SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Navarrov algoritam za približno uspoređivanje teksta

Petra Mikulić (0036490588)

Ivan Šego (0036495633)

Đesika Kolarić(0006036358)

Voditelj: Krešimir Križanović

Zagreb, siječanj, 2020.

**Sadržaj**

[1. Uvod 1](#_Toc159987575)

1.1. Klasični algoritam i preklapanje uzorka

1.2. Navarrov algoritam

1.3. Zadatak

[2. Rezultati 2](#_Toc159987576)

[3. Zaključak 3](#_Toc159987577)

[4. Literatura 4](#_Toc159987578)

[5. Sažetak 5](#_Toc159987579)

# Uvod

Ideja približnog uspoređivanje teksta (*approximate pattern matching)* predstavljena je (i riješena) još krajem prošlog stoljeća u radu [1]. Algoritam se može primijeniti za rješavanje različitih problema – od pretraživanja teksta do problema računalne biologije.

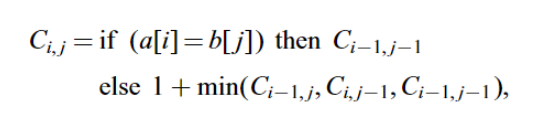
**1.1. Klasični algoritam i preklapanje uzorka**

Problem je definiran na sljedeći način: pretpostavimo da imamo *text* duljine *n* i *uzorak* (odnosno kratki tekst) duljine *m.* Pretpostavljeni algoritam mora pronaći sve dijelove *teksta* čija je udaljenost uređivanja od *uzorka* ≤ *k,* pri čemu je *k* ukupan broj mogućih promjena koje možemo primijeniti na tekstu ili uzorku. Budući da je duljina uzorka *m,* maksimalan broj dozvoljenih pogrešaka je *k < m.*

Udaljenost uređivanja zapravo je minimalan broj promjena koje moramo napraviti na *uzorku* ili *tekstu* da bi se oni potpuno preklapali. Pri tome, znakove možemo ili brisati, ili mijenjati, ili ubacivati.

Dakle, algoritam nam daje sve pozicije *j* u *tekstu* koje predstavljaju sufiks *teksta*[1...*j*] za koji je udaljenost uređivanja od *uzorka* ≤ *k.*

Jedna od opcija je da *tekst* i *uzorak* predstavimo kao matricu dimenzija *m\*n*. Klasično rješenje tako postavljenog problema tada je dinamičko programiranje u kojem se kroz matricu krećemo stupac po stupac i za svaki par sufiksa određujemo njihovu udaljenost uređivanja. Računamo udaljenost uređivanja za svaku promjenu koju možemo napraviti (brisanje, zamjena, ubacivanje) i odaberemo najmanju [3].



Slika 1.1. Pseudokod računanja udaljenosti uređivanja

Budući da je vremenska složenost takvog rješenja O(*mn*), takav nam je algoritam prespor za rješavanje tekstova velikih duljina, poput genoma.

Model preklapanja uzorka predlaže da umjesto matrice napravimo graf koji se sastoji od *N* čvorova (*nodes)* i *E* veza (*edges*). Uzorak je i dalje jednostavni string duljine *m,* a tekst je prikazan unutar čvorova grafa. Svaki čvor sadrži kratki string, a veze predstavljaju mogući tekst koji može slijediti nakon stringa u našem čvoru [2].

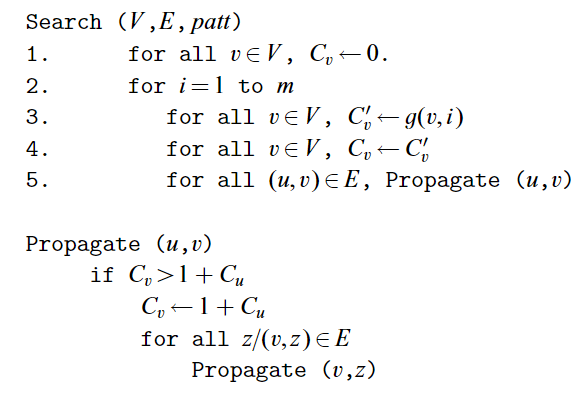
**1.2. Navarrov algoritam**

U Navarrovom algoritmu, iteriramo po grafu koji sadrži *N* čvorova i *E* veza. Imamo *m* iteracija*.* Na kraju i-te iteracije, za svaki smo čvor *v* izračunali najmanju udaljenost uređivanja između *uzorka*[1...*i*] i bilo kojeg sufiksa *teksta* koji u grafu završava na čvoru *v.*

U svakoj iteraciji moramo pamtiti stanje, što pohranimo u *Cv.* Svaka iteracija pamti nove vrijednosti *Cv'.* Vremenska složenost takvog algoritma je *O(m(n + e)).*

U algoritam dodajemo funkciju Propagate koja nam kod cikličkih grafova osigurava da se, ako se promijeni vrijednost u jednom čvoru, promjena propagira u krug [3].

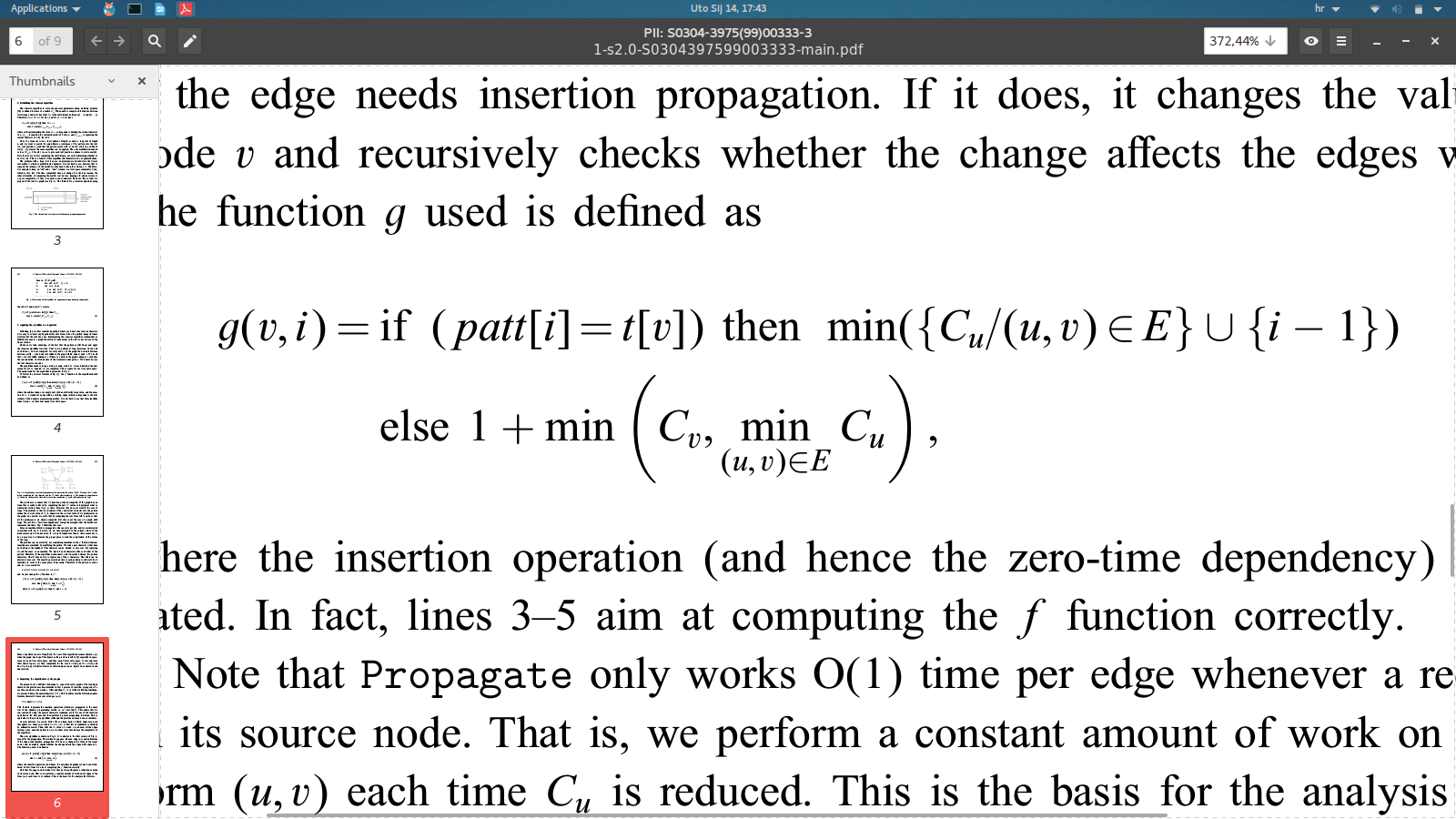
Pseudokod Navarrovog algoritma je:



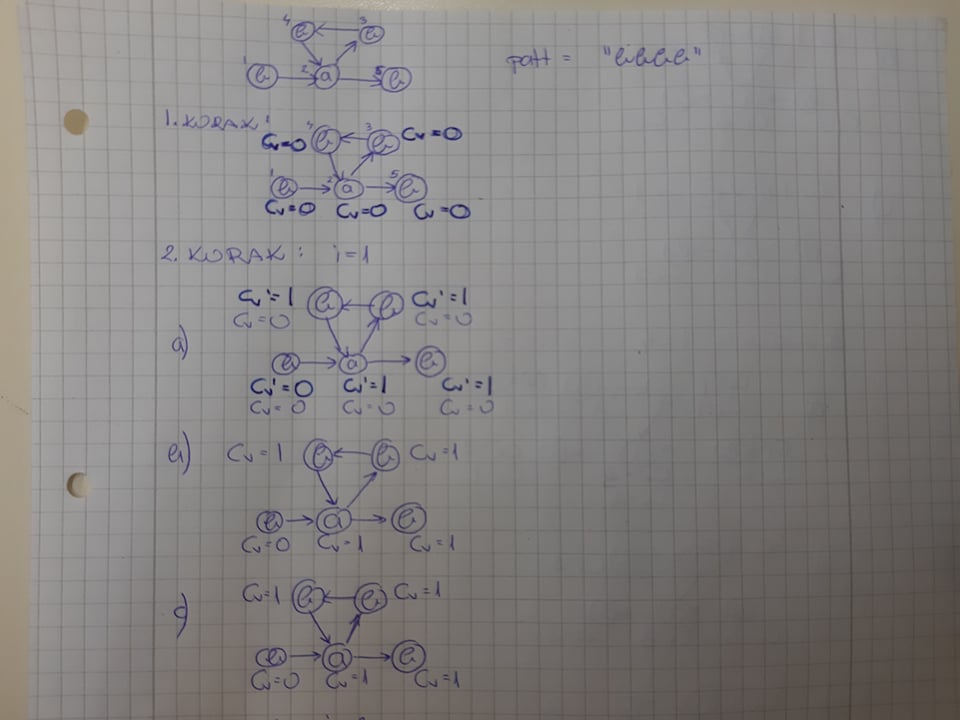
Slika 1.2. Pseudokod Navarrovog algoritma

Na primjeru i vizualizaciji algoritma (Slika1.4.), korak 2 odgovara koraku 1 na vizualizaciji, korak 3 u pseudokodu odgovara koracima 2.a), 3.a) i 4.a); korak 4 pseudokoda odgovara koracima 2.b), 3.b) i 4.b), dok korak 5 (odnosno funkcija propagate) odgovara koracima 2.c), 3.c), 4.c) i 4.d) vizualizacije.

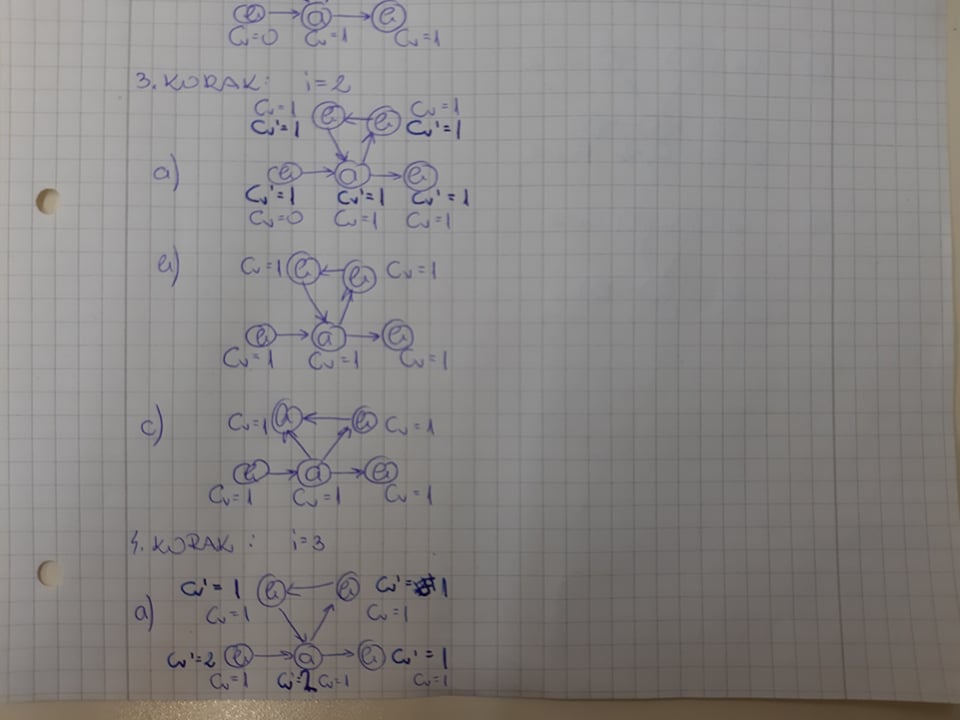
Pseudokod funkcije g prikazan je na slici niže. Funkcija g vraća minimalni edit distance za pojedini čvor.



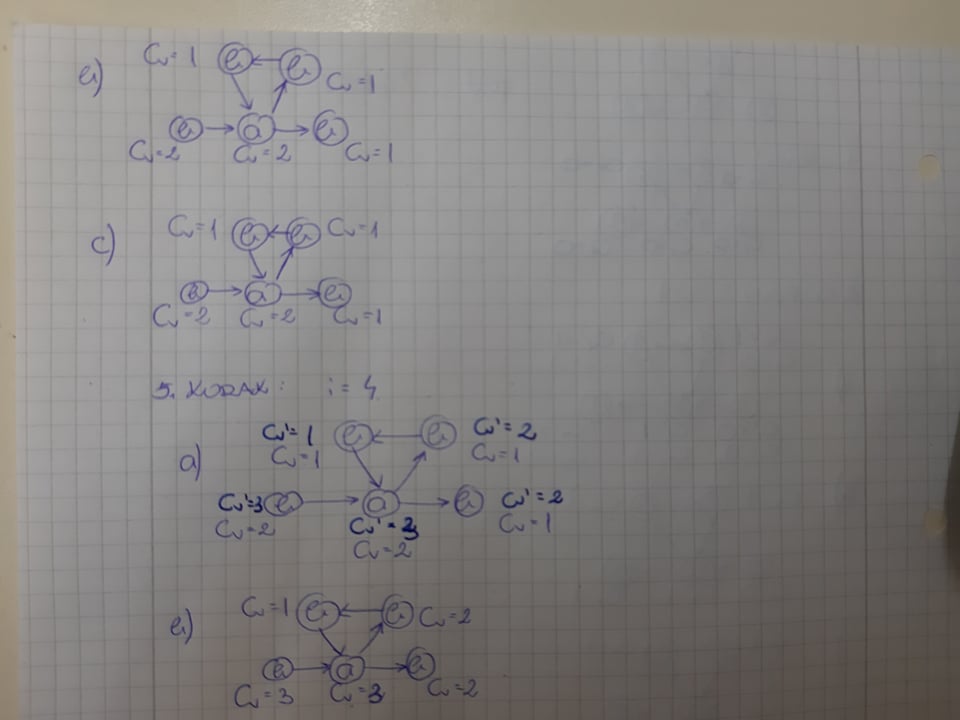
Slika 1.3. Funkcija g



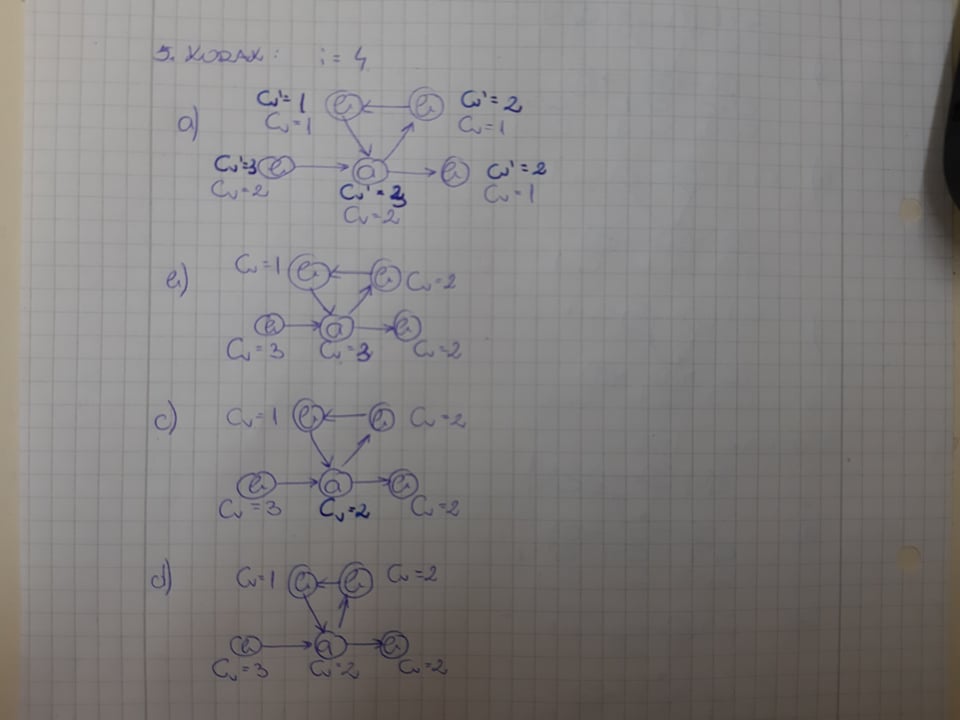
Slika 1.4. a) Vizualizacija koraka 1. i 2. algoritma



Slika 1.4. b) Vizualizacija koraka 3. i 4.a) algoritma



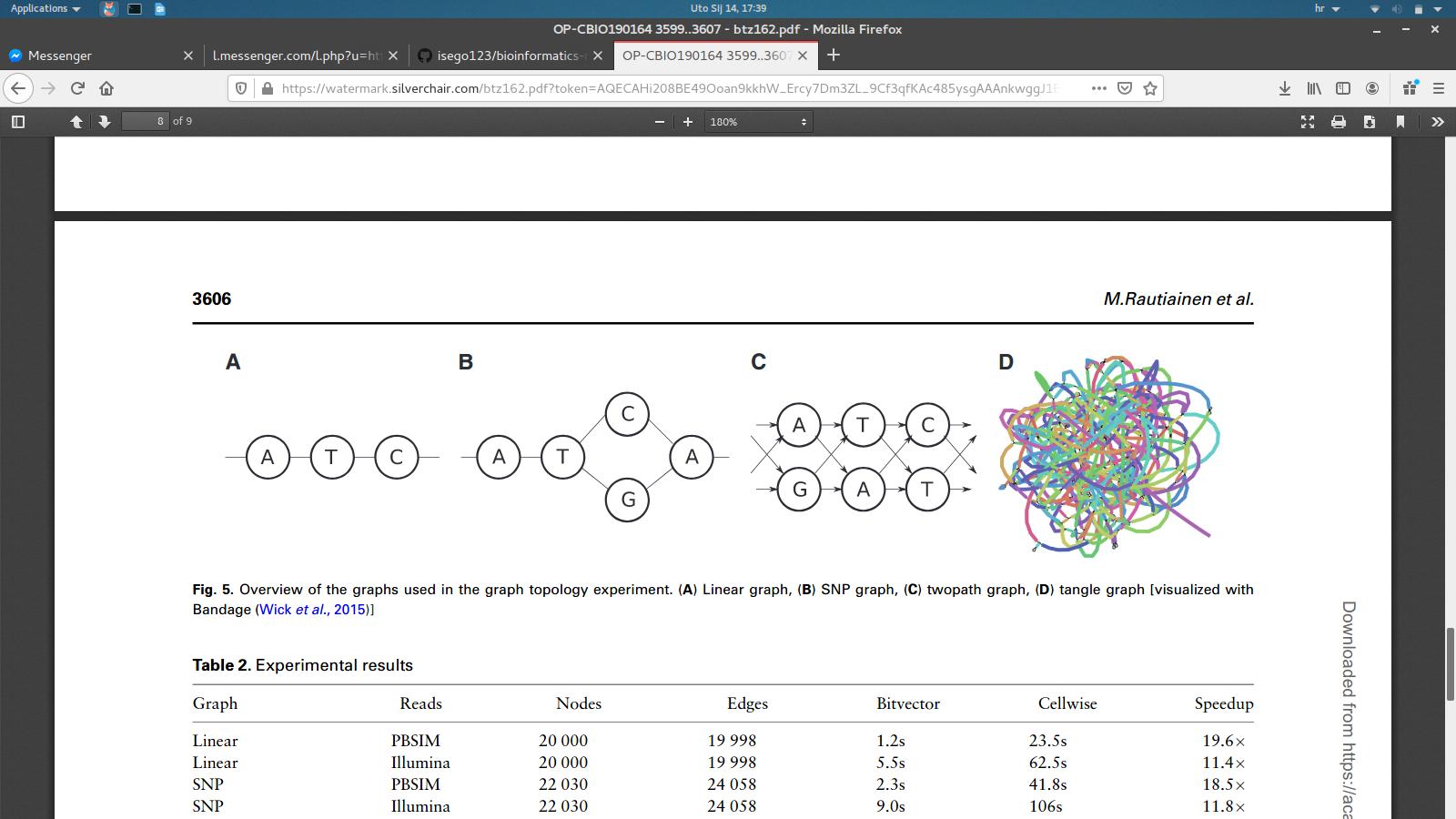
Slika 1.4.c) Vizualizacija koraka 4.b) i c) i koraka 5. a) i b)



Slika 1.4.d) Vizualizacija koraka 5. algoritma

**1.3. Zadatak**

Naš je zadatak bio implementirati Navarrov algoritam za približno uspoređivanje teksta i usporediti ga s bit paralelnim algoritmom poravnanja sekvence u graf (*bit – parallel sequence to graph alignment algorithm).* Algoritme uspoređujemo na 4 vrste graf topologija opisanih u radu [4]. Uspoređujemo na linearnom grafu, SNP grafu, *tangle* grafu i *twopath* grafu prikazanim na slici ispod.



Slika 1.5. Prikaz linearnog (A), SNP (2), twopath (3) i tangle (4) grafa

Usporedili smo točnost, vrijeme izvođenja i utrošak memorije oba algoritma. Rezultati vremena i memorije su prikazani tablično.

# Rezultati

Tablica 1. Usporedba vremena za Navarro i bitvector algoritam

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Time / s** | | |
|  | **Navarro** | **Bitvector** | **Ratio** |
| **Linear** | 438.34 | 12.59 | 34.82 |
| **SNP** | 573.85 | 22.34 | 25.69 |
| **Twopath** | 1223.05 | 89.04 | 13.74 |
| **Tangle** | 478.85 | 64.31 | 7.45 |

Tablica 2. Usporedba memorije za Navarro i bitvector algoritam

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Memory / GB** | | |
|  | **Navarro** | **Bitvector** | **Ratio** |
| **Linear** | 0.48 | 0.38 | 1.26 |
| **SNP** | 0.47 | 0.37 | 1.27 |
| **Twopath** | 0.50 | 0.38 | 1.32 |
| **Tangle** | 0.48 | 0.37 | 1.3 |

# Zaključak

Iz rezultata prikazanih u tablici vidimo da je Navarrov algoritam i sporiji i memorijski zahtjevniji od bit-vektor algoritma. Najveća je razlika u vremenu izvođenja Navarrovog algoritma na linearnom grafu i sve manja i manja u SNP, twopath i tangle.

Memorijski je najzahtjevnije izvođenje Navarrovog algoritma na twopath grafu, iako neznačajno više nego u ostala 3 grafa.

Vremenska potrošnja ipak je do 50 puta sporija, što je unutar zahtjevanog ograničenja.

Možemo zaključiti da je za velike genome pogodnije koristiti bit-vektor algoritam.

# Sažetak

Cilj ovog projekta bio je usporediti vrijeme izvođenja i memorijsku složenost Navarrovog algoritma za približno uspoređivanje teksta s bit-paralelnim algoritmom za poravnanje teksta u graf, na 4 vrste graf topologije – linearni graf, SNP graf, tangle i twopath. Oba algoritma predstavljaju koristan alat za rješavanje bioinformatičkih problema.

Na temelju dobivenih rezultata, zaključujemo da je za velike genome pogodnije koristiti bit-vektor algoritam.

# Literatura

[1] A. Amir, M. Lewenstein, N. Lewenstein, Pattern matching in hypertext, Lecture Notes in Computer Science, Springer, Vol. 1272, 1997, 160-173.

[2] G. Navarro, Improved approximate pattern matching on hypertext, Theoretical Computer Science, Vol. 273, 1999, 455-463.

[3] K. Park, D. K. Kim, String matching in hypertext, Lecture Notes in Computer Science, Springer, Vol. 937, 2005, 318-329.

[4] M. Rautiainen, V. Makinen, T. Marschall, Bit-parallel sequence-to-graph alignment, Bioinformatics, Vol. 35, 2019, 3599 – 3607.