FM TREE

Potrebno implementirati:

Count() I Locate() metodu.

Locate metodu cemo izmijenti da troši manje memorije I time cemo napraviti FMtree.

**Count() ALG imamo u radu pod Algorithm 1: backward\_serach(P,FM-index)**

**ULAZ:** P-pattern npr (aca) I FM index

FM index se sastoji od F I BWT(T) u (prvi I zadnji stupac iz matrice M):

**Kako gradimo matricu M:**

Uzmemo nas genom I na kraj niza stavimo $ kao znak kraja. I napravimo sve kružne rotacije našeg niza:

Npr: niz acat$

acat$  
$acat  
t$aca  
at$ac  
cat$a

kad smo napravili rotacije treba sortirati abecedno tj od lexicki manjih do leksički vecin nizova(uzmimo u obzir da je $ lexicni najmanji):

i F BWT SA

0 $acat 4

1 acat$ 0

2 at$ac 2

3 cat$a 1

4 t$aba 3

Sada je matrica gotova, prvi stupac je predstavlja F ,a zadnji je BWT, (plava ce biti kasnije objašnjena)

**SA** je sufixsalno polje:

To su zapravo indexi od kuda pocinje niz u originalnom nizu. Npr acat$ pocinje na 0. Indexu

,a $acat na zadnjem, tj. 4.

**IZLAZ:** SA interval koji počinje sa traženim patternom u našem nizu

Za algoritam je potrebno prethodno izračunati **C[s]**, gdje je s slovo, a C[s] je koliko puta se ponavljaju leksicki manja slova od s.

Za naš primjer:

s $ a c t

C[s] 0 1 3 4

Prvo poredamo od lexicni manjih do lex vecih.

Zatim gledamo koliko ima lex manjih od $, pošto je on najmanji to je 0

Od a je jedino lex manji $ stoga gledamo koliko se puta $ pojavljuje, to je 1

Od c su lex manji $ i a i pogledamo koliko se puta a ponavlja posto znamo da se $ ponavlja 1, I to zbrojima a se ponavlja 2x a $ 1x pa je to ukupno 3x. isto tako gledamo za t

Dodatno C[s+1] ,znaci prvi lexicki veci od s. Npr (za nas primjer )ako je s=a C[a+1]=C[c]

Zatim imamo računanje ranka

**Ranks(BWT(T),index)**, racunamo ovako:

Koliko puta se pojavljuje slovo s u BWT(T)[0,index-1]

Npr Ranka(BWT(T),3), koliko puta se pojavljuje a u BWT(T)[0,2]

Za to nam je potrebno broj ponavljana na određenom indexu I to cemo nazvati **Occ(s)**

0 1 2 3 4

BWT(T) t $ c a a

Occ($) 0 1 1 1 1

Occ(a) 0 0 0 1 2

Occ(c) 0 0 1 1 1

Occ(t) 1 1 1 1 1

Sada mozemo lako iscitati iz tablice da je je pojava slova a u BWT(T)[ 0-2 ] = 0

Algoritam LOCATE(sp,ep,FM-index)

Gradimo prema Algorithm 3 : FMtree(P,sp,ep,sp1,ep1,D,FM-index)

**ULAZ**:

sp,ep kao intervali pojave patterna P

sp1,ep1 kao intervali pojave P[1,|P|-1] gdje |P| predstavlja length od P

D kao dubina stabla i FM-index.

Kako ne bismo zauzeli previse memorije SA cemo uzorkovati.

Time SA[i] za neki i necemo moci direktno dohvatiti nego cemo morati doci do njega preko rank funkcija.

Nacin kojim cemo dohvacati te neuzorkovane pozicije zvat cemo FMtree.

Koristit cemo value sampling, tj. uzorkujemo svaki S[i] ako vrijedi SA[i]==0(mod D), gdje je D udaljenost uzorkovanja. S time dobivamo da cemo pri dohvatu neuzorkovane S[j] pozicije biti potrebno samo D-1 koraka tj. Rank izracuna. Npr ako uzmemo za naš primjer da je D=4, plava ce predstavljati uzorkovane pozicije, dok ostale necemo znati vec cemo ih morati izracunati preko plavih.

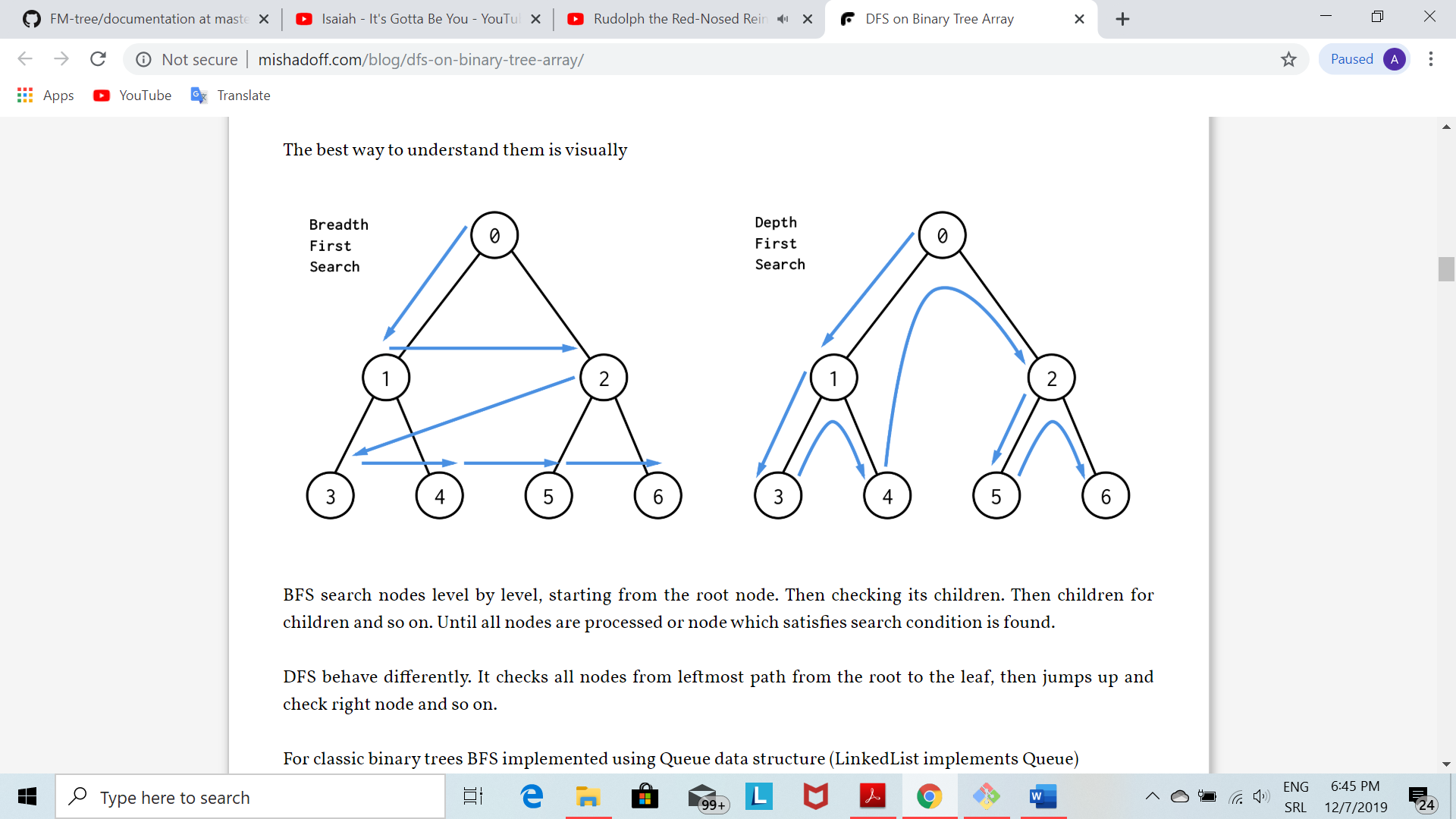
Teorem 1 iz rada pokazuje da ako je sP uzorkovan ( SA[i]) sa value sampling, a mi zelimo saznati SA[j] od P, mozemo izracunati tako da SA[i] dodamo 1, tj. SA[j]=SA[i]+1,

Takodjer ako je ssP (s time da ss mogu biti bilo koje 2 kombinacije iz abecede(a,c,g,t) npr aa,ac,ag…) uzorkovan (SA[i]), SA[j] od P mozemo izracunati da SA[i] dodamo 2, tj SA[j]=SA[i]+2

I u istom smislu vrijedi za sssP, ssssP do s\*(D-1)P(MAX).

S tom idejom gradimo stablo sa slike Fig 2 iz rada gdje je D=3.

Stablo gradimo putem BFS pretrage, Podsjetnik:



S time da je na nultoj razini P, na 1. 1 slovo iz abeced (a,c,g,t) a na 2. Kombinacije 2 slova iz abecede.  
I tako do D razine. Tako putujemo po stablu i kada dodjemo do neke uzorkovane SA[i] (0<i<length(niza T) )njoj dodajemo razinu kako bismo dobili SA[j](sp<j<ep, gdje su sp i ep intervali na kojima se nalazi pattern P s nulte razine) , tj vrijedi SA[j]=SA[i]+razina i spremamo to u neku strukturu(npr lista) koju cemo zvati R. Kada dohvatimo potreban broj uzorkovanih SA[i] tj count(P) prestajemo s pretragom i vracamo R, tj sve lokacije nasih P.

PRIMJER

Pokazat cemo to jednim malo kompliciranijim primjerom: uzmimo da je nas pocetni niz T: ACACATAACA$

Kada napravimo matricu M dobivamo:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  |  |  |
|  | F |  |  |  |  |  |  |  |  |  | BWT |  |  | SA |
| 0 | $ | A | C | A | C | A | T | A | A | C | A |  |  | 10 |
| 1 | A | $ | A | C | A | C | A | T | A | A | C |  |  | 9 |
| 2 | A | A | C | A | $ | A | C | A | C | A | T |  |  | 6 |
| 3 | A | C | A | $ | A | C | A | C | A | T | A |  |  | 7 |
| 4 | A | C | A | C | A | T | A | A | C | A | $ |  |  | 0 |
| 5 | A | C | A | T | A | A | C | A | $ | A | C |  |  | 2 |
| 6 | A | T | A | A | C | A | $ | A | C | A | C |  |  | 4 |
| 7 | C | A | $ | A | C | A | C | A | T | A | A |  |  | 8 |
| 8 | C | A | C | A | T | A | A | C | A | $ | A |  |  | 1 |
| 9 | C | A | T | A | A | C | A | $ | A | C | A |  |  | 3 |
| 10 | T | A | A | C | A | $ | A | C | A | C | A |  |  | 5 |

Tablica1 (matrica M sa SA)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| s | $ | A | C | T |
| C[s] | 0 | 1 | 7 | 10 |

Tablica2 (C[s])

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| BWT(T) | A | C | T | A | $ | C | C | A | A | A | A |
| Occ($) | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Occ(A) | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 |
| Occ(C) | 0 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | 3 | 3 | 3 |
| Occ(T) | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Tablica3 (Occ(s))

Plavom bojom su prikazane uzorkovane pozicije(value sampling za D=4, SA[i]==0(mod 4))

Recimo da trazimo pattern ACA preko count(backtracking) metode odredimo da je range gdje se nalazi

SA[3,5] (narandasta boja)

Count(ACA):

AC**A**s=A  
sp=C[s]=C[A]=1, ep=C[s+1]-1=C[A+1]-1=C[C]-1=7-1=6   
redci koji počinju sa A su [1,6]

A**CA**s=C  
sp=C[s]+rankC[s](BWT(T),sp)=C[C]+rankc(BWT(T),1)=7+0=7  
ep=C[s]+rankC[s](BWT(T),ep+1)-1=C[C]+rankC(BWT(T),6+1)-1=7+3-1=9

Redci koji pocinju sa CA su [7,9]

**ACA**

s=A   
sp=C[A]+rankA(BWT(T),7)=1+2=3  
ep=C[A]+rankA(BWT(T),9+1)-1=1+5-1=5  
Redci u kojima se pojavljuje ACA su [3,5].

Sad kad smo nasli trazeni range gradimo stablo! Slika 1 prikazuje kako bi naše stablo trebalo izgledati.

A close up of a map

Description automatically generated

Slika 1- FM tree

Racunski bi to islo ovako:

U na razini 0. Je nas pattern P 🡪 ACA i za njega imamo sp,ep -> [3,5]  
Pogledamo dali iti jedan redak iz tog intervala ima uzorkovan SA(podsjetnik! uzorkovali smo sa value sampling za D=4 te su oni koji su uzorkovani oznaceni plavom bojom), uočavamo da nam je redak 4 uzorkovan i spremamo u R 🡪 SA[4]+razina=SA[4]+0=0+0=0. Znaci jos moramo naci 2 pozicije tj 3. i 5. Redak.

Racunamo po redu prema BFS redom intervale za:

Za razinu 1.

**A**ACA, **C**ACA,**G**ACA,**T**ACA tj (**A**P,**C**P,**G**P,**T**P)  
npr za S=**A**P 🡪 S=**A**ACA racunamo interval:

sp = C[S[0]]+rankS[0](BWT(T),sp) = C[A]+rankA(BWT(T),3)=1+1=2  
ep = C[S[0]]+rankS[0](BWT(T),ep+1) -1 = C[A]+rankA(BWT(T),5+1) -1=1+2-1=2

Redci gdje se pojavljuje AACA su [2,2] tj samo redak 2.   
Sada pogledamo jeli redak 2 uzorkovan, nije i idemo dalje.

**C**ACA   
sp = C[C]+rankC(BWT(T),3)=7+1=8  
ep= C[C]+rankC(BWT(T),5+1)-1=7+2-1=8   
I dobijamo da se CACA pojavljuje u retku 8. Nije uzorkovana. Idemo dalje.

**G**ACA  
G se uopće kod nas u alfabetu nije spominjalo stoga preskacemo.

**T**ACA  
sp = C[T]+rankT(BWT(T),3)=10+1=11  
ep= C[T]+rankT(BWT(T),5+1)-1=7+2-1=10+1-1=10  
sp > ep. Nema pojave tog patterna.

Sada prelazimo na razinu 2.

Interval za **A**ACA=[2,2]

**AA**ACA 🡪 izracun… (bit ce sp>ep) nigdje se ne pojavljuje

**CA**ACA   
sp = C[C]+rankC(BWT(T),2)=7+1=8  
ep= C[C]+rankC(BWT(T),2+1)-1=7+1-1=7  
sp<ep nema uzorka

**GA**ACA -> nema G

**TA**ACA   
sp = C[T]+rankT(BWT(T),2)=10+0=10  
ep= C[T]+rankT(BWT(T),2+1)-1=10+1-1=10  
TAACA se spominje u 10. Retku. Provjeravamo SA. Nije uzorkovan!

Prelazimo na cvor **C**ACAte je njegov interval [8,8]

**AC**ACA  
sp = C[A]+rankA(BWT(T),8)=1+3=4  
ep= C[A]+rankA(BWT(T),8+1)-1=1+4-1=4  
ACACA se spominje u 4. Retku. Provjeravamo SA. UZORKOVAN! Spremamo u R -> SA[4]+razina=0+2=2. Ostaje nam naci lokacijuza jos samo jedan ACA (2 smo vec spremili u R).

**CC**ACA 🡪 … nema pojave

**GC**ACA 🡪 nema G

**TC**ACA 🡪 nema pojave

Prelazimo na cvor **G**ACA no posto ce svaki sadrzavati G a G nemamo 🡪 preskacemo!

Prelazimo na cvor **T**ACA ..nije bilo intervala za taj niz i preskacemo!

Prelazimo na razinu 3.

Krecemo za cvor **AA**ACA , nije bilo pojave, tj intervala i preskacemo!

Za cvor **CA**ACA je bilo pojave iz intervala [4,4]

**ACA**ACA 🡪 nema pojave

**CCA**ACA 🡪 nema pojave

**GCA**ACA 🡪 nema G

**TCA**ACA 🡪 nema pojave

Za cvor **GA**ACA nije bilo pojave, preskacemo!

Za cvor **TA**ACA je bilo pojave iz intervala [10,10]

**ATA**ACA  
sp = C[A]+rankA(BWT(T),10)=1+5=6  
ep= C[A]+rankA(BWT(T),10+1)-1=1+6-1=6  
ATAACA se pojavljuje u retku 6. Pogledamo SA. UZORKOVAN JE! Dodajemo u R -> SA[6]+razina=4+3=7  
Sad smo dodali sve 3 pojave I zavrsavamo! Vracamo na izlaz R.

OPTIMIZACIJA:

**Early leaf node calculation**

Naime kada pretražujemo stablo, D-1 razina uzima jako puno vremena, broj rank operacija se povecava sa svakom razinom exponencijalno, u svakoj razini se racuna 2x4i rank operacija, gdje je i broj razine, 2 zbog racunanja sp i ep u svakom koraku te 4 za broj slova u abecedi(a,c,g,t). Kada imamo zadnji korak 2x4D-1, broj izracuna je veci od zbroja svih prethodnih izracuna na prethodnim razinama, stoga želimo to nekako rjesiti i zato je tu early leaf node calculation.

Teorem 2 iz rada kaze:  
Ako imamo pattern P, npr P=ACA i ako za P[i,|P|-1] vrijedi P[0,i-1]=T[SA[j] - i, SA[j] - 1], gdje je i=1..D, a gdje je j iz intervala gdje se pojavljuje P[i,|P|-1], .tj [spi,epi], da vrijedi SA[k]=SA[j]-i gdje je k iz intervala u kojem se pojavljuje nas P. Ako to podredimo nasoj zadnjoj razini D-1 dobit cemo da vrijedi:

Izracunamo za P[1,|P|-1] (skuži da je i=1) da je iz intervala [sp1,ep1], ako VRIJEDI  
T[SA[j]-1,SA[j]-1]=T[SA[j]-1] == P[0,1-1]=P[0] i ako je neki od elemenata iz intervala [sp1,ep1] uzorkovan, u R dodajemo S[j]+i = S[j]+1.

Npr. P=ACA  
P[1,|P|-1]=CA, tražim preko count metode interval pojave CA …izracun… dobijamo da je interval [sp1,ep1]=[7,8], T=ACACATAACA$ (podsjetnik! originalni niz)

Idem za j=7  
Prvo pogledam vrijedi li 🡪T[SA[7]-1]=P[0]🡪 T[8-1]=P[0] 🡪 A=A 🡪 VRIJEDI!! Zatim pogledam da li je SA[7] uzorkovan. UZORKOVAN! Dodajem u R -> SA[7]-1=8-1=7

Idem za j=8  
Vrijedi li T[SA[8]-1]=P[0]🡪 T[1-1]=P[0] 🡪 A=A 🡪 VRIJEDI!! Zatim pogledam je li SA[8] uzorkovan. NIJE uzorkovan!

Kraj!

**Drugi tip** optimizacije je da imamo neki threshold i ako je nas interval dobiven iz count(P) (ep-sp+1) manii od zadanog threshold-a onda racunamo lokacije preko Algorthm 2 (stari locate) iz rada, inace ako je veci onda racunamo locate preko FMtree.