Bioinformatika

21. maj 2020..

Glava 1

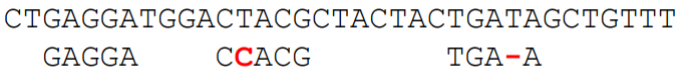
Kako locirati mutacije koje izazivaju bolesti?

U bioinformatici je čest problem ispitivanja da li se jedna ili više kraćih niski pojavljuje unutar duže niske. Zbog brojnih primena, od velikog je značaja da algoritmi za rešavanje ovih problema budu efikasni. Predstavićemo jednu od primena a u nastavku ćemo razmotriti nekoliko načina za rešavanje ovog problema.

1.1 Mapiranje očitavanja

Do sada smo govorili o sekvencioniranju genoma na nivou vrste i pomenuli nekoliko biljnih i zivotinjskih vrsta za koje je sekvencioniran genom, između ostalog i čoveka. Ipak, znamo da je za svaku jedinku genom jedinstven i da ne postoje dve jedinke sa istim genomom, čak ni ako su jednojajčani blizanci. S tim u vezi možemo postaviti sledeće pitanje: kada govorimo o genomu na nivou vrste, da li govorimo o genomu jedne jedinke? Odgovor je negativan. Naime, u projektu prvog sekvencioniranja genoma čoveka, korišćeni su uzorci čak 13 osoba koji su potom sastavljeni u sekvencu koju danas nazivamo *referentnim ljudskim genomom*. Ova sekvenca predstavlja veliko uopštenje pojma genoma čoveka jer između genoma dva čoveka postoje brojne razlike: na oko 3 miliona pozicija (0.1%) postoje supstitucije, zatim insercije i delecije koje se mogu prostirati na nekoliko hiljada nukleotida, kao i izmene pozicija celih gena.

Ipak, bez obzira na velike razlike, referentni genom može poslužiti kao *gotova slika* slagalice koju pokušavamo da sastavimo od očitavanja genoma iz sekvencera. Takođe, referentni genom nam može pružiti uvid u promene koje se dešavaju na određenim genima a koje mogu ukazivati na postojanje različitih genetskih oboljenja ili sklonost ka njima. Neki geni se pojavljuju u različitim formama i slični su kod retko koje dve osobe, a neki su konzervirani kod većine jedinki i promene na njima su od značaja. Da bismo utvrdili razlike, potrebno je da, nakon što dobijemo očitavanja genoma jedne osobe, utvrdimo da li se ona pojavljuju u referentnom genomu koji predstavlja mnogo dužu nisku od očitavanja. Ovaj postupak se često naziva *mapiranje očitavanja* i podrazumeva egzaktno ili približno uparivanje šablona (očitavanja, kraće niske) u tekstu (genoma, duže niske), što je prikazano na slici 1.1.



Slika 1.1: Primer mapiranja očitavanja; gornja niska predstavlja referentni genom, a donje niske očitavanja individualnog genoma

1.2 Egzaktno uparivanje šablona

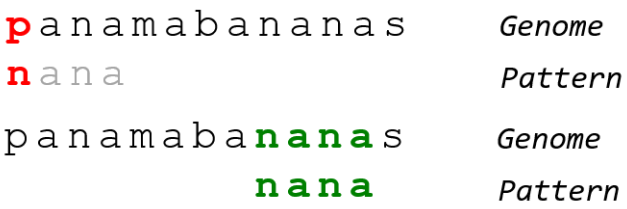
Za početak, posmatraćemo jednostavniji problem pronalaženja pozicija na kojima očitavanja egzaktno poklapaju sa referentnim genomom. Razlikujemo dve verzije ovog problema: jednostruko i višestruko uparivanje šablona.

Problem jednostrukog uparivanja šablona:
Ulaz: Niske *Pattern* i *Genome*.
Izlaz: Sve pozicije u niski *Genome* gde se niska *Pattern* pojavljuje kao podniska.

Problem višestrukog uparivanja šablona:
Ulaz: Kolekcija niski *Patterns* i niska *Genome*.
Izlaz: Sve pozicije u niski *Genome* gde se niske iz kolekcije *Patterns* pojavljuju kao podniske.

1.2.1 Rešenje pomoću grube sile

Ovaj problem se može rešiti grubom silom. Algoritam se sastoji u tome da se linearno krećemo kroz genom i proveravamo da li se dati šablon poklapa sa podniskom genoma iste dužine, koja počinje na toj poziciji (slika 1.2).



Slika 1.2: Uparivanje šablona grubom silom - nepoklapanje i poklapanje

Vreme izvršavanja algoritma u slučaju jednostrukog uparivanja je $O(|Genome| \cdot |Pattern|)$, dok je u slučaju višestrukog $O(|Genome| \cdot |Patterns|)$, gde je $|Patterns|$ suma dužina elemenata liste *Patterns*. Dužine $|Genome|$ i $|Patterns|$ su veoma velike, na primer dužina ljudskog genoma je oko 3 GB, dok ukupna dužina svih očitavanja može biti veća od 1 TB. Zbog toga je algoritam složenosti $O(|Genome| \cdot |Patterns|)$ previše spor.

1.2.2 Rešenje pomoću stabala

Razlog velike neefikasnosti prethodnog algoritma jeste u tome što se za svaki šablon nezavisno ispituje da li se nalazi u genomu. Ako nisku *Genome* zamislimo kao put, onda bi izvršavanje algoritma grube sile bilo analogno vožnji svakog šablona po putu u zasebnom automobilu. Ono što želimo jeste da sve šablone smestimo u jedan autobus, čime bi nam bio dovoljan samo jedan prolazak kroz nisku *Genome*. Ova ideja se može realizovati na dva načina:

1. tako što ćemo sve šablone iz *Patterns* predstaviti jednim stablom (*prefiksno stablo*, eng. *trie*)

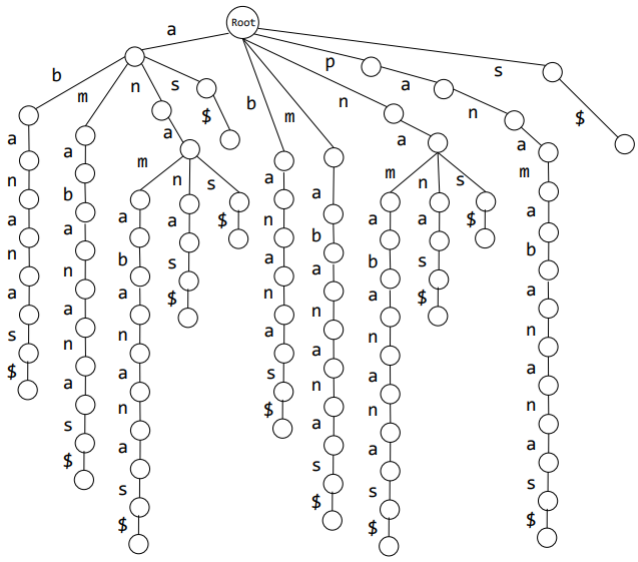
2. tako što ćemo nisku *Genomes* predstaviti jednim stablom (*sufiksno stablo*, eng. *suffix trie*)

U ovom poglavlju ćemo se fokusirati na sufiksna stabla. Rešenje dato preko prefiksnih stabala kao i način njihove konstrukcije mogu se pronaći u materijalima sa vežbi.

Sufiksno stablo za datu nisku date niske predstavlja stablo formiran na osnovu svih sufiksa te niske. Ovo stablo ima sledeće karakteristike:

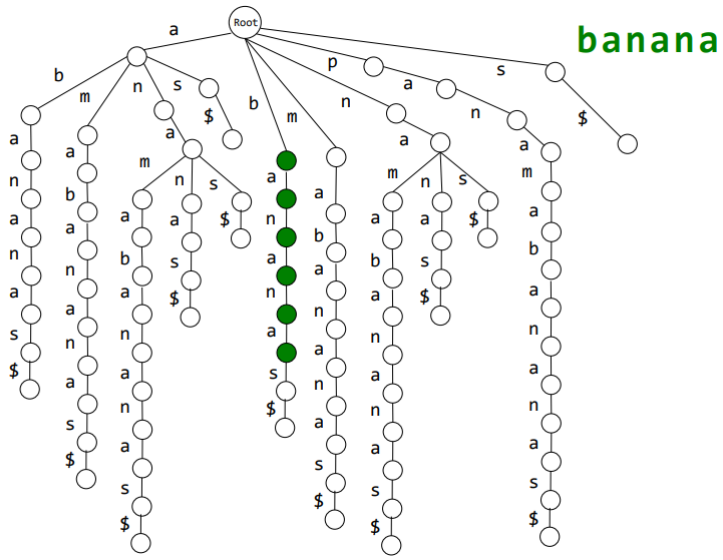
- Svaka grana je obeležena jednim slovom
- Grane koje izlaze iz jednog čvora obeležene su različitim slovima
- Svaka putanja stabla od korena do lista predstavlja jedan sufiks niske
- Svaka putanja stabla od korena do unutrašnjeg čvora predstavlja jednu podnisku niske

Pre početka konstrukcije sufiksnog stabla, na nisku dodajemo simbol $ kao oznaku kraja. Na slici 1.3 nalazi se sufiksno stablo za nisku `panamabananas`.

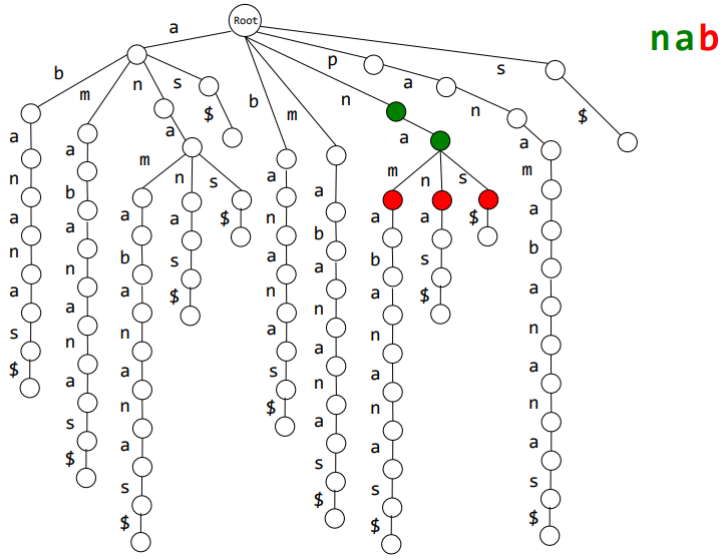


Slika 1.3: Sufiksno stablo za nisku `panamabananas`

Proveru da li se dati šablon nalazi u tekstu vršimo tako što tražimo putanju u sufiksnom stablu koja je jednaka šablonu. Ukoliko dođe do nepoklapanja, pretraga je neuspešna. U suprotnom, pretraga je uspešna. Primeri za obe pretrage su dati na slikama 1.4 i 1.5.

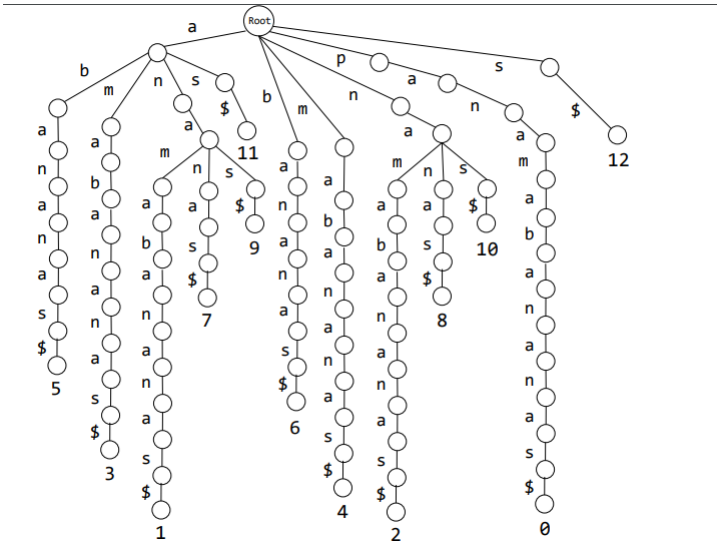


Slika 1.4: Uspešno pronalaženje niske u sufiksnom stablu



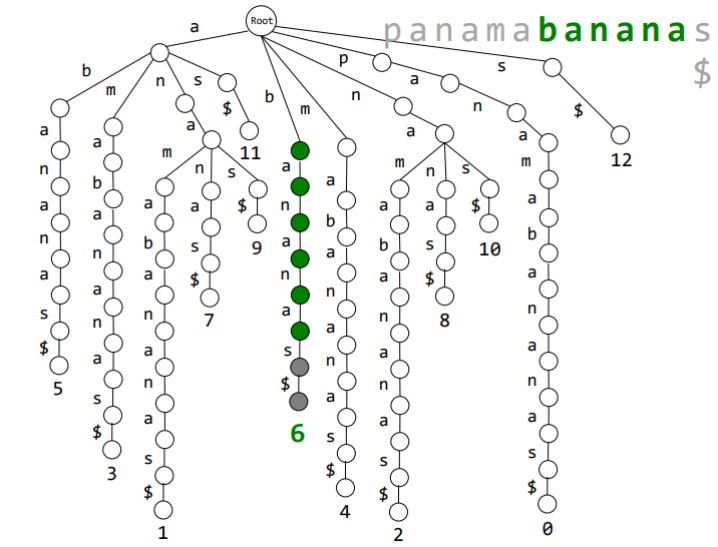
Slika 1.5: Neuspešno pronalaženje niske u sufiksnom stablu

Ovim postupkom možemo utvrditi da li se šablon pojavljuje u tekstu, ali ne i na kojoj poziciji. Za to moramo dodati još informacija u stablo. Svakom listu dodamo početnu poziciju sufiksa u niski *Genome* koji se završava u tom listu (slika 1.6).

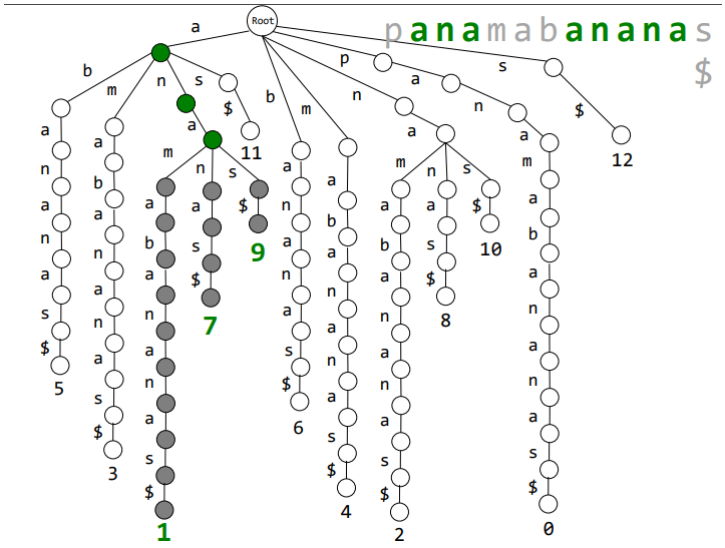


Slika 1.6: Sufiksno stablo sa numerisanim listovima

Sada lako možemo da utvrdimo pozicije u tekstu odakle počinje pojavljivanje šablona. Kada pronađemo uparivanje, nastavimo sa kretanjem naniže do lista, gde je označena pozicija odakle počinje pojavljivanje podniske (slike 1.7 i 1.8).



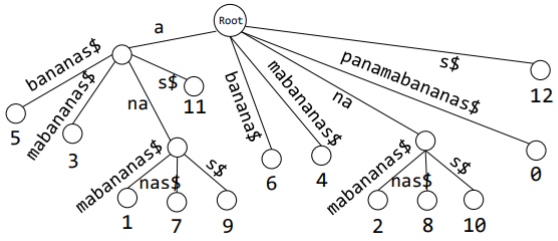
Slika 1.7: Primer nalaženja pozicije nakon uparivanja



Slika 1.8: Primer nalaženja pozicije nakon uparivanja - više poklapanja

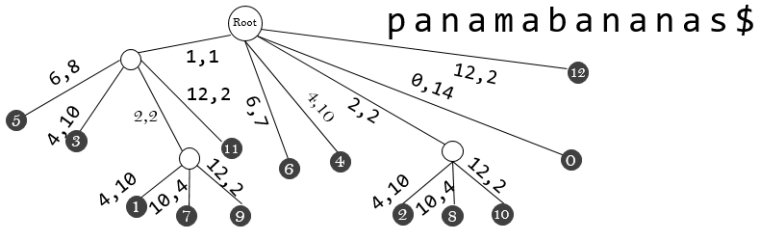
Razmotrimo koliko je prostora potrebno za smeštanje ovakve strukuture. Najviše mesta će zauzeti stablo kod koga svi unutrašnji čvorovi imaju tačno jednog potomka. Za njegovo smeštanje biće neophodno $O(|Suffixes|)$ čvorova gde je $|Suffixes|$ suma dužina svih sufiksa koja za genom dužine n iznosi $n(n+1)/2 = O(n^2)$. To znači da je za smeštanje sufiksnog stabla niske $Genome$ potrebno $O(|Genome|^2)$ prostora.

Prostornu složenost je moguće smanjiti kompresijom grana tako što ćemo sve susedne grane sa ulaznim i izlaznim stepenom 1 spojiti u jednu granu i označiti niskom dobijenom oznakama spojenih grana (slika 1.9). Ovako dobijena struktura se naziva *kompresovano sufiksno stablo* (eng. *suffix tree*). U nastavku ćemo sufiksno stablo nazivati nekompresovanim sufiksnim stablom, a kompresovano samo sufiksnim stablom.



Slika 1.9: Kompresovano sufiksno stablo

Umesto niskama, grane sufiksnog stabla možemo označiti uređenim parovima gde je prvi član indeks početka sufiksa a drugi njegova dužina. Uz to, čuvamo celu nisku $|Genome|$ (slika 1.10).



Slika 1.10: Sufiksno stablo sa granama označenim uređenim parovima

Prostor za smeštanje ovakvog stabla proporcionalan je broju njegovih čvorova. U stablu se nalazi $|Genome|$ listova, po jedan za svaki sufiks. Broj unutrašnjih čvorova je strogo manji od $|Genome|$ jer da nije, to bi značilo da postoji unutrašnji čvor stepena jedan koji je spojen sa listom, a takvi čvorovi su kompresijom eliminisani. Ukupan broj čvorova je stoga $\leq |Genome| + |Genome| - 1$ i linearan je po dužini teksta ($O(|Genome|)$).

Analizirajmo prostornu i vremensku složenost algoritma za konstrukciju sufiksnog stabla (koji se može dobiti prilagođavanjem algoritma za konstrukciju prefikcsnog stabla i koji ovde ne navodimo) i za nalaženje šablona unutar teksta:

- Vremenska složenost:
  - O($|Genome|^2$) za konstrukciju sufiksnog stabla tako što se prvo konstruiše nekompresovano sufiksno stablo.
  - O($|Patterns|$) za nalaženje uparivanja.
- Prostorna složenost:
  - O($|Genome|^2$) za konstrukciju sufiksnog stabla tako što se prvo konstruiše nekompresovano sufiksno stablo.
  - O($|Genome|$) za čuvanje sufiksnog stabla.

Pored toga, postoje algoritmi sa linearnom prostornom i vremenskom složenošću (koje takođe ne pokrivamo):

- Vremenska složenost
  - O($|Genome|$) za konstrukciju sufiksnog stabla direktno.
  - O($|Patterns|$) za nalaženje uparivanja.
  - Ukupno O($|Genome| + |Patterns|$)
- Prostorna složenost:
  - O($|Genome|^2$) za konstrukciju sufiksnog stabla direktno.
  - O($|Patterns|$) za čuvanje sufiksnog stabla.
  - Ukupno O($|Genome|$)

1.2.3 Kompresija niski i Barouz-Vilerova transformacija

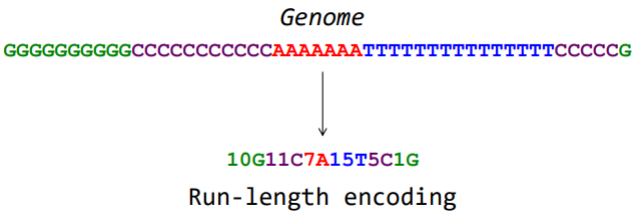
Linearna prostorna i vremenska složenost deluju kao idealno rešenje za problem egzaktnog uparivanja šablona. Međutim, O-notacija ignoriše konstante, a najpoznatija implementacija sufiksnih stabala zahteva $20 \cdot |Genome|$ (npr. veličina humanog genoma je 3GB, pa je za smeštanje sufiksnog stabla potrebno 60 GB što je i dalje unapređenje u odnosu na 1TB). Postavlja se pitanje da li možemo smanjiti faktor konstante. Odgovor nam daje kompresija genoma.

1.2.4 Kompresija genoma

Glavna ideja ovog rešenja jeste da se smanji količina memorije potrebna za čuvanje niske *Genome*. U niski *Genome*1 na slici 1.12 imamo nekoliko uzastopnih ponavljanja jedne aminokiseline koje nazivamo *ranovima* (eng. *runs*): prvo uzastopna ponavljanja aminokiseline G, pa C i tako dalje. U niski *Genome*1 na istoj slici imamo uzastopna ponavljanja nizova aminokiselina koje nazivamo *ripitima* (eng. *repeats*): prvo uzastopna ponavljanja GAC, pa CATT i tako dalje.

Slika 1.11: Ranovi i ripiti

Ponavljanje ranova možemo lako kodirati kao što je prikazano na slici 1.12.

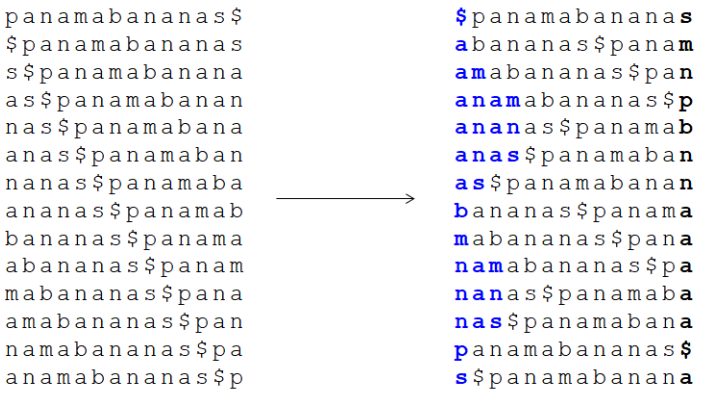


Slika 1.12: Kodiranje dužina ranova

Ovakvo kodiranje ne bismo mogli da primenimo direktno na realne genomske sekvence jer u njima nema mnogo ranova. Sa druge strane, u njima se nalazi veliki broj ripita. To znači da bismo, ako bismo uspeli da ripite konvertujemo u ranove, mogli na tako konvertovanu genomsku sekvencu da primenimo kodiranje dužine ranova. Konverzija ripita u ranove se može izvesti pomoću takozvane *Barouz-Vilerove transformacije* (skraćeno BWT).

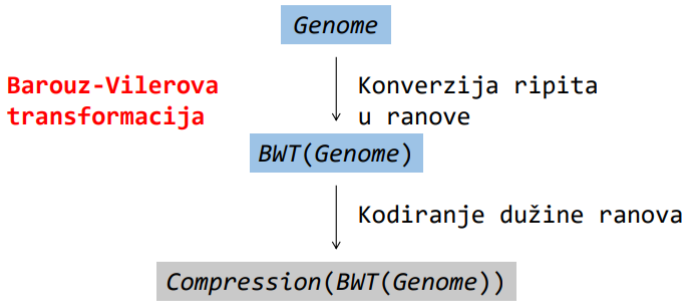
1.2.5 Barouz-Vilerova transformacija

Na slici 1.13 prikazan je postupak za dobijanje BWT niske `panamabananas`. Za datu nisku, potrebno je formirati sve njene ciklične rotacije (levo). Zatim, potrebno je leksikografski sortirati dobijene rotacije (desno). Barouz-Vilerova transformacija predstavlja poslednju kolonu matrice sortiranih rotacija: `smnpbnnaaaaa$a`.



Slika 1.13: Barouz-Vilerova transformacija

Kao što vidimo, poslednja kolona ima veći broj ranova nego izvorna niska i njihove dužine se mogu kodirati. Dakle, kompresija genoma podrazumeva sledeće korake:

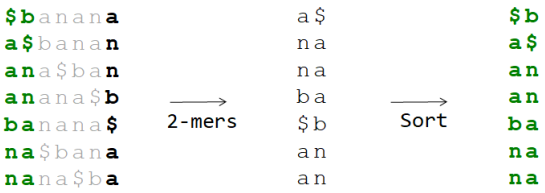


Slika 1.14: Kompresija genoma

Dalje je neophodno definisati kako od kompresovanog genoma dobijamo polazni. Korak dekompresije na osnovu kodiranja dužine ranova je trivijalan. U nastavku ćemo pažnju posvetiti koraku dobijanja polazne niske na osnovu njene BWT.

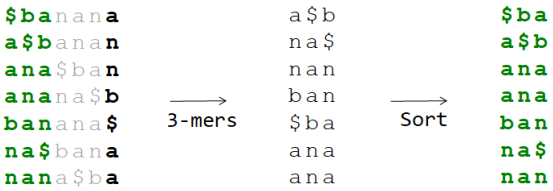
1.2.6 Inverzna BWT

Inverzna BWT biće opisana na slici 1.15 na primeru niske `banana`. Na početku nam je poznata samo njena BWT, niska `annb$aa` (levo, podebljano). Ako ovu nisku sortiramo, dobićemo prvu kolonu matrice rotacija sa leve strane. Spajanjem ove dve kolone dobijamo 2-gramski sastav polazne niske (sredina). Sortiranjem matrice 2-grama dobijamo prve dve kolone matrice rotacija (desno).



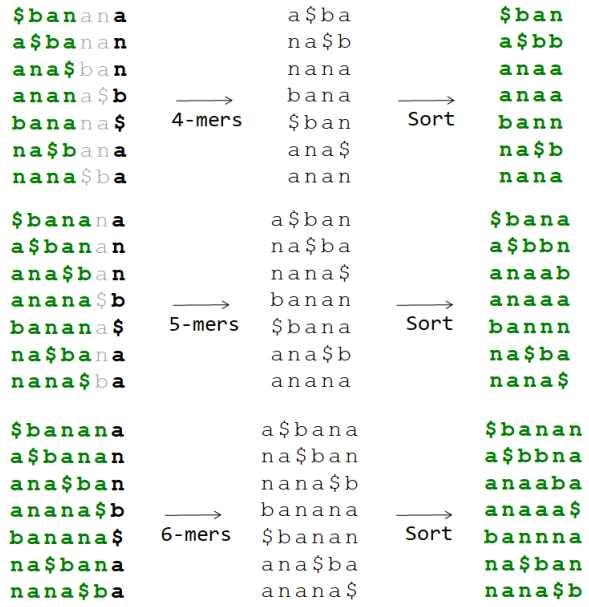
Slika 1.15: Prvi korak inverzne BWT

U sledećem koraku, prikazanom na slici 1.16, poznati su BWT i prve dve kolone matrice rotacija polazne niske. Spajanjem BWT sa njima dobijamo 3-gramski sastav polazne niske (sredina). Sortiranjem matrice 3-grama dobijamo prve tri kolone matrice rotacija (desno).



Slika 1.16: Drugi korak inverzne BWT

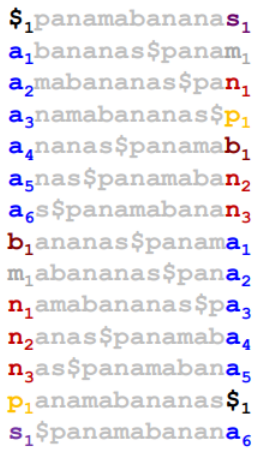
Postupak nastavljamo dok ne obradimo sve kolone matrice rotacija (slika 1.17). Polazna niska je u prvom redu poslednje matrice rotacija (levo), iza simbola $.



Slika 1.17: Preostali koraci inverzne BWT

Analizirajmo koliko je prostora potrebno za rekonstrukciju niske na osnovu BWT. Kao što smo videli, potrebno je da čuvamo celu matricu rotacija čija dimenzija je $|Genome|^2$. To je kvadratna složenost a naša želja je bila da poboljšamo faktor konstante kod algoritma linearne složenosti. Postavlja se pitanje da li se inverzna BWT može efikasnije implementirati.

Uočimo sledeće svojstvo kod matrice rotacija: k-to pojavljivanje simbola u prvoj koloni i k-to pojavljivanje simbola u poslednjoj koloni odgovaraju istoj poziciji simbola u niski Genome. Zaista, prvo pojavljivanje simbola a u prvoj koloni, označeno sa a_1, predstavlja 6. karakter u niski `panamabananas` (ako brojimo od 1), a prvo pojavljivanje ovog simbola u poslednjoj koloni predstavlja isti karakter (slika 1.18). Isto važi i za ostale simbole u prvoj i poslednjoj koloni matrice rotacija. Ovo svojstvo se naziva *first-last* svojstvo i može posližiti sa efikasnije nalaženje inverzne BWT.



Slika 1.18: First-Last svojstvo

Ovo svojstvo omogućava rekonstrukciju polazne niske samo na osnovu prve i poslednje kolone matrice rotacija. Postupak je opisan za primer na slici 1.18:

- Počnemo od simbola $\$_1$ u prvoj koloni. Potražimo poslednji simbol u istom redu, to je s_1. Njihovim spajanjem dobijamo poslednja dva simbola niske, $s\$$
- Isti postupak sprovodimo u onom redu u kom je prvi simbol s_1. Poslednji simbol u tom redu je a_6. Spajanjem ovog simbola sa ostatkom niske dobijamo poslednja tri simbola niske $as\$$.
- Postupak nastavljamo i završavamo kada dođemo do reda čiji je poslednji simbol $\$_1$.

Na ovaj način je niska rekonstruisana samo na osnovu dve niske dužine $|Genome|$ za čije je smeštanje potrebno $2|Genome|$ prostora, što predstavlja poboljšanje u odnosu na oko $20|Genome|$ kod sufiksnih stabala.

1.2.7 Korišćenje BWT za uparivanje šablona

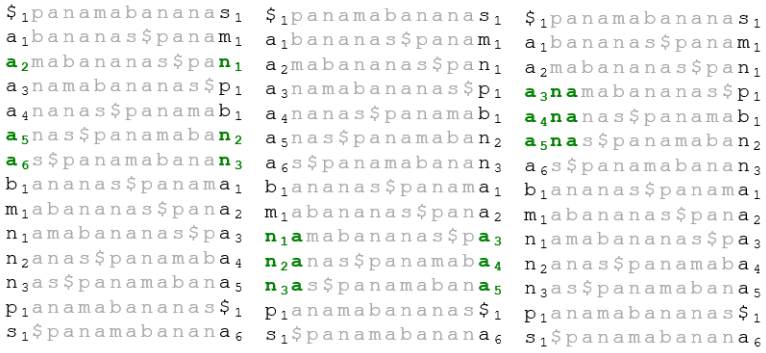
Da se podsetimo, uparivanje šablona korišćenjem sufiksnih stabala zahtevalo je vremensku složenost od $O(|Genome| + |Patterns|)$, prostorna $O(|Genome|)$, uz napomenu da je u praktičnim primerima sufiksno stablo tražilo $20|Genome|$ prostora. Korišćenje BWT je donelo poboljšanje usled smanjene prostorne složenosti. Razmotrimo sada kako možemo utvrditi da li se šablon pojavljuje u tekstu ako je tekst predstavljen pomoću BWT.

Postupak će biti opisan na primeru traženja šablona `ana` u tekstu `panamabananas` (slika 1.19):

- Počinjemo od poslednjeg simbola šablona, `a`, i u prvoj koloni pronađemo sve ovakve simbole. Na istim pozicijama posmatramo simbole u poslednjoj koloni i tražimo da li je neki simbol jednak drugom simbolu šablona, `n` (levo)

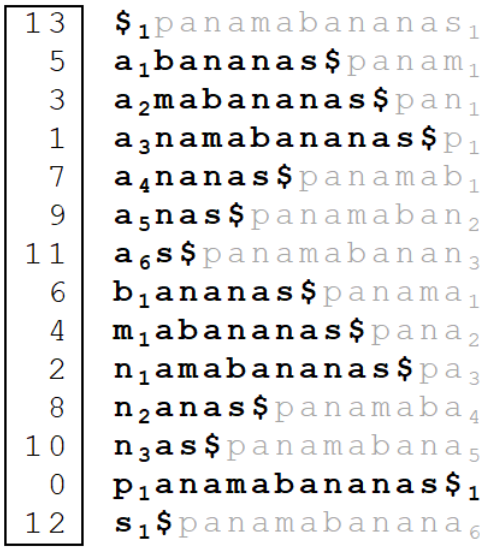
- Ukoliko postoji bar jedan simbol u poslednjoj koloni koji je jednak simbolu `n`, nastavljamo isti postupak od rednog broja tog simbola u prvoj koloni (sredina). Korektnost ovog koraka nam obezbeđuje *first-last* svojstvo.

- Postupak nastavljamo dok postoji poklapanje ili dok ne ispitamo sve simbole iz šablona (levo).



Slika 1.19: Traženje šablona u tekstu predstavljenom kao BWT

Na osnovu ovog postupka možemo utvrditi da se dati šablon tri puta pojavljuje u datom tekstu ali ne i na kojim pozicijama. Pri određivanju ove informacije može nam pomoći *sufiksni niz*. Vrednosti sufiksnog niza odgovaraju početnim pozicijama sufiksa u tekstu, u redosledu kojim su navedeni u matrici rotacije (slika 1.20). Možemo videti da se šablon `ana` pojavljuje na pozicijama 1, 7 i 9.



Slika 1.20: Sufiksni niz

Prostorna složenost za smeštanje sufiksnog niza je $4|Genome|$ (ako koristimo 4B za cele brojeve kao elemente niza). Algoritam za formiranje sufiksnog niza možete pogledati u materijalima sa vežbi.

1.3 Približno uparivanje šablona

Do sada smo razmatrali problem egzaktnog uparivanja šablona koji je podrazumevao da za dati šablom utvrdimo da li se pojavljuje u tekstu ili ne. U praktičnim primenama je to vrlo retko i dopuštena su odstupanja, odnosno korisno je pretraživati da se *skoro svi* simboli šablona pojavljuju u tekstu u istom redosledu. Ovakav problem se naziva *problemom približnog uparivanja šablona*.

Problem približnog uparivanja šablona
Ulaz: Niska *Pattern*, niska *Genome*, ceo broj d (kod višestrukog uparivanja ulaz je kolekcija niski *Patterns*).
Izlaz: Sve pozicije niske *Genome*, gde se niska *Pattern* pojavljuje kao podniska sa najviše d razlika.

Pretragu šablona možemo realizovati na sličan način preko BWT i sufiksnog niza, kao i pre, s tim što sada prihvatamo i kad imamo različite karaktere, sve dok je broj razlika $\leq d$. Primer je prikazan na slikama 1.21 i 1.22.

	# Mismatches
$$_1$panamabananas$_1$	
a$_1$bananas$$panam$_1$	1
a$_2$mabananas$$pan$_1$	0
a$_3$namabananas$$p$_1$	1
a$_4$nanas$$panamab$_1$	1
a$_5$nas$$panamaban$_2$	0
a$_6$s$$panamabanan$_3$	0
b$_1$ananas$$panama$_1$	
m$_1$abananas$$pana$_2$	
n$_1$amabananas$$pa$_3$	
n$_2$anas$$panamaba$_4$	
n$_3$as$$panamabana$_5$	
p$_1$anamabananas$$_1$	
s$_1$$panamabanana$_6$	

	# Mismatches
$$_1$panamabananas$_1$	
a$_1$bananas$$panam$_1$	
a$_2$mabananas$$pan$_1$	
a$_3$namabananas$$p$_1$	
a$_4$nanas$$panamab$_1$	
a$_5$nas$$panamaban$_2$	
a$_6$s$$panamabanan$_3$	
b$_1$ananas$$panama$_1$	1
m$_1$abananas$$pana$_2$	1
n$_1$amabananas$$pa$_3$	0
n$_2$anas$$panamaba$_4$	0
n$_3$as$$panamabana$_5$	0
p$_1$anamabananas$$_1$	1
s$_1$$panamabanana$_6$	

Slika 1.21

	# Mismatches
$$_1$panamabananas$_1$	
a$_1$bananas$$panam$_1$	
a$_2$mabananas$$pan$_1$	
a$_3$namabananas$$p$_1$	
a$_4$nanas$$panamab$_1$	
a$_5$nas$$panamaban$_2$	
a$_6$s$$panamabanan$_3$	
b$_1$ananas$$panama$_1$	1
m$_1$abananas$$pana$_2$	1
n$_1$amabananas$$pa$_3$	0
n$_2$anas$$panamaba$_4$	0
n$_3$as$$panamabana$_5$	0
p$_1$anamabananas$$_1$	2
s$_1$$panamabanana$_6$	

	# Mismatches
$$_1$panamabananas$_1$	
a$_1$bananas$$panam$_1$	1
a$_2$mabananas$$pan$_1$	1
a$_3$namabananas$$p$_1$	0
a$_4$nanas$$panamab$_1$	0
a$_5$nas$$panamaban$_2$	0
a$_6$s$$panamabanan$_3$	
b$_1$ananas$$panama$_1$	
m$_1$abananas$$pana$_2$	
n$_1$amabananas$$pa$_3$	
n$_2$anas$$panamaba$_4$	
n$_3$as$$panamabana$_5$	
p$_1$anamabananas$$_1$	
s$_1$$panamabanana$_6$	

Slika 1.22

Literatura