SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

BIOINFORMATIKA - PROJEKT

Određivanje ograničenog najkraćeg zajedničkog nadslijeda

Ante Gojsalić 0036464577

Martin Kurtoić 0036468903

Marko Jelavić 0023093052

Zagreb, siječanj, 2016.

**Sadržaj**

[1. Opis problema 1](#_Toc441427518)

[2. Opis algoritma 3](#_Toc441427519)

[2.1 3/2-aproksimacijski algoritam 3](#_Toc441427520)

[2.2 k – ograničen najkraći zajednički nadslijed 3](#_Toc441427521)

[3. Mjerenja 8](#_Toc441427522)

[4. Zaključak 12](#_Toc441427523)

[5. Literatura 13](#_Toc441427524)

# Opis problema

Problem određivanja ograničenog najkraćeg zajedničkog nadslijeda proizlazi iz istoimene jednostavnije varijante u kojoj nema ograničenja. U problemu se radi sa slijedom znakova, te za proizvoljne slijedove treba pronaći najkraći nadslijed takav da su svi početni slijedovi sadržani u njemu, odnosno da su početni slijedovi njegovi podslijedovi. U početku vidimo da konkatenacija svih znakovnih slijedova vodi do rješenja, no ona često nije najkraći takav slijed. Također, očito je da dobiveni najkraći zajednički slijed mora biti duži ili jednake duljine kao i najduži među početnim slijedovima. Određivanje najkraćeg zajedničkog nadslijeda za 2 znakovna niza se može riješiti u polinomnom asimptotskom vremenu, dok je za proizvoljan broj stringova taj problem NP-težak, čak i za ograničenje broja znakova u nizu na 2. [5]

Na gore zadane uvjete, problem predstavljen nama postavlja još uvjeta. Tražimo ograničeni najkraći zajednički nadslijed. Ograničenja su postavljena u vidu ograničavanja dužine dobivenog nadslijeda, te minimalnog broja pojavljivanja pojedinog znaka u nadslijedu [1]. Više slijedi u ilustraciji na primjeru 1.1.

***Primjer 1.1.***

S1 = ABC

S2 = CAB

*Gdje su ograničenja po broju znakova A=2,B=2, C=1. Najkraći zajednički nadslijed bi dao sekvencu:*

S = CABC

*jer su u njemu sadržani i S1 i S2, no zbog ograničenja po minimalnom broju znakova to rješenje nije izvedivo u našem problemu. Rješenje glasi:*

S = ABCAB

*gdje su ponovno S1 i S2 sadržani u dobivenom nadslijedu, no ovaj put su zadovoljena i ograničenja.*

Također, pri konstrukciji rješenja moramo pripaziti i na ograničenje po duljini nadslijeda, ukoliko rješenje postoji, naš algoritam vrati optimalno rješenje, odnosno najkraće uz zadane uvjete.

# Opis algoritma

U radu [1] su predstavljena 2 algoritma, o kojima slijedi.

## 3/2-aproksimacijski algoritam

Nastao je idejom iz 2-aproksimacijskog algoritma. 2-aproksimacijski algoritam uzima kao rješenje konkatenaciju dvaju zadanih znakovnih slijedova, te ukoliko su ograničenja nad brojem znakova ostvariva za dva zadana znakovna slijeda, ona će ujedno biti i ispunjena, no neće dati optimalno rješenje u smislu najkraće duljine. [1]

3/2-aproksimacijski algoritam radi na sličnom tragu. Prvo se u polinomskom vremenu izračuna najkraći zajednički slijed dvaju znakovnih nizova te se u konačnici dodaju onaj broj znakova koji fali da bi ograničenja bila ispunjena. Dobiveni nadslijed po definiciji zadovoljava ograničenja, no on je također i nadslijed od oba zadana znakovna niza jer smo u početku izračunali najkraći zajednički nadslijed bez ograničenja. [1]

## k – ograničen najkraći zajednički nadslijed

Ovaj algoritam temelji se na stablju za pretraživanje ograničenom dubinom. Opisat ćemo algoritam kroz jednostavan primjer.

*Primjer 2.2.1*

*S1=ABAC*

*S2=BABC*

*T\_L=(A,1,B,2,C,1) – ograničenja*

*k=6*

*Algoritam prvo provjerava da li su ograničenja ostvariva za zadane dvije sekvence. Ukoliko jesu, u root element stabla se postavlja najdesniji indeks od obe sekvence (u ovom slučaju 3), duljina izgrađene sekvence dosad (u rootu 0), međurješenje (u rootu prazan skup), te postavljena ograničenja.*

*U prvom prolazu vidimo da se sekvence poklapaju na najdesnijem članu, stoga njega dodajemo kao privremeno rješenje, indekse pomaknute za jedan u lijevo, duljinu međurješenja povećavamo za 1 te smanjujemo ograničenje za dodan znak iz alfabeta u ovom slučaju C. „Otac” ovom čvoru je root element.*

*U drugom prolazu, ovisno po kojem slučaju se zadani algoritam aktivira dobijemo različite čvorove stabla. Konkretno ćemo napisati za slučaj koji se realizira, a to je u radu slučaj broj 3. Vidimo da se na poziciji 2,2 ne preklapaju sekvence, stoga dodamo oba ta znaka kao međurješenje u tom koraku, pomaknemo indekse za 1 ulijevo te umanjimo ograničenja za zadane znakove. Kao roditelja mu postavimo čvor iz kojeg smo došli do njega, odnosno čvor prethodnog koraka. Ovo je bio kratak opis, slijedi i slikovitija demonstracija na gore zadanom primjeru.*

*Nulti prolaz*

*Provjera uvjeta. Dodavanje roota u stablo*

*Root=(indeks1=3, indeks2=3, duljina rješenja=0, međurješenje=””, A=1,B=2,C=1, parent=NULL)*

*Prvi prolaz*

*Čvor11=(indeks1=2,indeks2=2,duljina rješenja=1, međurješenje=”C”,A=1,B=2,C=0, parent=ROOT)*

*Drugi prolaz*

*Čvor21=(indeks1=1,indeks2=0,duljina rješenja=3,međurješenje=”AB”,A=0,B=1,C=0,parent=Čvor11)*

*Čvor22=(indeks1=0,indeks2=1,duljina rješenja = 3, međurješenje=”BA”,A=0,B=1,C=0,parent=Čvor11)*

*Čvor 23=(indeks1=1,indeks2=1,duljina rješenja=3, međurješenje=”BA”, A=0,B=1,C=0,parent=Čvor11)*

*Treći prolaz*

*Razvoj po trećem*

*Čvor31=(indeks1=0,indeks2=-1, duljina rješenja=5, međurješenje=”BA”,A=-1,B=0,C=0, parent=Čvor23)*

*Čvor 32=(indeks1=-1,indeks2=0,duljina rješenja=5, međurješenje=”AB”,A=-1,B=0,C=0, parent=Čvor23)*

*Čvor33=(indeks1=0,indeks2=0, duljina rješenja=5, međurješenje=”AB”,A=-1,B=0,C=0, parent=Čvor23)*

*Četvrti prolaz*

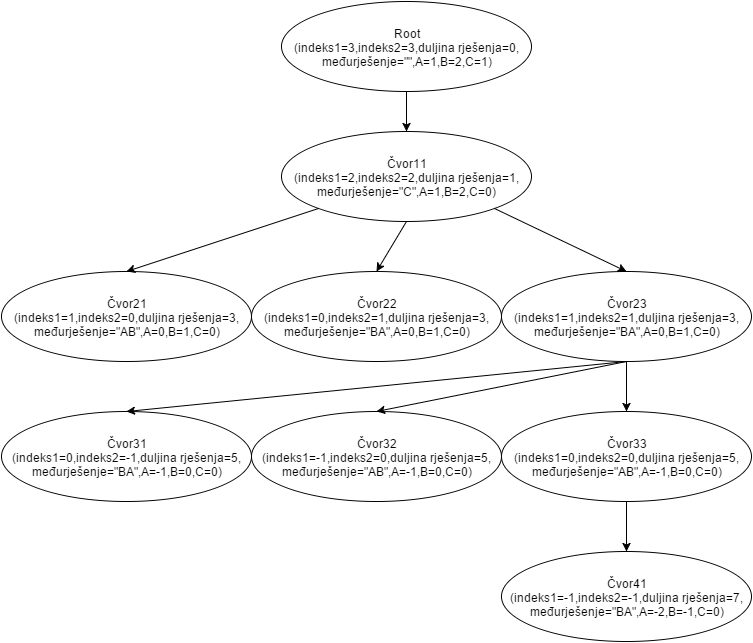
*Čvor41=(indeks1=-1,indeks2=-1,duljina rješenja=7,međurješenje=”BA”,A=-2,B=-1,C=0, parent=Čvor33).*

*Pošto je zadani k=6, nama ovo rješenje otpada te se onda granamo po Čvoru32.*

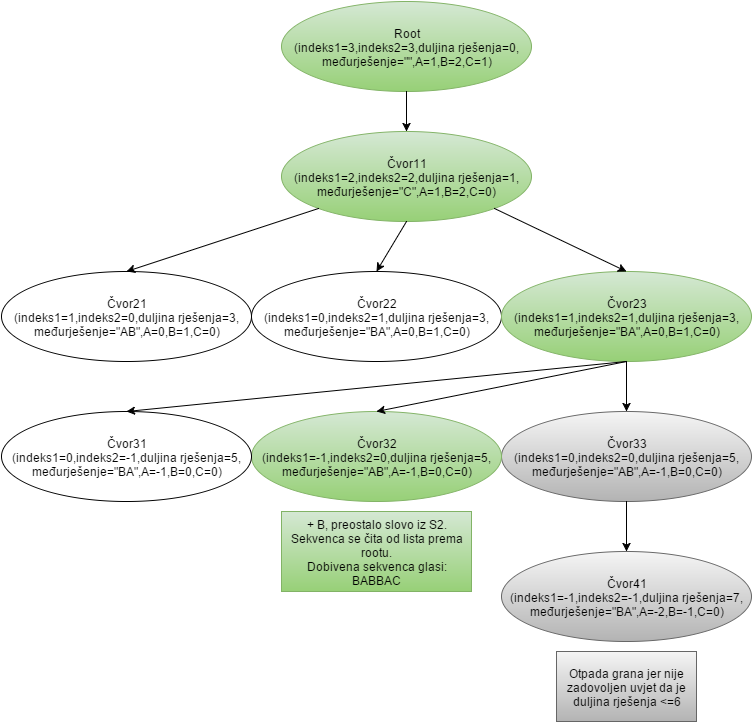
*Peti prolaz*

*Vraćamo se u stablo na Čvor32. Jedan od uvjeta je zadovoljen (jedan od indeksa=-1). Zaustavljamo grananje. Nadodajemo preostale elemente iz sekvence koja nije došla do kraja (ovdje je to S2 i preostali element je B). Provjeravamo uvjet duljine, zadovoljen je (duljina 6). Provjeravamo uvjete po znakovima (svi su 0 ili -), zadovoljeni. Rješenje je prihvatljivo te ispisujemo preostale znakove S2, te međurješenja Čvora32 i svih njegovih roditelja.*

*Konačna sekvenca glasi S=BABBAC. Ilustracija stabla slijedi (Slika 2.2.1)*

**

Slika 2.2.1: Ilustracija gradnje stabla na primjeru

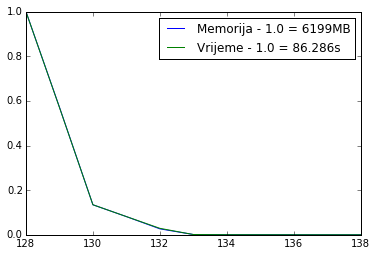


Slika 2.2.2: Konačno rješenje u obliku stabla

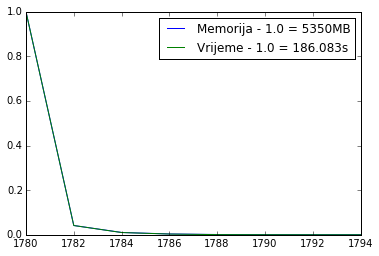
# Mjerenja

Sva mjerenja su provedena na računalu navedenom pod [2] u poglavlju Literatura, uz napomenu da je korištena verzija s i3 procesorom i 8 GB radne memorije, operacijski sustav Ubuntu 14.04 LTS sa biolinux paketom. Mjerenja su provedena u komandnoj liniji instrukcijom zadanom od strane profesora.

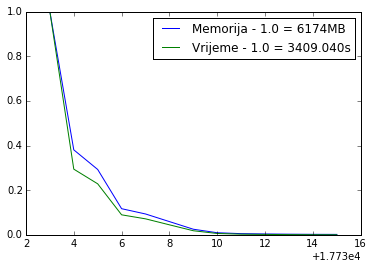
Dobiveni rezultati prikazani su u ovisnosti o k-u (ograničenju dužine) za različite duljine sekvenci:



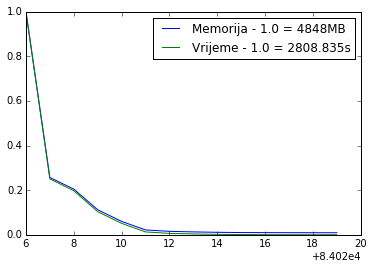
Slika 3.1: Za dvije sekvence duljine 100, potrošnja memorije za krajnji k iznosi 2 MB, a vrijeme izvođenja 0.004s



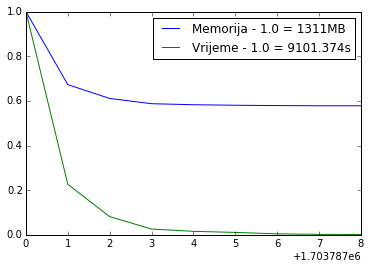
Slika 3.2: Za dvije sekvence duljine 1000, potrošnja memorije za krajnji k iznosi 3MB, a vrijeme izvođenja 0.019s



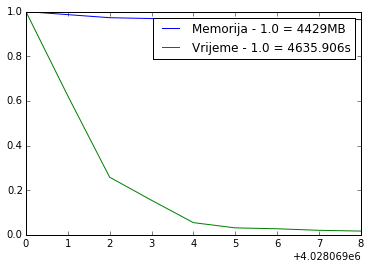
Slika 3.3: Za dvije sekvence duljine 10 000, potrošnja memorije za krajnji k iznosi 10MB, a vrijeme izvođenja 0.109s



Slika 3.4: Za dvije sekvence duljine 50 000, potrošnja memorije za krajnji k iznosi 45MB, a vrijeme izvođenja 0.532s



Slika 3.5: Za dvije sekvence duljine 1 000 000, potrošnja memorije za krajnji k iznosi 758MB, a vrijeme izvođenja 10.050s



Slika 3.6: Za prepolovljene sekvence navedene u literaturi [3] i [4], potrošnja memorije za krajnji k iznosi 4271MB, a vrijeme izvođenja 77.131s

# Zaključak

Određivanje najkraćeg zajedničkog nadslijeda je pronašlo svoju uporabu u mnogim znanstvenim granama, od planiranja o umjetnoj inteligenciji do bioinformatike za usporedbe genoma i slično. U umjetnoj inteligenciji pri planiranju važno je uočiti interakciju među dijelovima u shemi globalnog planiranja dok kod bioinformatike genomi se prikazuju kao znakovni nizovi te se zatim uspoređuju za određivanje sličnost i razlika. Pošto je redoslijed znakova bitan, mjera poput najkraćeg zajedničkog nadslijeda postaje vrlo korisna. [1]

Posljednjih godina se pojavilo mnogo različitih pristupa sa postavljanjem ograničenja nad najkraćim zajedničkim nadslijedom, gdje ta ograničenja također imaju svrhu. Kod planiranja to može biti ograničenost resursima dok u bioinformatici to može biti dobivanje originalne kopije gena iz koje su proizašle sve ostale kopije kroz duplikaciju. [1]

U ovom projektu su predstavljena i ostvarena rješenja za probleme u radu [1] pod poglavljem Literatura. U radu [1] su autori detaljnije analizirali i dokazivali složenosti algoritama, dok je na nama bilo da ih tehnički ostvarimo, što i jesmo, te prezentiramo naša mjerenja, što također jesmo. Algoritmi su korektni u smislu da, uz zadana ograničenja, uvijek pronađu rješenje ukoliko postoji.

# Literatura

[1] Riccardo Dondi, The constrained shortest common supersequence problem, Journal of Discrete Algorithms, Volume 21, July 2013, Pages 11-17, ISSN 1570-8667, http://dx.doi.org/10.1016/j.jda.2013.03.004. Whttp://www.sciencedirect.com/science/article/pii/S1570866713000208

[2] https://www.asus.com/Notebooks/N53Jg/specifications/

[3] <http://www.ncbi.nlm.nih.gov/nuccore/26111730?report=fasta>

[4] <http://www.ncbi.nlm.nih.gov/nuccore/56384585?report=fasta>

[5] <https://en.wikipedia.org/wiki/Shortest_common_supersequence_problem>