGLSPGVTYYFKVFAVSHGRESKPLTAQQTTKLDAPTNLQFVNETDSTVLVRWTPPRAQITGYRLTVGLTRRGQPR
        QYNVGPSVSKYPLRNLQPASEYTVSLVAIKGNQESPKATGVFTTLQPGSSIPPYNTEVTETTIVITWTPAPRIGFKLGVR
        PSQGGEAPREVTSDSGSIVVSGLTPGVEYVYTIQVLRDGQERDAPIVNKVVTPLSPPTNLHLEANPDTGVLTVSWERSTT
        PDITGYRITTTPTNGQQGNSLEEVVHADQSSCTFDNLSPGLEYNVSVYTVKDDKESVPISDTIIPAVPPPTDLRFTNIGP
        DTMRVTWAPPPSIDLTNFLVRYSPVKNEEDVAELSISPSDNAVVLTNLLPGTEYVVSVSSVYEQHESTPLRGRQKTGLDS
        PTGIDFSDITANSFTVHWIAPRATITGYRIRHHPEHFSGRPREDRVPHSRNSITLTNLTPGTEYVVSIVALNGREESPLL
        IGQQSTVSDVPRDLEVVAATPTSLLISWDAPAVTVRYYRITYGETGGNSPVQEFTVPGSKSTATISGLKPGVDYTITVYA
        VTGRGDSPASSKPISINYRTEIDKPSQMQVTDVQDNSISVKWLPSSSPVTGYRVTTTPKNGPGPTKTKTAGPDQTEMTIE
        GLQPTVEYVVSVYAQNPSGESQPLVQTAVTNIDRPKGLAFTDVDVDSIKIAWESPQGQVSRYRVTYSSPEDGIHELFPAP
        DGEEDTAELQGLRPGSEYTVSVVALHDDMESQPLIGTQSTAIPAPTDLKFTQVTPTSLSAQWTPPNVQLTGYRVRVTPKE
        KTGPMKEINLAPDSSSVVVSGLMVATKYEVSVYALKDTLTSRPAQGVVTTLENVSPPRRARVTDATETTITISWRTKTET
        ITGFQVDAVPANGQTPIQRTIKPDVRSYTITGLQPGTDYKIYLYTLNDNARSSPVVIDASTAIDAPSNLRFLATTPNSLL
        VSWQPPRARITGYIIKYEKPGSPPREVVPRPRPGVTEATITGLEPGTEYTIYVIALKNNQKSEPLIGRKKTDELPQLVTL
        PHPNLHGPEILDVPSTVQKTPFVTHPGYDTGNGIQLPGTSGQQPSVGQQMIFEEHGFRRTTPPTTATPIRHRPRPYPPNV
        GEEIQIGHIPREDVDYHLYPHGPGLNPNASTGQEALSQTTISWAPFQDTSEYIISCHPVGTDEEPLQFRVPGTSTSATLT
        GLTRGATYNVIVEALKDQQRHKVREEVVTVGNSVNEGLNQPTDDSCFDPYTVSHYAVGDEWERMSESGFKLLCQCLGFGS
        GHFRCDSSRWCHDNGVNYKIGEKWDRQGENGQMMSCTCLGNGKGEFKCDPHEATCYDDGKTYHVGEQWQKEYLGAICSCT
        CFGGQRGWRCDNCRRPGGEPSPEGTTGQSYNQYSQRYHQRTNTNVNCPIECFMPLDVQADREDSRE
Number of tryptic products: 169
Number of glycosylation sites: 10
Number of isoforms: 16
```

Slika 21. Utišani prikaz konzole

```
def Main():
        # reading config
        with open("config.json", "r") as read:
                cfg = json.load(read)

        # setting up argparser
        parser = argparse.ArgumentParser()
        parser.add_argument("-p", "--path", help="Path to XML file containing protein data you want to mine. Acce
ssion "

                                                 "numbers must be separated.", type=str)
        parser.add_argument("sequences", help="As input write the string of your sequence or your accession numbe
r",

                                                 type=str, nargs='+')
        parser.add_argument("--debug", help="Show debug messages to stdout", action="store_true")
        parser.add_argument("-q", "--quiet", help = "Show compact output", action="store_true")
        group = parser.add_mutually_exclusive_group()
        group.add_argument("-o", "--output", help="Output your search or result as a human readable file", action
="store_true")
        group.add_argument("-j", "--json", help="Output your query results to json file(computer readable file)",

                                             action="store_true")
```

Slika 22. Definiranje korisničkog sučelja i pomoći pri otklanjanju grešaka

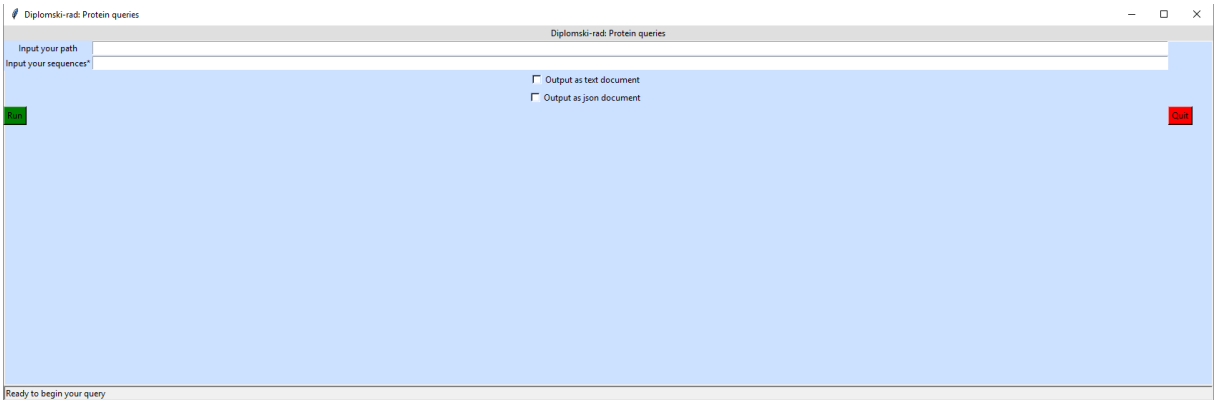
Kada korisnik unese sekvencu, program će provjeriti o kakvoj se sekvenci radi, da li je zadan put do lokacije dokumenta koji se želi koristiti (*-p*) i jesu li dodane dodatne opcije poput *-o* ili *-j*. Zadnja stvar koja se provjerava je opcija *-q* da program odluči kakav će prikaz dati korisniku (skraćeni u slučaju odabira opcije u odnosu na izostanak naredbe *-q*).

Najprije provjerava put, a ukoliko nije zadan, zadati će standardni koji je definiran unutar programa s odgovarajućom bazom. Potom provjerava je li unesena sekvenca pristupni broj ili aminokiselinski slijed. Ukoliko je u pitanju pristupni broj, koristi se estetska pomoćna klasa *ProteinOutput* koja definira kako će program prikazivati podatke. Slična opcija postoji za unos aminokiselinskog slijeda, no zbog znatno manje količine informacija postupak prikaza je drukčiji. Pomoćna funkcija *sequence_helper* pretvara unesen aminokiselinski slijed u oblik koji će odgovarati estetskoj klasi za uređeni prikaz u konzoli (Slika 23). U svakom slučaju, obje funkcije definiraju prikazivanje imena sekvence, sekvence, mase, glikozilacijskih mjesta, tripsinskih rezova i izoformi. Sve sekvence su prikazane na način da duljina jednog reda bude točno 80 znakova i da izgled odgovara standardnom FASTA formatu (tekstualnom obliku spremanja podataka o sekvencama peptida/proteina i nukleotidnih sekvenci). Prije konačnog ispisa, program provjerava da li je korisnik zatražio ispis dobivenih podataka u nekom od dva ponuđena formata te ispisuje te dokumente u direktoriju u kojem se korisnik nalazi na konzoli. Ponuđeni formati ispisa su *.txt* format predviđen za ljudsko čitanje pronađenih podataka te *.json* za računalno čitanje pronađenih podataka.

```
if args.path:
        if args.path.endswith(".xml"):
                path = os.path.normpath(args.path)
        else:
                raise EnvironmentError(f"Initializing parsing failed with file: {args.path}")
else:
        path = None

for seq in args.sequences:
        t1 = datetime.datetime.timestamp(datetime.datetime.now())
        arg_sequence = seq.upper()  # arg_sequences upper cased for proper matching
        logger.debug(f"Working on sequence: {arg_sequence}")
        if re.match('[OPQ][0-9][A-Z0-9]{3}[0-9]|[A-NR-Z][0-9]([A-Z][A-Z0-9]{2}[0-9]){1,2}', arg_sequence):
                protein = protein(xmlfile=path, acc=arg_sequence)
                logger.debug(f"Looking for your entry in input file: {args.path}")
                logger.debug(f"MATCH FOUND FOR SEQUENCE: {arg_sequence} -- PROCESSING --")
                for protein_number in protein.isoforms.keys():
                        isoforms[protein_number] = protein.isoforms[protein_number].sequence
                parsed_protein = ProteinOutput(
                                                name=protein.name,
                                                sequence=protein.sequence,
                                                mass=protein.mass,
                                                glycosylationSites=protein.glycosylation,
                                                trypticDigestion=protein.trypsine_cutter,
                                                isoforms=isoforms
                                                )
                if args.json:
                        output[parsed_protein.name] = parsed_protein.json_output()
                else:
                        output[parsed_protein.name] = parsed_protein

        elif re.match("[ACDEFGHIKLMNPQRSTVWY]+$", arg_sequence):
                logger.debug("WORKING ON CUSTOM IMPORT SEQUENCE")
                protein = form_protein(xmlfile=path, sequence=arg_sequence)
                parsed_protein = sequence_helper(protein, sequence_counter)
                if args.json:
                        output[parsed_protein.name] = parsed_protein.json_output()
                else:
                        output[parsed_protein.name] = parsed_protein
        else:
                raise ValueError(
                        'You have entered a non-existent sequence, please review your sequence
                         before trying to process it again.')
                t2 = datetime.datetime.timestamp(datetime.datetime.now())
                logger.debug(f"Processing took: {t2-t1}s")
                sequence_counter += 1

        if args.output:
                logger.debug("Outputting data as text")
                with open(cfg["text_output"], 'w') as out:
                        with redirect_stdout(out):
                                for protein in output.values():
                                        protein.formatted_output()
        elif args.json:
                logger.debug("Outputting data as json")
                with open(cfg["json_output"], 'w') as outfile:
                        json.dump(output, outfile)
        else:
                if args.quiet:
                        for protein in output.values():
                                print(protein)
                else:
                        for protein in output.values():
                                protein.formatted_output()
```

Slika 23. Izgled koda konzolnog sučelja

Osim konzolnog korisničkog sučelja, korisniku je dostupno i grafičko sučelje koje je dizajnirano korištenjem tkinter modula i dizajnirano je da bude jednostavno i lagano za korištenje. Nakon pokretanja grafičkog sučelja pojavljuje se prozor koji u sebi sadržava naslovnu i statusnu traku, tipke za početak i kraj rada programa, dvije trake za unos podataka (prva je neobvezna i služi za unos dokumenta koji se treba pretraživati, a ukoliko nije zadan odabran će biti standardni, druga obavezna u koju se unose sekvence i pristupni brojevi proteina od interesa) te dvije neobvezne kućice za dobivanje dodatnih ispisa u *.json* i *.txt* formatima (Slika 24.). Pri korištenju korisnik mora unijeti sekvence od interesa te označiti niti jedan, jedan ili oba formata (ukoliko nije odabrana niti jedna kućica podaci pretraživanja će biti prikazani samo u tekstualnom prozoru grafičkog sučelja). Kada je korisnik gotov s pretraživanjem, treba stisnuti crvenu tipku *quit* za prekid rada sučelja.



Slika 24. Izgled grafičkog sučelja

3.6. TESTIRANJE PROGRAMA

Točnost programa koji se izvodi provjerava se putem Pytest modula. Pomoću te biblioteke i poznatih očekivanih rezultata moguće je uvjeriti se u ispravnost rada programa. Za potrebe ovog diplomskog rada testirane su sljedeće opcije programa: točnost *name* svojstva, pretraživanja kanonske sekvence, glikozilacijskih mjesta, postavljanja rječnika potrebnih za konstrukciju izoforme, konstruirane izoforme koja ima točno jednu promjenu, konstruirane izoforme koja ima više od jedne promjene, glikozilacije izoforme, točnost izračunate mase proteina i prvog tripsinskog reza sekvence. Za provjeru točnosti korišteni su podaci dobiveni iz Unpirot baze podataka. Ispitivanje točnosti je provedeno na uparenom imunoglobulinu sličnom tip 2 receptoru beta (engl. *Paired imunoglobulin-like type 2 receptor beta*, Q9UKJO). Taj protein je odabran zbog odgovarajućih kriterija. Dovoljno je malen da bi

se lagano prikazao na konzoli te ima barem dvije izoforme od kojih jedna ima točno jednu promjenu, a druga više (odabrani protein ima točno dvije promjene u odnosu na kanonsku strukturu). Posljednji kriterij po kojem je odabran kao testni protein je činjenica da je glikoziliran (N-glikozilacija na stotoj aminokiselini). Nakon odabira proteina bilo je nužno pronaći sve navedene podatke i postaviti ih za referencu poput konstante (takvo postavljanje u pytest modulu definirano dekoratorom *@pytest.fixtures*) (Slika 25). Nakon toga su postavljeni testovi koji su formirani na način da se baza podataka pretražuje za određeni protein i nakon toga dobiveni rezultat uspoređuje s prethodno poznatim podatkom. To se izvršava pomoću naredbe *assert* i znaka dvostruke jednakosti == (Slika 26).

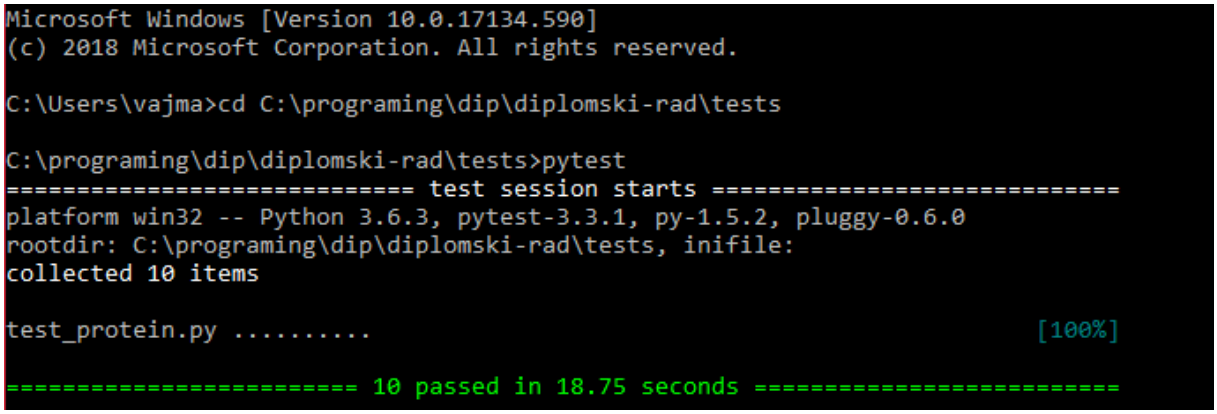
```
import pytest

from protein import protein

@pytest.fixture(scope="module")
def entry ():
        entry = protein(xmlfile="gli.xml", acc="Q9UKJ0")
        return entry


@pytest.fixture(scope="module")
def Q9UKJ0_1():
        Q9UKJ0_1_sequence = """
                        MGRPLLLPLLLLLQPPAFLQPGGSTGSGPSYLYGVTQPKHLSASMGGSVEIPFSFYYPWE
                        LAIVPNVRISWRRGHFHGQSFYSTRPPSIHKDYVNRLFLNWTEGQESGFLRISNLRKEDQ
                        SVYFCRVELDTRRSGRQQLQSIKGTKLTITQAVTTTTTWRPSSTTTIAGLRVTESKGHSE
                        SWHLSLDTAIRVALAVAVLKTVILGLLCLLLLWWRRRKGSRAPSSDF
                        """
        Q9UKJ0_1_sequence = ''.join(Q9UKJ0_1_sequence.split())
        return Q9UKJ0_1_sequence


@pytest.fixture(scope="module")
def Q9UKJ0_2():
        Q9UKJ0_2_sequence = """
                        MGRPLLLPLLLLLQPPAFLQPGLCEPALSELDRGSGERLPQDLKPAEGGPVCVFLPSRAG
                        HPEIREAAVAVHQGDQTHHHPGCHNHHHLEAQQHNHHSRPQGHRKQRALRIMAPKSGHCH
                        QGCIGCRCAQNCHFGTAVPPPPVVEEKER
                        """
        Q9UKJ0_2_sequence = ''.join(Q9UKJ0_2_sequence.strip().split())
        return Q9UKJ0_2_sequence


@pytest.fixture(scope="module")
def Q9UKJ0_3():
        Q9UKJ0_3_sequence = """
                        MGRASYGGSPVGHQSSSDHPRAKTCRSPVRDGPRTLCGVPTVALSLHFLREASSGSRTCG
                        RRTSLCTSAKSSWTYRSGRLSWQSIKGTHLTITQALRQPLHRAPLLPGQLCWSPRPLEKN
                        KAMGRPLLLPLLLLLQPPAFLQPGLCEPALSELDRGSGERLPQDLKPAEGGPVCVFLPSR
                        AGHPEIREAAVAVHQGDQTHHHPGCHNHHHLEAQQHNHHSRPQGHRKQRALRIMAPKSGH
                        CHQGCIGCRCAQNCHFGTAVPPPPVVEEKER
                        """
        Q9UKJ0_3_sequence = ''.join(Q9UKJ0_3_sequence.strip().split())
        return Q9UKJ0_3_sequence
```

Slika 25. Fiksiranje poznatih vrijednosti za izvođenje testova

```
@pytest.fixture(scope="module")
def trypsine_cut_1():
        trypsine_cut_1 = ['MGRPLLLPLLLLLQPPAFLQPGGSTGSGPSYLYGVTQPK', 1, 39]
        return trypsine_cut_1

def test_protein_name(entry):
        name = entry.name
        assert name == "Paired immunoglobulin-like type 2 receptor beta"


def test_canonical_sequence(Q9UKJ0_1,entry):
        sequence = entry.sequence
        assert sequence == Q9UKJ0_1


def test_glycosylation_sites(entry):
        glycosylation = entry.glycosylation
        assert glycosylation == {100: 'N-linked (GlcNAc...) asparagine'}



def test_alternative_sequence_isoform_changes_ids(entry):
        ids_data = entry.isoforms[3]._alternative_products()
        assert ids_data == {2: ['VSP_017504'], 3: ['VSP_017503', 'VSP_017504']}


def test_isoform_data_changes_information(entry):
        isoform_data = entry.isoforms[3]._vsps()
        assert isoform_data == {
                'VSP_017503': ('MGRASYGGSPVGHQSSSDHPRAKTCRSPVRDGPRTLCGVPTVALSLHFLREASSGSRTCGRRTSLCTSAKSSWTYRS
GRLSWQSIKGTHLTITQALRQPLHRAPLLPGQLCWSPRPLEKNKAM', 1, 1),
                'VSP_017504': ('LCEPALSELDRGSGERLPQDLKPAEGGPVCVFLPSRAGHPEIREAAVAVHQGDQTHHHPGCHNHHHLEAQQHNHHSR
PQGHRKQRALRIMAPKSGHCHQGCIGCRCAQNCHFGTAVPPPPVVEEKER', 23, 227)
        }

def test_single_vsp(Q9UKJ0_2,entry):
        assert entry.isoforms[2].sequence == Q9UKJ0_2

def test_multiple_vsp(Q9UKJ0_3,entry):
        assert entry.isoforms[3].sequence == Q9UKJ0_3

def test_isoform_glycosylation(entry):
        assert entry.isoforms[3].glycosylation == {}

def test_mass(entry):
        assert entry.mass == 25542.989100000003


def test_trypsine_cutter(trypsine_cut_1,entry):
        assert entry.tryptic_peptides[0] == trypsine_cut_1
```

Slika 26. Postavljanje testova koji će biti izvedeni (testovi za provjeru metoda koje određuju ime, kanonsku sekvencu, glikozilacijska mjesta kanonske sekvence, poznate izoforme sa VSP ključevima, VSP ključevi sa svojim podacima, glikozilacijska mjesta izoforme, masu, tripsinske rezove)

Tako napisane naredbe uspoređuju vrijednosti i dodjeljuju testu prolaznu ili neprolaznu ocjenu. Pri izvšavanju testova pomoću *windows* konzole potrebno je locirati dokument s testovima putem naredbe *change directory ili cd* („promijeni direktorij“, u slučaju računala na kojem je izrađen ovaj diplomski rad to bi glasilo ovako: *cd C:\programing\dip\diplomski-rad\tests*) i upisati naredbu *pytest*. Nakon toga se testovi izvršavaju redom i konzola vraća broj testova koji se provode, njihovu prolaznost (dodjeljuje im ocjenu prolaz/pad ovisno o tome odgovara li dobiveni rezultat programa očekivanoj vrijednosti) i vrijeme potrebno za njihovo izvođenje (Slika 27).



Slika 27. Primjer 10 rezultata izvedenih testova za provjeru programa.

3.7. ANALIZA GLIKOPROTEINA HUMANE PLAZME

Na ovaj način dizajniran program je korišten za pretraživanje proteina humane plazme. Kao i u prethodnom pretraživanju UniProta, određen je humani proteom koji je filtriran za glikoproteine (annotation:(type:carbohyd) tissue:plasma AND reviewed:yes AND organism:"Homo sapiens (Human) [9606]"). Pronađeni proteini su skinuti u obliku liste i XML dokumenta koji su pretraženi. Zatim je napisan kod koji učitava listu proteina koje treba pretraživati i XML dokument sa zadanim proteinima (Slika 28).

```
plasma_proteins = []
input = open("plasma.list", "r")

input_content = input.read()
input_list = input_content.split()

for plasma_protein in input_list:
        protein_instance = protein(acc=plasma_protein,
xmlfile="C:\programing\dip\diplomski-rad\human_plasma_analysis\plasma.xml")
        plasma_proteins.append(protein_instance)```
```

Slika 28. Učitavanje i postavljanje koda za analizu

Kako bi se dobile vrijednosti koje želimo analizirati, napisane su 3 funkcije za svaki od naših podataka od interesa koje pretražuju sve instance proteina i izvlače podatke o duljinama, masama i hidrofobnostima glikoproteina te vraćaju liste tih podataka (Slika 29).

```
def length_set(instance_list):
        sequences = []

        for plasma_protein in instance_list:
                sequences.append(len(plasma_protein.sequence))

        return sequences


def avg_hydrophobicity_set(instance_list):
        hydrophobicities = []

        for plasma_protein in instance_list:
                hydrophobicities.append(plasma_protein.avg_hydrophobicity)

        return hydrophobicities


def mass_set(instance_list):
        masses = []

        for plasma_protein in instance_list:
                masses.append(plasma_protein.mass)

        return masses

def sequence_set(instance_list):
        sequences = []

        for plasma_protein in instance_list:
                sequences.append(plasma_protein.sequence)

        print (sequences)
```

Slika 29. Stvaranje listi za crtanje grafova

Napisani su algoritmi za crtanje histograma korištenjem matplotlib biblioteke. Histogramima je dan naslov, imena osi, set proteina i prilagođeni koševi za strukturu podataka za bolji prikaz (Slika 30).

```
plt.hist(length_set(plasma_proteins), l_bins, rwidth=0.9)
plt.ylabel("Number of proteins")
plt.xlabel("Aminoacid length")
plt.title("Protein length distribution")
plt.show()

plt.hist(mass_set(plasma_proteins), m_bins, rwidth=0.9)
plt.ylabel("Number of proteins")
plt.xlabel("Protein mass")
plt.title("Protein mass distribution")
plt.show()

plt.hist(avg_hydrophobicity_set(plasma_proteins),h_bins, rwidth=0.9
plt.ylabel("Number of proteins")
plt.xlabel("Average protein hydrophobicity")
plt.title("Protein hydrophobicity distribution")
plt.show()
```

Slika 30. Crtanje histograma duljine, mase hidrofobnosti.

Osim prikazanih distribucija, izračunata je i distribucija vrsta glikozilacija zabilježenih UniProt bazom podataka. Program računa svaku od pet vrsta glikozilacija koje ta baza bilježi (N-, O-, S-, C-glikozilacija i glikacija). Nakon što metoda skupi podatke od interesa vraća ih kao listu vrijednosti za pojedinu glikozilaciju (Slika 31).

```
def glycosylation_analysis(list):
        N = 0
        O = 0
        C = 0
        S = 0
        G = 0

        for protein in list:
                for glycosylation in protein.glycosylation.values():
                        if 'N-linked' in glycosylation:
                                N += 1
                        elif 'O-linked' in glycosylation:
                                O += 1
                        elif 'C-linked' in glycosylation:
                                C += 1
                        elif 'S-linked' in glycosylation:
                                S += 1
                        elif 'Glycation' in glycosylation:
                                G += 1

        glycosylations = [N, O, C, S, G]
        return glycosylations
```

Slika 31. Izračunavanje tipova glikozilacije glikoproteina humane plazme.

Posljednji napisani kod programa je kod za analizu peptida dobivenih tripsinskom digestijom proteina. Budući da se proteini spremaju kao instance, imaju sve mogućnosti obrade kao i neanotirani proteini. Taj aspekt programa je ovdje iskorišten te je napravljen kod za distribucije duljina, masa i hidrofobnosti. Prvo su napravljeni setovi podataka za svaku vrstu informacija od interesa. Peptidi su raspoređeni prema vrsti glikozilacije odnosno ukoliko nisu glikozilirani (Slika 32 i 33).

```
def peptide_set_len(instance_list):
        peptides = []
        N_glyco_peptides = []
        O_glyco_peptides = []
        C_glyco_peptides = []

        for protein in instance_list:
                for glycosylation in protein.glycosylation.keys():
                        peptides.append(glycosylation)
                        for peptide in protein.tryptic_peptides:
                                if int(glycosylation) > peptide.start-1 and glycosylation < peptide.end:
                                        if 'N-linked' in protein.glycosylation[glycosylation]:
                                                N_glyco_peptides.append(len(peptide.sequence))
                                        elif 'O-linked' in protein.glycosylation[glycosylation]:
                                                O_glyco_peptides.append(len(peptide.sequence))
                                        elif 'C-linked'in protein.glycosylation[glycosylation]:
                                                C_glyco_peptides.append(len(peptide.sequence))
                                else:
                                        peptides.append(len(peptide.sequence))

        list_peptides = [N_glyco_peptides, O_glyco_peptides, C_glyco_peptides, peptides]
        return list_peptides

def peptide_set_hydro(instance_list):
        peptides = []
        N_glyco_peptides = []
        O_glyco_peptides = []
        C_glyco_peptides = []

        for protein in instance_list:
                for glycosylation in protein.glycosylation.keys():
                        peptides.append(glycosylation)
                        for peptide in protein.tryptic_peptides:
                                if int(glycosylation) > peptide.start-1 and glycosylation < peptide.end:
                                        if 'N-linked' in protein.glycosylation[glycosylation]:
                                                N_glyco_peptides.append(peptide.avg_hydrophobicity)
                                        elif 'O-linked' in protein.glycosylation[glycosylation]:
                                                O_glyco_peptides.append(peptide.avg_hydrophobicity)
                                        elif 'C-linked'in protein.glycosylation[glycosylation]:
                                                C_glyco_peptides.append(peptide.avg_hydrophobicity)
                                else:
                                        peptides.append(peptide.avg_hydrophobicity)

        list_peptides = [N_glyco_peptides, O_glyco_peptides, C_glyco_peptides, peptides]
        return list_peptides
```

Slika 32. Računanje listi za crtanje violinskih grafova

```
def peptide_set_mass(instance_list):
        peptides = []
        N_glyco_peptides = []
        O_glyco_peptides = []
        C_glyco_peptides = []

        for protein in instance_list:
                for glycosylation in protein.glycosylation.keys():
                        peptides.append(glycosylation)
                        for peptide in protein.tryptic_peptides:
                                if int(glycosylation) > peptide.start-1 and glycosylation < peptide.end:
                                        if 'N-linked' in protein.glycosylation[glycosylation]:
                                                N_glyco_peptides.append(peptide.mass)
                                        elif 'O-linked' in protein.glycosylation[glycosylation]:
                                                O_glyco_peptides.append(peptide.mass)
                                        elif 'C-linked'in protein.glycosylation[glycosylation]:
                                                C_glyco_peptides.append(peptide.mass)
                                else:
                                        peptides.append(peptide.mass)

        list_peptides = [N_glyco_peptides, O_glyco_peptides, C_glyco_peptides, peptides]
        return list_peptides
```

Slika 33. Računanje listi za crtanje violinskih grafova (nastavak koda)

U konačnici su napravljeni violinski plotovi distribucije peptida prema duljini (Slika 34), masi (Slika 35) i hidrofobnosti (Slika 36). Napravljeni su pojedinačno te jedan do drugog za direktnu usporedbu. Crtani su pyplot bibliotekom s isključenim ekstremnim vrijednostima. Dani su im naslovi i označene su osi.

```
data_dict_len = {"N-glikozilirani":peptide_set_len(plasma_proteins)[0], "O-glikozili
rani":peptide_set_len(plasma_proteins)[1],
                                "C-glikozilirani":peptide_set_len(plasma_protein
s)[2], "Neglikozilirani":peptide_set_len(plasma_proteins)[3]}

N, O, C, un = data_dict_len.keys()

categories = (N, O, C, un)

plt.figure()
for i, cat in enumerate(categories):
        ax_ = plt.gca()
        ax_.violinplot(data_dict_len[cat], showextrema=False, showmedians=True, poin
ts=4000, positions=[i+1])
        ax_.set

ax_.set_ylim(0, 100)
plt.title("Violinski graf distribucije duljina svih analiziranih tipova glikozilacij
e peptida")
plt.ylabel("Duljina peptida / aa")
plt.xticks(range(1, 5), categories)
plt.savefig('lengths.png')
```

Slika 34. Kod za crtanje violinskog grafa distribucije duljina analiziranih tipova glikozilacije peptida

```
data_dict_mass = {"N-glikozilirani":peptide_set_mass(plasma_proteins)[0], "O-glikozilirani":peptide_set_mass
(plasma_proteins)[1],
                                "C-glikozilirani":peptide_set_mass(plasma_proteins)[2], "Neglikozilirani":pep
tide_set_mass(plasma_proteins)[3]}

N, O, C, un = data_dict_mass.keys()

categories = (N, O, C, un)

plt.figure()

for i, cat in enumerate(categories):
        ax_ = plt.gca()
        ax_.violinplot(data_dict_mass[cat], showextrema=False, showmedians=True, points=4000, positions=[i+1
])
        ax_.set
ax_.set_ylim(0, 15000)
plt.title("Violinski graf distribucije masa svih analiziranih tipova glikozilacije peptida")
plt.ylabel("Masa peptida / Da")
plt.xticks(range(1, 5), categories)
plt.savefig('mass.png')
```

Slika 35. Kod za crtanje violinskog grafa distribucije masa analiziranih tipova glikozilacije peptida

```
data_dict_hydro = {"N-glikozilirani":peptide_set_hydro(plasma_proteins)[0], "O-glikozilirani":peptide_set_hyd
ro(plasma_proteins)[1],
                                        "C-glikozilirani":peptide_set_hydro(plasma_proteins)[2], "Neglikozili
rani":peptide_set_hydro(plasma_proteins)[3]}


N, O, C, un = data_dict_hydro.keys()

categories = (N, O, C, un)

plt.figure()
for i, cat in enumerate(categories):
        ax_ = plt.gca()
        ax_.violinplot(data_dict_hydro[cat], showextrema=False, showmedians=True, points=4000, positions=[i+1
])
        ax_.set
ax_.set_ylim(-500, 200)
plt.title("Violinski graf distribucije hidrofobnosti svih analiziranih tipova glikozilacije peptida")
plt.ylabel("Hidrofobnost peptida / kcal pri 25°C")
plt.xticks(range(1, 5), categories)
plt.savefig('hydro.png')
```

Slika 36. Kod za crtanje violinskog grafa distribucije hidrofobnosti analiziranih tipova glikozilacije peptida

4. REZULTATI I RASPRAVA

4.1 REZULTATI

4.1.1 KORISNIČKI SLUČAJ

Za olakšavanje rada s programom, dizajniran je poseban slučaj primjene korištenja programa. Objašnjenje rada na programu nalazi se u dokumentu *use_case.py* s danim primjerom i rezultatima u *pydoc* obliku. Objašnjenje je ponuđeno na primjeru Epifrina 4. Za potrebe slučaja primjene je definiran protein korištenjem tvornice instanci. Pri definiranju proteina zadan je testni dokument *gli.xml* i pristupni broj efrina A4 (acc="P52798"). Potom su pokazane odabrane značajke proteina koje se mogu pretraživati. Ovim slučajem za upoznavanje korisnika je opisano pretraživanje imena proteina, proteinske kanonske sekvence i glikozilacijskh mjesta proteina od interesa, mase proteina i njegovog drugog tripsinskog reza u sekvenci. Također je prikazan pristup instanci izoforme koja je dio instance proteina. Takva instanca izoforme se može pretraživati kako bi se dobile željene inofrmacije. U ponuđenom korisničkom slučaju je prikazano kako pretražiti ime i sekvencu izoforme efrin A4-2. Takav korisnički slučaj se može pokrenuti u konzoli i prikazati korisniku.

Korisnički slučajevi diplomskog rada

Učitavanje potrebnih modula:

```
>>> from protein import form_protein
```

Inicijalizacija:

```
>>> protein = protein(xmlfile="gli.xml", acc="P52798")
```

Form_protein prima tri moguća argumenta: *xmlfile* (dokument u kojem će pretraživanje proteina od interesa biti provedeno, ukoliko nije dan, bit će korišten prethodno definirani), *acc* (pristupni broj proteina koji će biti korišten za pronalaženje sekvence od interesa), *sequence* (za osnovne metode na prethodno poznatoj peptidnoj sekvenci, npr. masa, tripsinski rez, sekvenca). Ukoliko nisu zadani ni *acc* ni *sequence* korisnik će dobiti grešku.

Pristup pohranjenim informacijama u instanci proteina:

- **ime proteina:**

```
>>> protein.name
'Ephrin-A4'
```

- **kanonska sekvenca proteina:**

```
>>> protein.sequence
'MRLLPLLRTVLWAAFLGSPLRGGSSLRHVVYWNSSNPRLLRGDAVVELGLNDYLDIVCPHYEGPGPP
EGPETFALYMVDWPGYESCQAEGPRAYKRWVCSLPFGHVQFSEKIQRFTPFSLGFEFLPGETYYYISV
PTPESSGQCLRLQVSVCCKERKSESAHPVGSPGESGTSGWRGGDTPSPLCLLLLLLLLILRLLRIL'
```

- **glikozilacijska mjesta proteina:**

```
>>> protein.glycosylation
{33: 'N-linked (GlcNAc...) asparagine'}
```

- **proteinske izoforme (pohranjuju se kao rječnik u kojem je broj izoforme ključ, a instanca sekvence vrijednost):**

```
>>> protein.isoforms[2].sequence
'MRLLPLLRTVLWAAFLGSPLRGGSSLRHVVYWNSSNPRLLRGDAVVELGLNDYLDIVCPHYEGPGPP
EGPETFALYMVDWPGYESCQAEGPRAYKRWVCSLPFGHVQFSEKIQRFTPFSLGFEFLPGETYYYISV
PTPESSGQCLRLQVSVCCKERNLPSHPKEPESSQDPLEEEGSLLPALGVPIQTDKMEH'

>>> protein.mass
22386.357689999997
```

- **vraćanje instance izoforme:**

Za povratak instance izoforme potreban je broj izoforme kao ključ *isoform* rječnika. Također može biti kreirana s funkcijom *protein* ukoliko postoji i ukoliko su zadani točan broj izoforme i pristupni broj odgovarajućeg proteina.

```
>>> Isoform_2 = protein.isoforms[2]
```

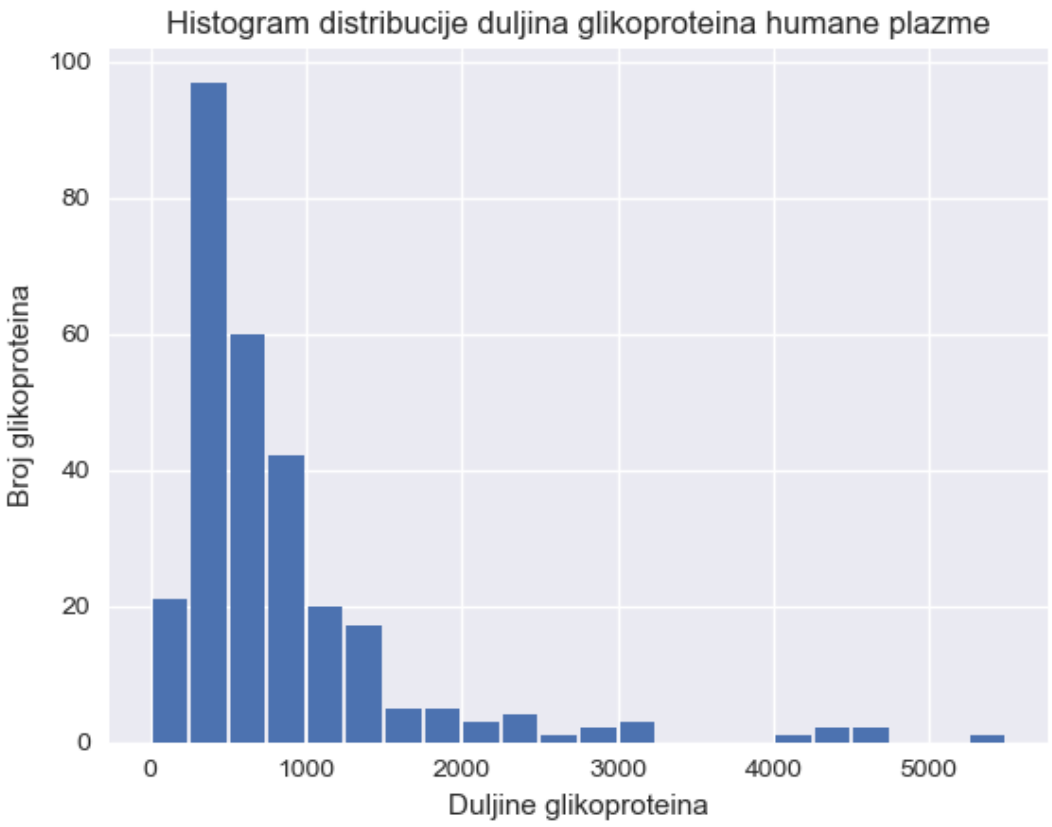
Isoform klasa ima sve opcije kao i *UniProtprotein* s iznimkom izoformi.

```
>>> Isoform_2.name
'Ephrin-A4-2'
```

4.1.2 REZULTATI ANALIZE GLIKOPROTEINA HUMANE PLAZME

Pretraživanjem 294 proteina plazme je filtrirano 286 glikoziliranih proteina. Prvi rezultati analize glikoproteina prikazuju izračunatu srednju vrijednost i standardnu devijaciju duljina, hidrofobnosti i mase glikoproteina njihovih duljina, hidrofobnosti i mase proteina, te prikazuju njihovu distribuciju histogramima. Za računanje srednje vrijednosti i standardne devijacije korištena je biblioteka statistics.

Nakon što su glikoproteini plazme obrađeni dobiveni su sljedeći rezultati. Prosječna duljina je bila 891.10 aminokiselina sa standardnom devijacijom od 783.16 aminokiseline. Na grafu distribucije se vidi velika zastupljenost glikoproteina veličine do 1000 aminokiselina i znatno manje proteina većih od te vrijednosti (Slika 37, Tablica 1).

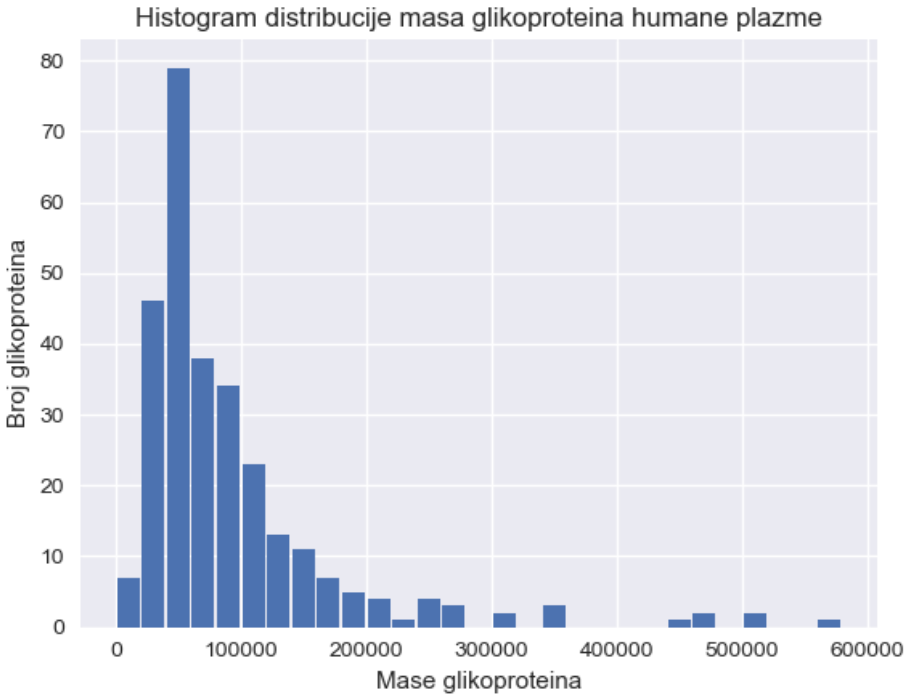


Slika 37. Distribucija glikoproteina plazme po duljinama.

Tablica 1. Tablica raspodjele aminokiselina po duljinama

Interval duljine/broj aminokislina	Broj proteina
od 0 do 250	21
od 250 do 500	97
od 500 do 750	60
od 750 do 1000	42
od 1000 do 1250	20
od 1250 do 1500	17
od 1500 do 1750	5
od 1750 do 2000	5
od 2000 do 2250	3
od 2250 do 2500	4
od 2500 do 2750	1
od 2750 do 3000	2
od 3000 do 3250	3
od 3250 do 3500	0
od 3500 do 3750	0
od 3750 do 4000	0
od 4000 do 4250	1
od 4250 do 4500	2
od 4500 do 4750	2
od 4750 do 5000	0
od 5000 do 5250	0
od 5250 do 5500	1
Ukupno	**286**

Distribucija masa je vrlo slična distribuciji duljina. Prosječna masa je 92.939 kDa sa standardnom devijacijom od 85.466.496 kDa. Većina proteina bila je mase ispod 100 kDa (Slika 38, Tablica 2).

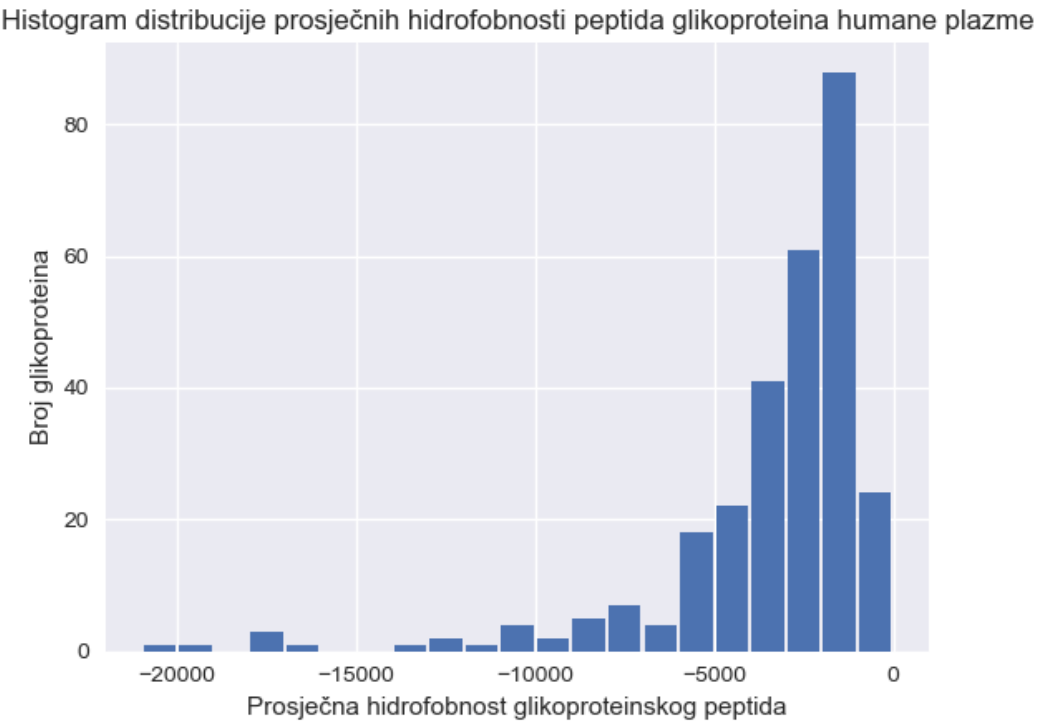


Slika 38. Distribucija glikoproteina plazme po masama

Tablica 2. Tablica raspodjele aminokiselina po masama.

Interval masa/kDa	Broj proteina
od 0 do 20	7
od 20 do 40	46
od 40 do 60	79
od 60 do 80	38
od 80 do 100	34
od 100 do 120	23
od 120 do 140	13
od 140 do 160	11
od 160 do 180	7
od 180 do 200	5
od 200 do 220	4
od 220 do 240	1
od 240 do 260	4
od 260 do 280	3
od 280 do 300	0
od 300 do 320	2
od 320 do 340	0
od 340 do 360	3
od 360 do 380	0
od 380 do 400	0
od 400 do 420	0
od 420 do 440	0
od 440 do 460	1
od 460 do 480	2
od 480 do 500	0
od 500 do 520	2
od 520 do 540	0
od 540 do 560	0
od 560 do 580	1
Ukupno	**286**

Sljedeća analiza provedena ovim radom bila je analiza distribucije prosječne hidrofobnosti humanih glikoproteina plazme. Rezultati u ovom slučaju su puno skupljeniji u odnosu na masu i duljine te pokazuju da su svi humani glikoproteini plazme negativne hidrofobnosti (hidrofilni). To ipak nije pravi pokazatelj njihove hidrofilnosti jer u nju nije uračunat i glikanski doprinos, tako da su još hidrofilniji zbog hidrofilne prirode glikana (Slika 39, Tablica 3).



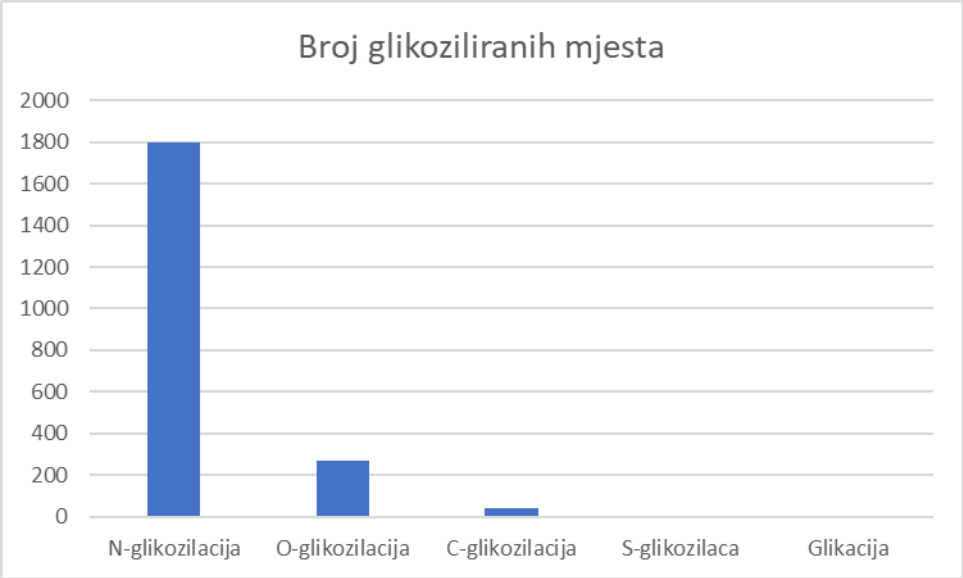
Slika 39. Distribucija glikoproteina plazme po masama

Tablica 3. Distribucija hidrofobnosti glikoproteina humane plazme

Interval hidrofobnosti/kCal 25°C	Broj proteina
od -21000 do -20000	1
od -20000 do -19000	1
od -19000 do -18000	0
od -18000 do -17000	3
od -17000 do -16000	1
od -16000 do -15000	0
od -15000 do -14000	0
od -14000 do -13000	1
od -13000 do -12000	2
od -12000 do -11000	1
od -11000 do -10000	4
od -10000 do -9000	2
od -9000 do -8000	5
od -8000 do -7000	7
od -7000 do -6000	4
od -6000 do -5000	18
od -5000 do -4000	22
od -4000 do -3000	41
od -3000 do -2000	61
od -2000 do -1000	88
od -1000 do 0	24
Ukupno	**286**

Nakon toga je uspoređen odnos tipova glikozilacija. UniProt prati pet tipova glikozilacija: N-, O-, C- i S-glikozilaciju te glikaciju. Analizom glikoproteina humane plazme dobiveno je da je daleko najzastupljeniji tip glikozilacije u humanoj plazmi N-glikozilacija. Takva situacija je očekivana.

Pregledom podataka može se primjetiti 1796 N-glikozilacijskih mjesta, 269 O-glikoziliranih, 39 C-glikoziliranih i 1 S-glikozilirani. Glikacija nije zabilježena UniProtovom bazom podataka (Slika 40, Tablica 4).



Slika 40. Distribucija tipova glikozilacije glikoproteina humane plazme.

Tablica 4. Tablica distribucija Glikozilacijskih mjesta glikoproteina humane plazme.

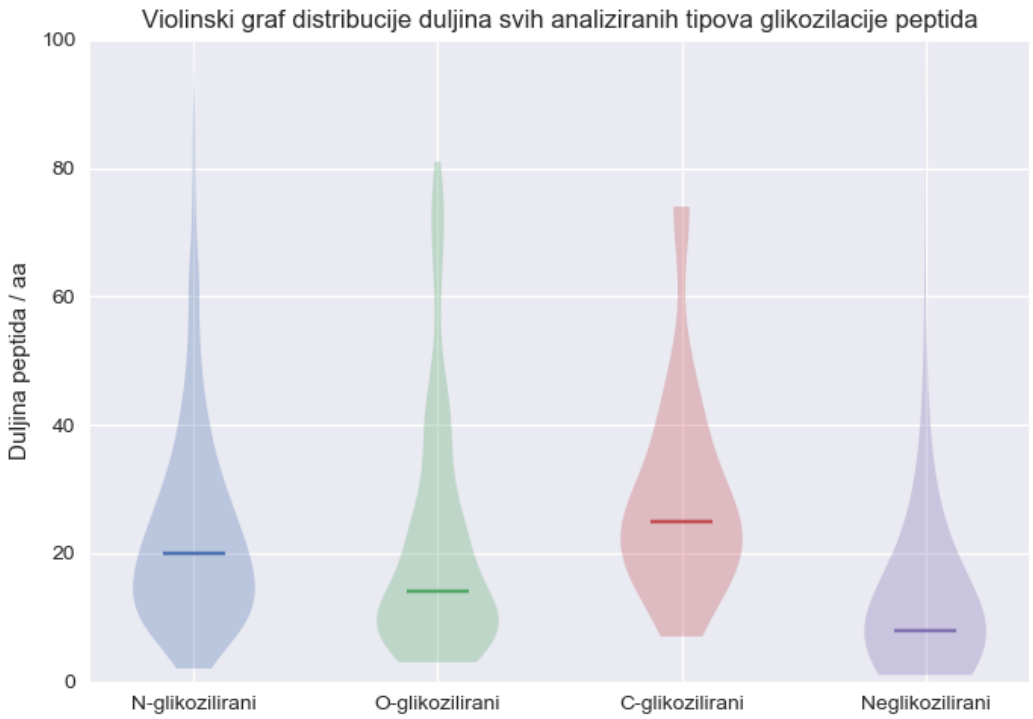
Tip Glikozilacije	Broj glikoziliranih mjesta
N-glikozilacija	1796
O-glikozilacija	269
C-glikozilacija	39
S-glikozilaca	1
Glikacija	0

4.1.3. DISTRIBUCIJA PEPTIDA GLIKOPROTEINA HUMANE PLAZME NASTALIH TRIPSINSKOM DIGESTIJOM

Pri analizi peptida glikoproteina humane plazme oni su podijeljeni u četiri skupine. Podijeljeni su prema vrsti glikozilacije. Razgradnjom peptida tripsinom nastaje 1790 proteina koji sadržavaju N-glikozilirane peptide, 261 O-glikozilirana peptida, 39 C-glikoziliranih peptida i 222 001 neglikozilirana peptida. Zbog toliko velikih razlika pri analizi peptida bilo je nužno uzeti to u obzir pri njihovoj usporednoj analizi.

Prva analiza je napravljena prema duljini peptida. Pri usporedbi svih peptida bez stršećih vrijednosti odmah se može primijetiti da su peptidi nastali razgradnjom manji od 100

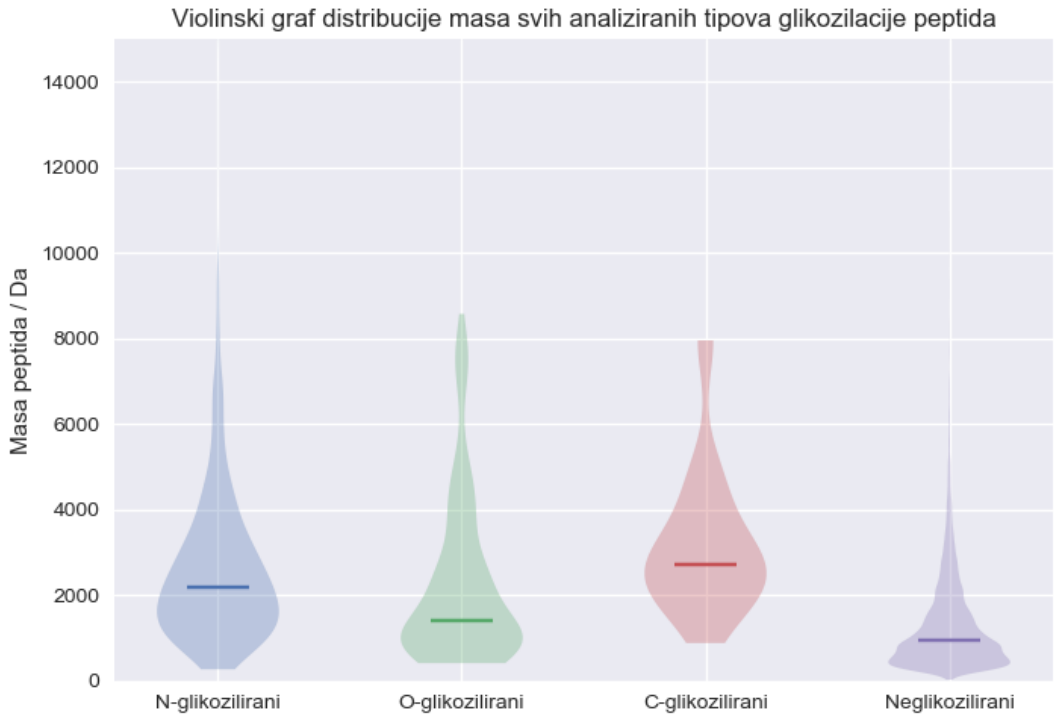
aminokiselina. Promatranjem distribucije N-glikoziliranih peptida može se zamijetiti da su duljine većine peptida ispod 20 aminokiselina, s veličinama koje dosežu 300 aminokiselina. Kod O-glikoziliranih peptida se primjećuje da su svi duljine manje od 80 aminokiselina. Može se primjetiti da je duljina većine O-glikoziliranih peptida ispod 20 (oko 10) aminokiselina dok je polovica C-glikoziliranih peptida duljine oko 20 aminokiselina sa znatno manjim brojem peptida duljih od 60 aminokiselina. Neglikozilirani peptidi su većinom mali i imaju najmanji medijan (Slika 41). Vrijednosti mogu biti i veće od 4000 aminokiselina, no takva ona je u ovom slučaju isključena kao stršeća vrijednost budući da ne predstavlja ispravno dobiveni uzorak.



Slika 41. Violinski graf distribucije duljina svih analiziranih tipova glikozilacije peptida.

Iduća usporedba je napravljena prema masi koja je očekivano pratila distribuciju prema duljini. Prvo se može pogledati distribuciju masa N-glikoziliranih peptida. Iz grafa je vidljivo da je većina peptida manja od 4 kDa, sa stršećim vrijednostima preko 30 kDa (isključene kao stršeće vrijednosti). Distribucija O-glikoziliranih peptida pokazuje da je

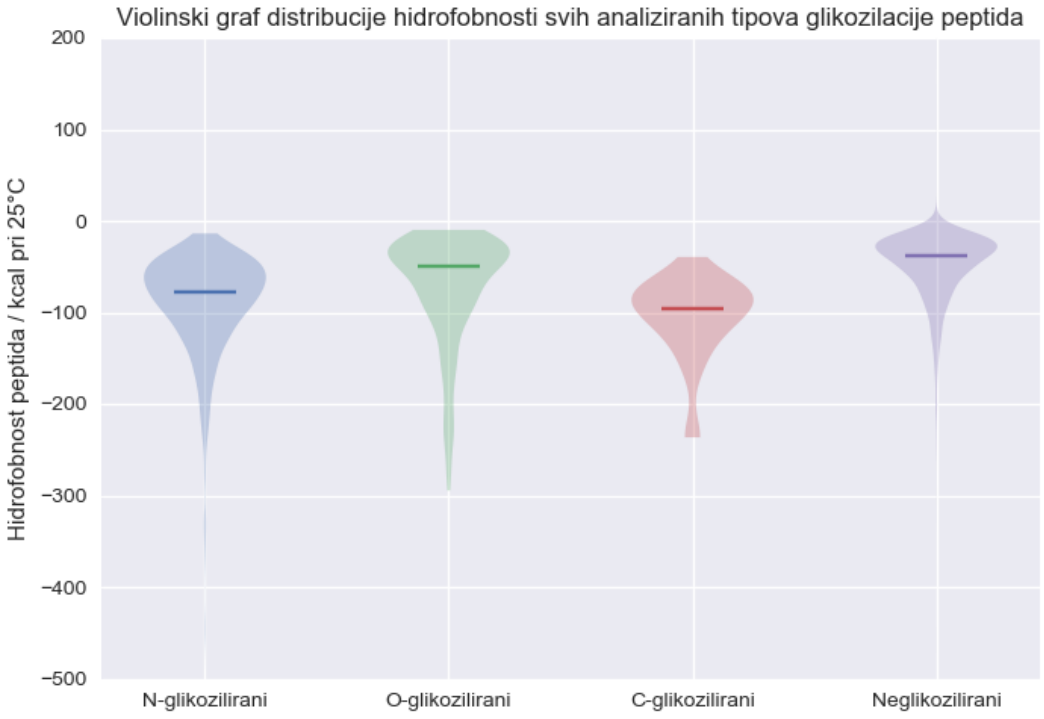
najveći broj peptida manji od 2 kDa te da distribucija pada prema vršnoj vrijednosti koja se nalazi na oko 10 kDa. C-glikozilirani peptidi nemaju velike stršeće vrijednosti kao O- i N-glikozilirani peptidi. Većina ih je veličine oko 2 do 4 kDa. Također se može primjetiti malu grupu peptida na 8 kDa. Neglikozilirani peptidi predstavljaju najveću skupinu peptida. Velikom većinom su mali ispod 10 kDa, a vrijednosti mogu narasti do 60kDa (grafom isključeno kao stršeća vrijednost). Kada uklonimo stršeće vrijednosti slika opet postaje jasnija i može se primjetiti da je samo mali broj peptida veći od 5 kDa (Slika 42).



Slika 42. Violinski graf distribucije masa svih analiziranih tipova glikozilacije peptida

Iz grafa distribucija hidrofobnosti vidljivo je da distribucija većinom naginje prema negativnim vrijednostima hidrofobnosti (hidrofilnijim peptidima), iako postoje stršeće vrijednosti neglikoziliranih hidrofobnih peptida (isključenih u grafu). N-glikozilirani peptidi su većinom lagano hidrofilni. O-glikozilirani peptidi imaju tendenciju prema negativnim vrijednostima hidrofobnosti. Vrijednosti hidrofobnosti su blizu 0 što je uzrokovano malim veličinama tih peptida. C-glikozilirani peptidi su većinom hidrofobni, a vrijednosti do oko -100 kcal pri 25°C. Pokazuju niže vrijednosti nego O-glikozilirani peptidi. Iako većina

neglikoziliranih peptida ima negativne vrijednosti hidrofobnosti, manji dio ima pozitivne vrijednosti (Slika 43).



Slika 43. Violinski graf distribucije hidrofobnosti svih analiziranih tipova glikozilacije peptida

4.2 RASPRAVA

Ovim radom je uspješno napravljen alat za pretraživanje proteinske baze podataka UniProt. Pokazana je njegova točnost i pouzdanost te je provjeren uzorak glikoproteina humane plazme dostupan na UniProtu. Posljedično je pokazano kako razvoj takvih alata može dopuniti rad na već postojećim alatima i popraviti neke njihove nedostatke.

Na ovaj način je uspješno riješen problem pojedinačnih gledanja proteina i uskog opsega pretraga proteina na ograničenom uzorku (od 400 proteina koji nudi UniProt). Programom je prikazana mogućnost pretraživanja cijelih proteoma/glikoproteoma, izvođenja statistike na njima, stvaranja grafova te pravljenja raznih izračuna na dostupnim podacima.

Iz programa je moguće vidjeti kako su razvijeni alati za izračunavanje masa na temelju podataka o sekvenci, izračunavanja načina na koji će enzim tripsin kidati proteine od interesa, računati hidrofobnost proteina na temelju njihove sekvence, stvaranje izoformi iz danih podataka od strane UniProta te cijeli niz pretraživačkih funkcija potrebnih za dobivanje informacija iz takvih baza podataka.

Također je važno spomenuti da takav program ima veliku mogućnost nadograđivanja. Fokus pri izradi aplikacijskog korisničkog sučelja je bio dizajnirati ga takvim da nadogradnja programa bude pristupačnija i jednostavnija. To je napravljeno na način da opseg funkcija može biti i veći, no to bi izašlo iz okvira ovoga rada. Kao neka od mogućih proširenja mogla bi se spomenuti proširenja u vidu dodavanja drugih enzima osim tripsina, dodavanja drugih izračuna vezano za strukturu, izoelektrične točke i sličnih funkcija.

5 ZAKLJUČCI

Cilj ovog rada je bio razviti alat koji pojednostavljuje korištenje UniProtove baze podataka i uspješno rješava njezine nedostatke. Tako je ovim radom uspješno napravljen program koji ima mogućnost pretraživanja jednog ili više proteina za njihova svojstva od interesa. Program u sebi sadrži alate za rad na pronađenim podacima s ciljem davanja još detaljnijeg opisa proteina od interesa.

Analizom glikoproteina određeno je da su peptidi nastali tripsinizacijom proteina humane plazme većinom manji od 100 aminokiselina i manji od 10 kDa. Potvrđeno je da su takvi peptidi hidrofilni kako je i očekivano jer se nalaze na površini globularnih proteina. U konačnici je pokazano kako je N-glikozilacija najzastupljeniji tip glikozilacije što je bilo i očekivano.

Programom su razvijeni alati za:

- precizno računanje masa proteina
- računanje enzimske digestije proteina i stvaranja informacija o takvim ostacima
- računanje prosječne hidrofobnosti proteina
- pretraživanje svojstava proteina poput imena, kanonske sekvence, glikozilacijskih mjesta, informacijama potrebnih za stvaranje izformi
- računanje izoformi
- računanje potencijalnih glikozilacijskih mjesta na izoformi iz kanonske sekvence
- stvaranje korisničkih sučelja za jednostavnije korištenje i bolju otvorenost prema korisnicima u vidu konzole i grafičkog sučelja

6 LITERATURA

Akeret J, Gamper L, Amara A, Refregier A. HOPE: A Python Just-In-Time compiler for astrophysical computations. *Astron Comput*, 2015, 10, 1-18.

Breitling J, Aebi M. N-linked protein glycosylation in the endoplasmic reticulum. *Cold Spring Harb Perspect Biol*, 2013, 5, 1, 1-15.

Chandler KB, Costello CE. Glycomics and glycoproteomics of membrane proteins and cell-surface receptors: Present trends and future opportunities. *Electrophoresis*, 2016, 37, 1407-1419.

Cummings RD, Etzler ME. Antibodies and lectins in glycan analysis. U: Essentials of Glycobiology. Varki A, Cummings RD, Esko JD, Freeze HH, Stanley P, Bertozzi CR, Hart GW, Etzler ME, urednici, New York, Cold Spring Harbor Laboratory Press, 2009, str. 633-649.

Vistoli G, De Maddis D, Cipak A, Zarkovic N, Carini M, Aldini G, Advanced glycoxidation and lipoxidation end products (AGEs and ALEs) an overview of their mechanisms of formation. *Free Radic Res*, 2013, 1, 3-27.

Cummings RD, Pierce JM. The Challenge and Promise of Glycomics. *Chem Biol*, 2014, 21, 1-15.

Francisco R, Pascoal C, Marques-da-Silva D, Morava E, Gole GA, Coman D, Jaeken J, Dos Reis Ferreira V. Keeping an eye on congenital disorders of O-glycosylation: A systematic literature review. *J Inherit Metab Dis*, 2019, 42, 29-48.

Gemmill TR, Trimble RB. Overview of N- and O-linked oligosaccharide structures found in various yeast species. *Biochim Biophys Acta Gen Subj*, 1999, 1426, 227-237.

General Python FAQ, 2001., https://docs.python.org/3.6/faq/general.html, pristupljeno 26.6.2019.

Goettig P. Effects of Glycosylation on the Enzymatic Activity and Mechanisms of Proteases. *Int J Mol Sci*, 2016, 17, 1-24.

Hossain M, Kaleta DT, Robinson EW, Liu T, Zhao R, Page JS, Kelly RT, Moore RJ, Tang K, Camp DG, Qian WJ, Smith RD. Enhanced Sensitivity for Selected Reaction Monitoring Mass Spectrometry-based Targeted Proteomics Using a Dual Stage Electrodynamic Ion Funnel Interface. *Mol Cell Proteomics*, 2011, 10, 1-9.

Ilg T. Proteophosphoglycans of Leishmania. *Parasitol Today*, 2000, 16, 489-497.

Khoury GA, Baliban RC, Floudas CA. Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database. *Sci Rep*, 2011, 1, 1-5.

Kindler E, Krivy I. Object-oriented simulation of systems with sophisticated control. *Int J Gen Syst*, 2011, 40, 313-343.

Kobs G. Improved Characterization and Quantification of Complex Cell Surface N-Glycans, 2015., https://www.promegaconnections.com/improved-characterization-and-quantification-of-complex-cell-surface-n-glycans/, pristupljeno 25.4.2019.

Liu BA, Engelmann B, Nash PD. High-throughput analysis of peptide binding modules. *Biol Chem*, 2012, 12, 1527-1546.

Lopez PHH, Schnaar RL. Determination of Glycolipid-Protein Interaction Specificity. *Methods Enzymol*, 2006, 417, 205-220.

Pan S, Chen R, Aebersold R, Brentnall TA. Mass Spectrometry Based Glycoproteomics - From a Proteomics Perspective. *Mol Cell Proteomics*, 2011, 10, 1-14.

Rudd P, Karlsson NG, Khoo KH, Packer NH. Glycomics and Glycoproteomics. U: Essentials of Glycobiology. Varki A, Cummings RD, Esko JD, Freeze HH, Stanley P, Bertozzi CR, Hart GW, Etzler ME, urednici, New York, Cold Spring Harbor Laboratory Press, 2017, str. 653-657.

Sacks DL, Modi G, Rowton E, Späth G, Epstein L, Turco SJ, Beverley SM. The role of phosphoglycans in Leishmania-sand fly interactions. *Proc Natl Acad Sci U S A*, 2000, 97, 406-411.

Schnaar RL. Glycolipid-mediated cell-cell recognition in inflammation and nerve regeneration. *Arch Biochem Biophys*, 2004, 426, 163-172.

Smith LM, Kelleher NL. Proteoform: a single term describing protein complexity. *Nat Methods*, 2013, 10, 186-187.

Voragen AGJ, Coenen GJ, Verhoef RP, Schols HA. Pectin a versatile polysaccharide present in plant cell walls. *Struct Chem,* 2009, 20, 263-275.

Zen of python, 2004., https://www.python.org/dev/peps/pep-0020/, pristupljeno 26.6.2019.

7 SAŽETAK/SUMMARY

7.1 SAŽETAK

Glikozilacija je skup čestih i složenih kotranslacijskih/postranslacijskih kovalentih modifikacija lipida i proteina. One su od velike važnosti jer utječu na funkciju i strukturu takvih kompleksnih molekula. Istraživanja kompleksnih molekula su vrlo zahtjevna i često koriste kombinacije metoda. Jedna od važnih grupa metoda koje se koriste u takvim istraživanjima su bioinformatičke metode. One se uglavnom fokusiraju na analizu podataka i simulacije. Kako raste broj radova na tu temu, raste i količina podataka koje bioinformatički alati trebaju koristiti te je nužno razvijati nove alate koji će omogućiti lakše i brže istraživanje složenih molekula.

Ovaj diplomski rad je pokušaj razvijanja takvog alata koji olakšava pretraživanje glikoproteoma UniProt baze podataka. Razvijeni alat je testiran na 592 anotirana glikoproteina humane plazme. Računalnom analizom tih glikoproteina utvrđeno je da su oni većinom manji od 1000 aminokiselina i mase obično ispod 100 kDa. Također je utvrđeno da su njihove proteinske strukture hidrofilne i da je najčešći tip glikozilacije zabilježenih proteina humane plazme N-glikozilacija. Analiza je provedena i na peptidima proteina nastalih računalnom tripsinskom razgradnjom proteina. Tom analizom je utvrđeno da je većina takvih peptida neglikozilirana, većinom manje mase od 10 kDa, odnosno duljine 100 aminokiselina te da su manje hidrofilni od izvornih glikoproteina od kojih potječu.

7.2 SUMMARY

Glycosylation is a collection of complex cotranslational/posttranslational covalent modifications of lipids and proteins. These modifications are of great importance as they affect the function and structure of such complex molecules. Researching complex molecules is very demanding and often requires combination of different types of methods. One such group includes bioinformatic methods whose main goals are data analysis and simulations. As the number of research papers grows, the amount of data on these molecules grows as well which consequently requires better and faster tools for data analysis.

This Master's thesis was an attempt of developing such a tool to improve the ones already available for UniProt database. The developed tool was tested on a set of 592 annotated human plasma glycoproteins. Computer analysis of the aforementioned set yielded the insight that most of these proteins were smaller than 1000 amino acids with masses under 100 kDa. It has also showed that most of these glycoprotein structures are hydrophilic in nature and that by far the most common type of glycosylation is the N-glycosylation. Analysis of peptides acquried by computer trypsine digestion was conducted as well. It showed that most of such peptides are much smaller than their glycoprotein counterparts, mostly ranging in size under 10 kDa or 100 amino acids in length and are less hydrophilic than full glycoproteomic structures.

8 PRILOZI

8.1 POPIS KRATICA

IUPAC – Međunarodna unija za čistu i primijenjenu kemiju

GAG – glikozaminoglikani

ER – endoplazmatski reticulum

GPI – glikozilfosfatidilinozitol

Asn – asparagin

Arg – arginin

Dol-P-P-GlcNAc – dolikol pirofosfat N-acetilglukozamin

UDP-GlcNAc – uridin difosfat N-acetilglukozamin

Man – manoza

glukoza – Glc

GA – Golgijev aparat

GalNAc – N-acetilgalaktozamin

Fuc – fukoza

NeuAc – sijalinska kiselina, N-acetilneuraminska kiselina

Xaa – opća oznaka za aminokiseline (osim prolina)

Pro – prolin

Thr – treonin

Ser – serin

Cys – cistein

Tyr – tirozin

Gal – galaktoza

Xyl – ksiloza

Trp – triptofan

AGE – krajnji produkti napredne glikacije

DNA – deoksiribonukleinska kiselina

GlcA – D-glukuronska kiselina

IdoA – L-iduronska kiselina

GBP – glikan vezujući proteini

HPLC – tekućinska kromatografija visoke djelotvornosti

CE – kapilarna elektroforeza

MS – spektrometrija masa

LC – tekućinska kromatografija

SILAC – stabilni izotopi aminokiselina u reagensima staničnih kultura

SRM – slektirano praćenje reakcija

ENA – Europski arhiv nukleotida

UniProtKB – UniProt baza znanja

UniParc – UniProt arhiva

UniRef – UniProt baza podataka koja uklanja redundantne podatke

SIB – Švicarski institute za bioinformatiku

EBI – Europski institute za bioinformatiku

EMBL – Europski laboratorij za molekularnu biologiju

DbC – dizajn ugovorom

API – aplikacijsko programsko sučelje

XML – jezik za označavanje podataka

ABC – apstraktna osnovna klasa

Sec – selenocistein

Temeljna dokumentacijska kartica

Sveučilište u Zagrebu
Farmaceutsko-biokemijski fakultet
Studij: Medicinska biokemija
Zavod za biokemiju i molekularnu biologiju ili
Samostalni kolegij Molekularna biologija s genetičkim inženjerstvom
A. Kovačića 1, 10000 Zagreb, Hrvatska ili druga adresa

Diplomski rad

ODREĐIVANJE GLIKOPROTEINA IZ HUMANE PLAZME

Borna Rapčan

SAŽETAK

Glikozilacija je skup čestih i složenih kotranslacijskih/postranslacijskih kovalentih modifikacija lipida i proteina. One su od velike važnosti jer utječu na funkciju i strukturu takvih kompleksnih molekula. Istraživanja kompleksnih molekula su vrlo zahtjevna i često koriste kombinacije metoda. Jedna od važnih grupa metoda koje se koriste u takvim istraživanjima su bioinformatičke metode. One se uglavnom fokusiraju na analizu podataka i simulacije. Kako raste broj radova na tu temu, raste i količina podataka koje bioinformatički alati trebaju koristiti te je nužno razvijati nove alate koji će omogućiti lakše i brže istraživanje složenih molekula.

Ovaj diplomski rad je pokušaj razvijanja takvog alata koji olakšava pretraživanje glikoproteoma UniProt baze podataka. Razvijeni alat je testiran na 592 anotirana glikoproteina humane plazme. Računalnom analizom tih glikoproteina utvrđeno je da su oni većinom manji od 1000 aminokiselina i mase obično ispod 100 kDa. Također je utvrđeno da su njihove proteinske strukture hidrofilne i da je najčešći tip glikozilacije zabilježenih proteina humane plazme N-glikozilacija. Analiza je provedena i na peptidima proteina nastalih računalnom tripsinskom razgradnjom proteina. Tom analizom je utvrđeno da je većina takvih peptida neglikozilirana, većinom manje mase od 10 kDa, odnosno duljine 100 aminokiselina te da su manje hidrofilni od izvornih glikoproteina od kojih potječu.

Rad je pohranjen u Središnjoj knjižnici Sveučilišta u Zagrebu Farmaceutsko-biokemijskog fakulteta.

Rad sadrži: 70 stranica, 43 grafičkih prikaza, 3 tablica i 38 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Ključne riječi: glikozilacija, glikomika, humana plazma, bioinformatika, python

Mentor: **Dr. sc. Gordan Lauc**, *redoviti profesor Sveučilišta u Zagrebu Farmaceutsko-biokemijskog fakulteta.*

Ocjenjivači: **Dr. sc. Gordan Lauc**, *redoviti profesor Sveučilišta u Zagrebu Farmaceutsko-biokemijskog fakulteta.*
Dr. sc. Lidija Bach Rojecky, *izvanredni profesor Sveučilišta u Zagrebu Farmaceutsko-biokemijskog fakulteta.*
Dr. sc. Nada Vrkić, *izvanredni profesor Sveučilišta u Zagrebu Farmaceutsko-biokemijskog fakulteta.*

Rad prihvaćen: srpanj 2019.

Basic documentation card

University of Zagreb
Faculty of Pharmacy and Biochemistry
Study: Medical biochemistry
Department of Biochemistry and molecular biology or
Independent course Molecular biology with genetical engineering
A. Kovačića 1, 10000 Zagreb, Croatia ili druga adresa

Diploma thesis

DETERMINING HUMAN PLASMA GLYCOPROTEINS

Borna Rapčan

SUMMARY

Glycosylation is a collection of complex cotranslational/posttranslational covalent modifications of lipids and proteins. These modifications are of great importance as they affect the function and structure of such complex molecules. Researching complex molecules is very demanding and often requires combination of different types of methods. One such group includes bioinformatic methods whose main goals are data analysis and simulations. As the number of research papers grows, the amount of data on these molecules grows as well which consequently requires better and faster tools for data analysis.

This Master's thesis was an attempt of developing such a tool to improve the ones already available for UniProt database. The developed tool was tested on a set of 592 annotated human plasma glycoproteins. Computer analysis of the aforementioned set yielded the insight that most of these proteins were smaller than 1000 amino acids with masses under 100 kDa. It has also showed that most of these glycoprotein structures are hydrophilic in nature and that by far the most common type of glycosylation is the N-glycosylation. Analysis of peptides acquried by computer trypsine digestion was conducted as well. It showed that most of such peptides are much smaller than their glycoprotein counterparts, mostly ranging in size under 10 kDa or 100 amino acids in length and are less hydrophilic than full glycoproteomic structures.

The thesis is deposited in the Central Library of the University of Zagreb Faculty of Pharmacy and Biochemistry.

Thesis includes: 70 pages, 43 figures, 3 tables and 38 references. Original is in Croatian language.

Keywords: glycosylation, glycomics, human plasma, bioinformatics, python

Mentor: **Gordan Lauc**, **Ph.D.** *Full Professor,* University of Zagreb Faculty of Pharmacy and Biochemistry

Reviewers: **Gordan Lauc**, **Ph.D.** *Full Professor,* University of Zagreb Faculty of Pharmacy and Biochemistry

Lidija Bach Rojecky, **Ph.D.** *Associate Professor,* University of Zagreb Faculty of Pharmacy and Biochemistry

Nada Vrkić, **Ph.D.** *Associate Professor,* University of Zagreb Faculty of Pharmacy and Biochemistry

The thesis was accepted: July 2019.