Bioinformatika

Izgradnja sufiksnog stabla Ukkonenovim algoritmom u linearnom vremenu

Projektna dokumentacija

Mentorica: doc. dr. sc. Mirjana Domazet-Lošo
Karlo Pavlović

16.1.2017.

Sadržaj

1. Ukkonenov algoritam

Ukkonenov algoritam je “online” algoritam u linearnom vremenu za izradu sufiksnog stabla. Algoritam čita ulazni niz od *n* znakova, znak po znak, te proširuje implicitno sufiksno stablo sa svakim upisanim znakom. Proširenje implicitnog stabla *T(i)*, čitanjem sljedećeg *(i+1)* znaka, u implicitno stablo *T(i+1)* se odvija u 3 pravila:

Pravilo 1: Za sufiks se produljuje postojeća grana do lista (tj. proširuje se oznaka grane).

Pravilo 2: Za sufiks se u stablo dodaju novi list i grana koja vodi do lista. Napomena: ovo je jedino pravilo prema kojemu se može dodati novi list u stablo.

Pravilo 3: Ako sufiks već postoji u stablu, ne obavlja se ništa.
(izlomak iz skripte za predavanja, Bioinformatika,Mile Šikić, Mirjana Domazet-Lošo,str 65)

Nakon čitanja posljednjeg znaka i izrade implicitnog stabla *T(n)* za taj znak, implicitno stablo se pretvara u eksplicitno dodavanjem znaka “*$*” koji predstavlja kraj niza. Na taj način implicitno stablo *T(n)* prelazi u eksplicitno stablo *T(n+$).*

2. Implementacija u linearnom vremenu

Implementacija algoritma u linearnom vremenu izvodi se nadogradnjom osnovnih pravila za proširenje stabla i korištenjem sufiksnih veza.

Pravilo 1: U listove sufiksnog stabla unose se vrijednosti za početak(*p*) i kraj podniza(*k*). Proširenje implicitnog sufiksnog stabla korakom *i+1*, svim listovima se proširiva kraj podniza za 1. *(p,k)->(p,k+1)*
Korištenjem globalnog indeksa *K*, sva proširenja se implicitno izvršavaju u vremenu O(1).

Pravilo 2: Ukoliko je posljednji eksplicitno dodani list u stablu s indeksom *jPrethodni (jprethodni,K)*, kandidati za eksplicitno proširenje su samo oni sufiksi za koje vrijedi *(j,K) j>jPrethodni*. Budući da uvijek mora vrijediti *j<i+1*, broj eksplicitnih dodavanja će uvijek biti *n*.

Pravilo 3: Ako u stablu postoji *[j, i], j<=i* koji počinje s znakom *i+1*, zbog implicitnog proširenja možemo zaključiti da u stablu postoji sufiks *[j, i+1],* a prema tome i svi sufiksni koji počinju s prvim znakom koji je veći od *j.* (*p > j*). Ovo pravilo je ujedno i "show stopper" jer omogućuje prekidanje trenutnog koraka i započinjanje sljedećeg.

Razlika između *jPrethodni i i+1* može se zapisati kao preostaloProširenja (*pp*), s tim da vrijedi *pp=i+1-jPrethodni*. Na početku svakog koraka *i+1*, broj preostalih dodavanja se poveća za 1 (*pd+=1*). Nakod svakog eksplicitnog proširenja, broj preostalih proširenja se smanjuje za 1 (*pp-=1*) Korak *i+1* se izvodi sve dok se broj preostalih dodavanja ne spusti na nula (*pp=0*) ili dok se ne ispuni 3. Pravilo

Sufiksne veze:

Posljednji alat koji osigurava linearnu izvedbu algoritma su sufiksne veze. Tijekom *i+1* koraka, kada eksplicitno proširimo stablo, a to proširenje se nije dogodilo u korijenu, niti je prvo proširenje koje se odvilo u tom koraku *(i+1)*, potrebno je dodati sufiksnu vezu koja će pokazivati na novonastali čvor iz čvora na kojem se odvilo prethodno proširenje. Ovakvo dodavanje sufiksne veze omogućava da, nakon eksplicitnog dodavanja sufiksa (*j,i+1*), nije nužno tražiti sljedeći sufiks iz korijena. Potrebno je samo slijediti sufiksnu vezu (ako postoji) do čvora koji počinje znakom *j+1.*

3. Primjer algoritma

Ulazni niz: abcab$

Korak1:
pp:0 K=1 Niz:**a**
stablo:
*
-[1,K] (a)

Korak2:
pp:0 K=2 Niz:a**b**
stablo:
*
-[1,K] (ab) -pravilo 1
-[2,K] (b) -pravilo 2

Korak3:
pp:0 K=3 Niz:ab**c**
stablo:
*
-[1,K] (abc)
-[2,K] (bc)
-[3,K] (c) -pravilo 1

Korak4:
pp:1 K=4 Niz:ab**ca**
stablo:
*
-[1,K] (abca) -pravilo 3
-[2,K] (bca)
-[3,K] (ca)

Korak5:
pp:2 K=5 Niz:ab**cab**
stablo:
*
-[1,K] (abcab)
-[2,K] (bcab) -pravilo 3
-[3,K] (cab)

Korak5:
pp:2 K=6 Niz:abc**ab$**
stablo:
*
-[1,2] (ab)
- -[3,K] (cab$)
- -[6,K] ($) -pravilo 2
-[2,K] (bcab$)
-[3,K] (cab$)

Korak5.1:
pp:1 K=6 Niz:abca**b$**
stablo:
*
-[1,2] (ab)
- -[3,K] (cab$)
- -[6,K] ($)
-[2,2] (b)
- -[3,K] (cab$)
- -[6,K] ($) -pravilo 2
-[3,K] (cab$)

Korak5.2:
pp:0 K=6 Niz:abcab**$**
stablo:
*
-[1,2] (ab)
- -[3,K] (cab$)
- -[6,K] ($)
-[2,2] (b)
- -[3,K] (cab$)
- -[6,K] ($)
-[3,K] (cab$)
-[6,K] ($) -pravilo 2

END

4. Rezultati testiranja

Pokretanje programa:

Izvorni kod se nalazi u datoteci ukkonen.c
Pomoćne datoteke koje su potrebne u lokalnom direktoriju su in.txt i test.txt(opcionalno)
Program koristi sadržaj datoteke in.txt kao ulazni niz podataka za izradu sufiksnog stabla.
Program automatski generira (ako već ne postoji) datoteku out.txt u koju sprema izgrađeno sufiksno stablo.
Ukoliko datoteka test.txt postoji i nije prazna, njen sadržaj se koristi kao test primjer za koji se provjerava da li je niz sadržan kao sufiks u sufiksnom stablu.

Test primjer 1:

in.txt: ACCTA
test.txt: Ta
Console: sample was not found

out.txt:
-A
--$
--CCTA$
-C
--TA$
--CTA$
-TA$
-$

Test primjer 2:

in.txt: banana
test.txt: nana
Console: sample was found

out.txt:
-banana$
-a
--$
--na
---$
---na$
-na
--$
--na$
-$

Test primjer 3:

In.txt: zapis lambda_reference.fasta
test.txt: **prazno**
Console: **prazno**
Out.txt: stablo za lambda_reference.fasta

Parametri:
Trajanje izvođenja: 49 sec
Veličina ulazne datoteke: 48 KB
Veličina izlazne datoteke 1 149 173 KB

5. Zaključak:

Iako pruža brz način izrade sufiksnog stabla, algoritam nije praktičan za korištenje u slučaju većih ulaznih nizova. To proizlazi iz činjenice da stablo nije idealna struktura za pohranu sufiksa. Veličina izlazne datoteke je prevelika, nepraktična za uporabu. Bolja struktura za pohranu sufiksa bi bila sufiksna polja koja zahtijevaju mnogo manje prostora za pohranu izlaznih podataka, za razliku od sufiksnog stabla.

6. Literatura

skripte za predavanja, Bioinformatika, Mile Šikić, Mirjana Domazet-Lošo
https://www.fer.unizg.hr/_download/repository/bioinformatika_skripta_v1.2.pdf

Stack overflow
Ukkonen's suffix tree algorithm in plain English?
http://stackoverflow.com/questions/9452701/ukkonens-suffix-tree-algorithm-in-plain-english

SUFFIX TREES IN COMPUTATIONAL BIOLOGY
http://homepage.usask.ca/~ctl271/857/suffix_tree.shtml

On–line construction of suffix trees, Esko Ukkonen
https://www.cs.helsinki.fi/u/ukkonen/SuffixT1withFigs.pdf