

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

PROJEKT IZ BIOINFORMATIKE

Računanje najduljeg zajedničkog prefiksa temeljeno na BWT

Autori

ZVONIMIR JURELINAC, TOMISLAV ŽIVEC, TONKO ČUPIĆ

Voditelj

doc.dr.sc MIRJANA DOMAZET- LOŠO

Zagreb, prosinac 2017.

Sadržaj

1	Uvod	2
2	Algoritmi	3
2.1	Podatkovne strukture	3
2.1.1	Sufiksno polje	3
2.1.2	Burrows- Wheelerova transformacija (BWT)	3
2.1.3	Stablo valića	5
3	Primjer rada algoritma	5
3.1	Izgradnja sufiksnog polja	5

1 Uvod

Bioinformatika je grana znanosti koja usko povezuje biologiju i računarstvo, a ubrzano se razvijala zadnja dva desetljeća. Pojeftinjenje i sve veća dostupnost tehnologije sekvenciranja rezultirale su stvaranjem velikih skupova bioloških podataka. Često se kao zadatak u bioinformatici nameće analiza sekvence genoma. Pošto su te sekvence predugačke za uobičajenu pohranu i analizu, potrebno je koristiti posebna sufiksna polja i polja najdužeg zajedničkog prefiksa. Cilj projekta je bio implementirati algoritme 1 i 2 iz rada Beller et al. (2013), koristeći gotovu knjižnicu za izgradnju sufiksnog polja. Zatim je implementacija testirana sa našom implementacijom stabla valića i njegove funkcije rang. Rješenje je uspoređeno s rezultatima iz studentskog rada Mrčela et al. (2016). Kao ulazni niz koristili smo genom bakterije E. Coli

2 Algoritmi

U problemima analiza sekvenci vrlo često se javlja potreba za izračunom najduljeg zajedničkog prefiksa (LCP). U tu svrhu koriste sufiksni nizovi koji spremaju u listu spremaju sve moguće sufikse sekvence, od duljine 1 do najdulje. Iz sufiksnog niza dobiva se LCP polje u linearnom vremenu. Veliki resursni zahtjevi analize sekvenci DNK nameću zahtjeve za korištenjem podatkovnih struktura koje koriste manje memorijskog prostora. Iz te potrebe razvijeno je stablo valića niza transformiranog Burrows- Wheelerovom transformacijom (BWT). Metoda je sljedeća: sekvenca se prvo podliježe Burrows- Wheelerovoj transformaciji, potom se transformirana sekvenca sprema u stablo valića. Stablo valića podržava pretraživanje unatrag po originalnom nizu pa tako dobivamo tražene sufikse. Opisani algoritam ima složenost $O(n \log \sigma)$, gdje je σ veličina abecede.

2.1 Podatkovne strukture

2.1.1 Sufiksno polje

Sufiksno polje sadrži sve sufikse od teksta S . Zamijenila je u toj zadaći sufiksno stablo jer koristi manje memorije od sufiksnog stabla za istu zadaću. Sufiksno polje se pokazalo kao vrlo važan alat u raspoznavanju i analizi teksta, bioinformatičari i drugim primjenama. Sufiksno polje SA od niza znakova S je cjelobrojno polje u intervalu od 1 do n koje određuje leksikografski poredak svih n sufiksa niza znakova S . Preciznije, sufiksno polje zadovoljava $S_{SA[1]} < S_{SA[2]} < \dots < S_{SA[n]}$, gdje S_i označava i -ti sufiks niza znakova S , te sadrži znakove $S[i..n]$.

2.1.2 Burrows- Wheelerova transformacija (BWT)

Burrows- Wheelerova transformacija pretvara niz znakova u sličan niz znakova $BWT[1..n]$ sa istom abecedom. Koristi se za kompresiju podataka. Transformacija je reverzibilna bez dodatnih resursnih zahtjeva. Elemente transformiranog niza računamo formulom:

Tablica 1: Pridruživanje indeksa sufiksima niza S , počevši od najduljeg.

i	$S_{SA}[i]$
1	abrakadabra\$
2	brakadabra\$
3	rakadabra\$
4	akadabra\$
5	kadabra\$
6	adabra\$
7	dabra\$
8	abra\$
9	bra\$
10	ra\$
11	a\$
12	\$

Tablica 2: Sufiksi su poredani leksikografski, a njihovi indeksi čine sufiksno polje SA .

i	$SA[i]$	$S_{SA}[i]$	$BWT[i]$
1	12	\$	a
2	11	a\$	r
3	8	abra\$	d
4	1	abrakadabras\$	\$
5	6	adabra\$	k
6	4	akadabra\$	r
7	9	bra\$	a
8	2	brakadabra\$	a
9	7	dabra\$	a
10	5	kadabra\$	a
11	10	ra\$	b
12	3	rakadabra\$	b

Tablica 3: Pridruživanje indeksa sufiksima niza S

i	$S_{SA}[i]$
1	mississippi\$
2	ississippi\$
3	ssissippi\$
4	sissippi\$
5	issippi\$
6	ssippi\$
7	sippi\$
8	ippi\$
9	ppi\$
10	pi\$
11	i\$
12	\$

2.1.3 Stablo valića

Stablo valića je struktura podataka koja rekurzivno particionira tok u 2 dijela sve dok su u svakom dijelu preostali homogeni podatci. Ime stabla je analogno valnoj transformaciji signala koji rekurzivno dekompresira signal prema frekvencijama komponenti.

Stablo valića može efikasno raditi upite rank i select nad nizovima proizvoljnih abeceda. To nam omogućuje pretraživanje unatrag u vremenskoj složenosti od $O(\log \sigma)$ po koraku. Prvo definiramo uzlazno poredanu abecedu η kao niz znakova veličine σ . Zatim definiramo interval $[1..r]$ kao podinterval abecede, $r \leq \sigma$. Za interval $[1..r]$, niz znakova $BWT[1..r]$ dobijemo tako da iz transformiranog niza znakova od S uklonimo sve znakove iz B-W transformacije koji ne pripadaju segmentu abecede $[1..r]$

3 Primjer rada algoritma

U nastavku dokumenta dan je primjer rada algoritma na stringu $S = \text{mississippi}$.

3.1 Izgradnja sufiksnog polja

1. Na kraj ulaznog niza dodaje se znak \$ te je sada $S = \text{mississippi\$}$. U daljnjem tekstu vrijedi pretpostavka da je znak \$ abecedno manji od svih ostalih znakova od kojih je S izrađen.
2. Svakom sufiksu niza S pridružuju se indeksi od 1 do n , počevši od najduljeg. Ovo je prikazano u **tablici 3**.
3. Sufiksno polje SA dobivamo soritiranjem dobivenih sufiksa leksikografski od najmanjeg prema najvećem. Ovo je prikazano u **tablici 4**.

Tablica 4: Sufiksi su poredani leksikografski, a njihovi indeksi čine sufiksno polje SA.

i	SA[i]	S _{SA} [i]
1	12	\$
2	11	i\$
3	8	ippi\$
4	5	issippi\$
5	2	ississippi\$
6	1	mississippi\$
7	10	pi\$
8	9	ppi\$
9	7	sippi\$
10	4	sissippi\$
11	6	ssippi\$
12	3	ssissippi\$

Dobiveno je sufiksno polje SA = [12, 11, 8, 5, 2, 1, 10, 9, 7, 4, 6, 3]

4. Izračunava se Burrows-Wheelerova transformacija BWT[1..n] za dobiveno sufiksno polje SA prema formuli navedenoj u 2.1.2 .

Npr. BWT[4]=S[SA[4]-1]=S[5-1]=S[4]=s.

Rezultat je prikazan u tablici **tablice 5**.

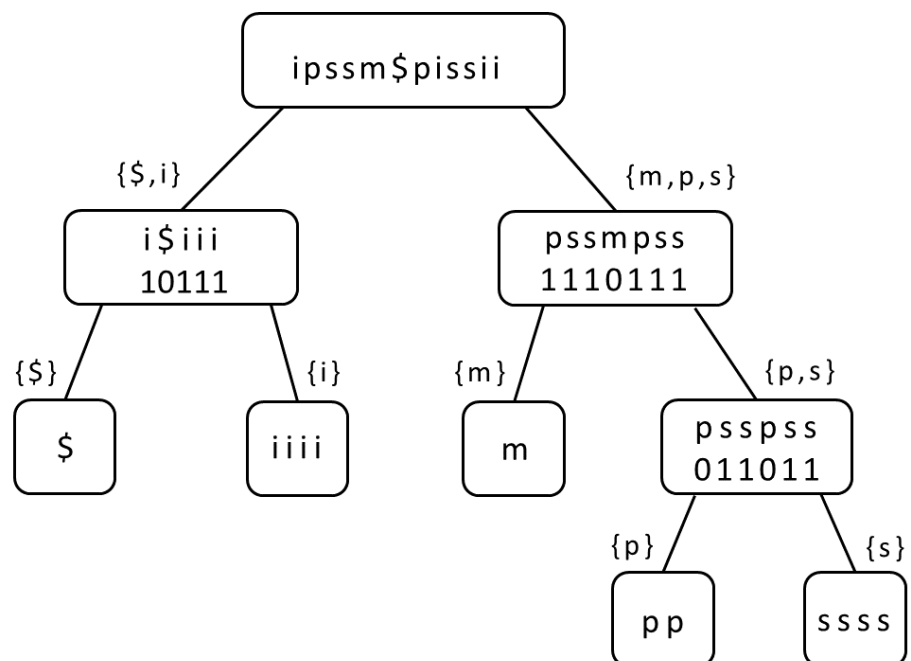
5. Iz dobivene Burrows-Wheelerove transformacije gradi se stablo valića opisanom u 2.1.3. Prvo se stvara sortirana abeceda ulaznog niza koja je u ovom slučaju veličine 5 znakova ($\Sigma[1..5]=\$imps$). Korijen stabla čini bit vektor dobiven kodiranjem niza Burrows-Wheelerove transformacije dobivene u prethodnom koraku. Abeceda se potom dijeli na pola te dobivamo dva podniza, ovom slučaju: $\Sigma[1..2]=\$i$ i $\Sigma[3..5]=mps$. Znakovi prve polovice u svakoj skupini kodiraju se vrijednošću 0, a ostali 1. Postupak se ponavlja sve dok se u čvoru ne nalaze samo jednaki znakovi koji čine listove stabla. Potpuno izgrađeno stablo za dani primjer prikazano je na slici 1.
6. Pomoću Algoritama 1 i 2 iz rada Beller et al. (2013) gradi se polje najdužih zajedničkih prefiksa (LCP) u nekoliko koraka:
- Inicijalizacija LCP polja i reda Q. Vrijednosti polja LCP se postavljaju na nevažeću (**NULL**) vrijednost, osim LCP[1] i LCP[n+1] koji se postavljaju na -1. U red Q stavljamo strukturu koja sadrži početni interval $I = [i..j]=[1..12]$ i broj $l=0$:

$$LCP = [-1, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, -1]$$

$$Q = [<[1..12], 0>]$$
 - Izračunavanje cw-intervalu za interval dobiven uzimanjem elementa iz reda Q (FIFO) unutar funkcije *getIntervals* iz Algoritma 1 Beller et al. (2013).

Tablica 5: Burrows-Wheelerova transformacija

i	SA[i]	$S_{SA}[i]$	BWT[i]
1	12	\$	i
2	11	i\$	p
3	8	ippi\$	s
4	5	issippi\$	s
5	2	ississippi\$	m
6	1	mississippi\$	\$
7	10	pi\$	p
8	9	ppi\$	i
9	7	sippi\$	s
10	4	sissippi\$	s
11	6	ssippi\$	i
12	3	ssissippi\$	i



Slika 1: Stablo valića za dani primjer

c - znak

$C[c]$ - zbroj rangova svih elemenata leksikografski sortirane abecede koji su manji od c

i - početak intervala

j - kraj intervala

- Funkcija $rang(a,k)$ vraća broj pojavljivanja znaka a do k -tog indeksa u polju.
- Znakovi abecede iz intervala I se sortiraju od najmanjeg prema najvećem. Ima onoliko $c\omega$ -intervala koliko je i jedinstvenih znakova. Kako je abeceda za dani primjer $\Sigma[1..5] = \$imps$, traži se 5 $c\omega$ -intervala.
- $C = [0, 1, 5, 6, 8]$
- Indeks početka intervala dobiva se po formuli $rang(c,i-1)+C[c]+1$ Indeks kraja intervala dobiva se po formuli $C[c]+rang(c,j)$.
- U intervalu I se nalaze svi znakovi abecede $(\$i,m,p,s)$. Za svaki od znakova računa se njegov interval prema formuli navedenoj iznad.

$$[rang(' \$',0)+C[' \$']+1..C[' \$']+rang(' \$',12)]=[0+0+1..+0+1]=[1..1]$$

$$[rang(' i',0)+C[' i']+1..C[' i']+rang(' i',12)]=[0+1+1..+1+4]=[2..5]$$

$$[rang(' m',0)+C[' m']+1..C[' m']+rang(' m',12)]=[0+5+1..+5+1]=[6..6]$$

$$[rang(' p',0)+C[' p']+1..C[' p']+rang(' p',12)]=[0+6+1..+6+2]=[7..8]$$

$$[rang(' s',0)+C[' s']+1..C[' s']+rang(' s',12)]=[0+8+1..+8+4]=[9..12]$$

- Povratna vrijednost $getIntervals$, prema tome, je lista $c\omega$ -intervala: $[[1..1],[2..5],[6..6]]$.
- Za svaki od dobivenih intervala $[lb .. rb]$ se potom provjerava vrijednost $LCP[rb+1]$. Ako je vrijednost tog polja **NULL** u red stavljamo strukturu $[<[lb..rb],l+1>]$, a na index $rb+1$ vrijednost l . Navedeni postupak se ponavlja sve dok u redu ima elemenata, a prva dva koraka prikazana su u nastavku.

$$1. \text{ LCP} = [-1, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, \perp, -1] \\ Q = [<[1..1],0>]$$

$$[lb .. rb] = [1..1], l = 0$$

$$LCP[rb+1]=LCP[2]=\perp \rightarrow \text{u red Q stavljamo strukturu } [<[lb..rb],l+1>] \\ = [<[1..1],1>], \text{ a } LCP[rb+1]=LCP[2]=0.$$

$$2. \text{ LCP} = [-1,0,\perp,\perp,\perp,\perp,\perp,\perp,\perp,\perp,\perp,\perp,-1] \\ Q = [<[2..5],0>,<[1..1],1>]$$

$$[lb .. rb] = [2..5], l = 0$$

$$LCP[rb+1]=LCP[6]=\perp \rightarrow \text{u red Q stavljamo strukturu } [<[lb..rb],l+1>] \\ = [<[2..5],1>], \text{ a } LCP[rb+1]=LCP[6]=0.$$

$$3. \text{ LCP} = [-1,0,\perp,\perp,\perp,0,\perp,\perp,\perp,\perp,\perp,\perp,-1] \\ Q = [<[6..6],0>,<[2..5],1>,<[1..1],1>]$$

$$[lb .. rb] = [6..6], l = 0$$

(...)

- (c) Vrijednost LCP polja koju dobijemo kao rezultat izvršavanja algoritma je:
 $LCP = [-1, 0, 1, 1, 4, 0, 0, 1, 0, 2, 1, 3, -1]$.