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 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 i prvi stupac je F a zadnji je BWT

**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.

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

Za algoritam je potrebno prethodno izračunati **C[s]**, gdje je s slovoa 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 od 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] = npr ako je s=1 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 0 1 1 1

Occ(a) 0 0 0 0 1

Occ(c) 0 0 0 1 1

Occ(t) 0 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 stable I FM-index.