SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**DOKUMENTACIJA**

**Navarrov algoritam za približno uspoređivanje teksta**

*Pavao Jerebić, Toni Jurjević, Lucija Šošić*

Voditelj: *Krešimir Križanović*

Zagreb, siječanj, 2020.

**Sadržaj**

[1.](#_gjdgxs) Uvod 2

[2.](#_30j0zll) Navarrov algoritam3

[3.](#_1fob9te) Rezultati testiranja 5

[4.](#_2et92p0) Zaključak 6

[5.](#_3znysh7) Literatura 7

1. OPISATI ZADANI ALGORITAM

2. OBJASNITI KORAKE NA JEDNOSTAVNOM PRIMJERU

3. PRIKAZATI REZULTATE TESTIRANJA: VRIJEME IZVOĐENJA I KOLIČINA ZAUZETE MEMORIJE

4. ZAKLJUČAK

5. LITERATURA

# **Uvod**

Približno podudaranje nizova je tehnika pronalaženja nizova koji se ne preklapaju u potpunosti, već parcijalno. Primjenjeno na hipertekst, ono pronalazi svoju motivaciju u strukturi World Wide Web-a koja modelira složene procese. Grafovi se koriste u reprezentaciji sekvenci, tako da svaki put u grafu čini jedan slijed [1]. Česta primjena približnog podudaranja nizova je kod provjere pravopisa te kod uspoređivanja nukleotidnih sekvenci u bioinformatici [2].

Sličnost podudaranja mjeri se brojem operacija potrebnih za transformaciju jednog niza u drugi. Levenshteinova udaljenost uređivanja računa minimalan težinski broj zamjena, umetanja i brisanja. Problem približnog podudaranja niza može se definirati kao: s obzirom na uzorak P = p1p2...pm i tekst T = t1t2...tn pronađi podniz Tj’,j = tj’...tj

unutar T, takav da od svih podnizova unutar T, on ima najmanju udaljenost uređivanja za transformaciju u uzorak P [3].

Navarrov algoritam koristi se u bioinformatici kada se žele poravnati proteinski ili nukleotidni nizovi. Zasniva se na dinamičkom programiranju koje dijeli veći problem, u ovom slučaju cijele sekvence, u slijedove manjih problema, a njihova rješenja koristi kako bi našao optimalno rješenje za veći problem.

Algoritam istodobno pamti samo dva retka matrice udaljenosti uređivanja, a kroz svaki redak prolazi dva puta. Prvim prolazom računa vrijednosti udaljenosti trenutnog retka na temelju vrijednosti prethodnog retka, a drugim prolazom propagira udaljenosti za vrijednosti u istom retku pomoću pretraživanja u dubinu [4].

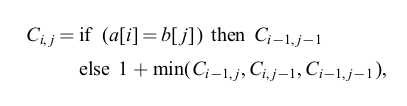
# **Navarrov algoritam**

**Opis algoritma**

U klasičnom pristupu, prilikom izračunavanja udaljenosti uređivanja između nizova a i b, matrica Ci,j predstavlja udaljenost uređivanja između a[1...i] te b[1….j].

Elementi prvog stupca postavljaju se na Ci,0 = i, za i>1, dok se elementi početnog retka postavljaju se na nulu, C0,j = 0 za svaki j, kako bi se svakoj poziciji u tekstu pružila prilika za započeti podudaranje.

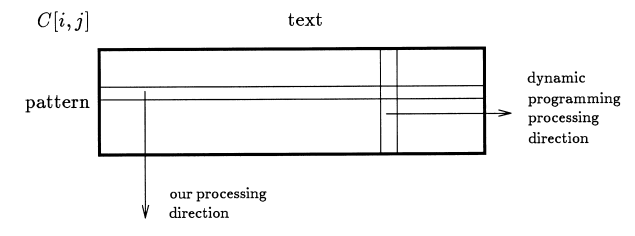
Preostali elementi matrice izračunavaju se preko formule (1):



(1)

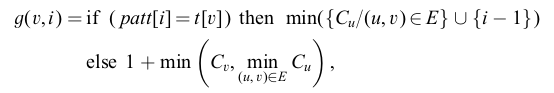
gdje Ci-1,j odgovara brisanju trenutnog karaktera niza a, Ci,j-1 odgovara umetanju trenutnog karaktera niza b u niz a, te Ci-1,j-1 predstavlja zamjenu trenutnih karaktera nizova a i b.

Problem nastaje prilikom pohranjivanja velikog teksta. Dovoljno je pohranjivati samo redak po redak matrice i tako sačuvati složenost O(n). Kretanje po matrici prikazano je na slici 1.



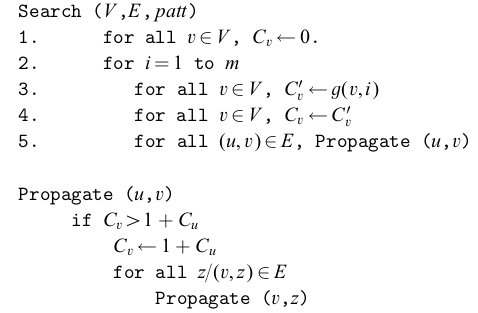
Slika 1. Prikaz kretanja po matrici uređivanja Ci,j

Navarrov algoritam uvodi izračunavanje elementa ćelije novog retka iz prethodnog slijedi preko formule (2):

(2)

gdje i-1 predstavlja čvorove grafa bez ulaznih bridova, odnosno prvi stupac matrice.

Pseudokod za približno podudaranje uzorka unutar teksta predoćenim grafom prikazan je na slici 2.



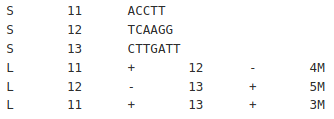
Slika 2. Pseudokod algoritma za približno podudaranje niza s hipertekstom

*Propagate* funkcija određuje zahtjeva li brid (u,v) propagaciju vrijednosti. U slučaju da zahtjeva, mijenja vrijednost čvora v te rekurzivno provjerava utjecaj nove vrijednosti na rubove koji započinju u v [2].

Navarrov algoritam time postiže vremensku složenost O(m(n+e)) te prostornu složenost O(n), dok je vremenska složenost *Propagate* funkcije jednaka O(1).

**Primjer algoritma**

Tekst u kojem se pretražuje približno poravnanje s uzorkom dan je u GFA formatu prikazanom na slici 3. To je tekst format koji opisuje skup sekvenci i njihova preklapanja. U sljedećem zapisu, linije sa sekvencama označene su sa S, dok su linije s preklapanjima dviju sekvenci označene s L.



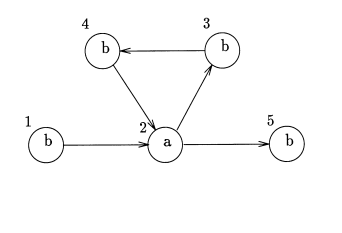
Slika 3. Primjer ulaznog teksta

Uzorak je sekvenca zapisana u FastQ formatu, prikazanom na slici 4. To je tekst format koji pohranjuje biološku sekvencu i njenu kvalitetu, a koji su enkodirani s ASCII karakterima radi sažetosti zapisa. Format koristi 4 linije po sekvenci:  
 1. linija koja sadrži oznaku “@” za kojom slijedi identifikator sekvence  
 2. linija koja sadrži znakove sekvence  
 3. linija koja sadrži oznaku “+” za kojom po izboru slijedi identifikator i opis  
 4. linija koja sadrži oznake kvalitete sekvence sadržane u drugoj liniji



Slika 4. Primjer ulaznog uzorka koji se pretražuje u tekstu

Korake algoritma objasnit ćemo na primjeru sa slike 5. Sekvenca koja predstavlja naš uzorak je “bbbb”. Izračunavanje redaka matrice udaljenosti uređivanja Cij vršimo prateći pseudokod sa slike 2.



Slika 5. Primjer jednostavnog cikličkog grafa

Prilikom prolaza kroz retke matrice, vrijednosti računamo preko formule (2). Za svaki čvor u našem grafu, inicijalno postavimo vrijednosti Cv na nulu.

Prvo uspoređujemo prvi znak “b” sekvence s prvim čvorom “b”: podudaraju se pa u C’v zapisujemo 0. Zatim uspoređujemo “b” s drugim čvorom “a”: ne podudaraju se, pa u C’v zapisujemo 1. Iterativnim postupkom, nakon obrade prvog znaka sekvence “b”, redak glasi [01000]. U trenutnom koraku, funkcija propagacije ne osvježava dobivene vrijednosti. Nakon usporedbe drugog znaka “b” sa čvorovima grafa, osvježavamo C’v vrijednosti na [11101], a nakon usporedbe s trećim znakom “b”, osvježavamo ponovno C’v vrijednosti na [21111]. Ponovno, ni u drugom ni u trećem koraku se ne propagiraju vrijednosti zbog ciklusa. Posljedni korak usporedbe zadnjeg znaka “b” daje rezultantne C’v oznake [32111].

# Rezultati testiranja

Vremena izvođenja dobivena testiranjem algoritma, uključujući backtrack, na 5 vrsta graf topologija prikazani su u tablici 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| topologija | vrijeme izvođenja s backtrack (s) | vrijeme izvođenja bez backtrack (s) | zauzeće memorije s backtracking (GB) | zauzeće memorije bez backtracking (GB) |
| linear | 32 | 25 | 1 | 0.012 |
| one\_char | 42 | 33 | 1 | 0.023 |
| snp | 50 | 39 | 1.3 | 0.030 |
| two\_path | 207 | 178 | 2.3 | 0.074 |
| tangle | 270 | 210 | 3.1 | 0.093 |

Tablica 1. Rezultati testiranja vlastite implementacije

Vrijeme izvođenja vlastitog algoritma uspoređujemo s rezultatima dobivenim “Bit parallel sequence-to-graph alignment” algoritmom (opisan u radu <https://academic.oup.com/bioinformatics/article/35/19/3599/5372677>), prikazano u tablici 2.

|  |  |  |  |
| --- | --- | --- | --- |
| topologija | bitvector | cellbycell | navarro |
| linear | 2563729us | 100056612us | 32049138us |
| one\_char | 4811866us | 166123730us | 42128587us |
| snp | 4286745us | 144580777us | 50368619us |
| two\_path | 29940661us | 420768577us | 207352993us |
| tangle | 14152466us | 144432665us | 270203062us |

Tablica 2.Vremena izvođenja algoritama

Ako za referentna vremena izvođenja uzmemo ona postigna “Bit parallel sequence-to-graph alignment” algoritmom, možemo vidjeti da je naša implementacija nad svim topologijama grafa sporija. Međutim, u usporedbi s “cell by cell” algoritmom, naša implementacija ima bolja vremena izvođenja u acikličkim topologijama. Omjeri su prikazani u tablici 3.

|  |  |  |
| --- | --- | --- |
| graf | ratio cellbycell | ratio navarro |
| linear | 39.0278 | 12.5009 |
| one\_char | 34.5238 | 8.7551 |
| snp | 33.7274 | 11.7498 |
| two\_path | 14.0534 | 6.9254 |
| tangle | 10.2055 | 19.0922 |

Tablica 3. Omjeri vremena izvođenja

# Zaključak

Navarrov algoritam pristupa problemu približnog podudaranja na inovativan način uz uštedu memorijskog prostora. Vremenska složenost algoritma iznosi O(m(n+e)) te prostorna O(n). Uz backtracking, prostorna složenost se povećava na O(mᐧn).

# **Literatura**

[1] Mikko Rautiainen, Veli Makinen, Tobias Marschall; *Bioinformatics*, Volume 35, Issue 19, 1 October 2019; Bitt-paralleel sequence-to-graph alignment <https://academic.oup.com/bioinformatics/article/35/19/3599/5372677>

[2] Gonzalo Navarro; *Theoretical Computer Science*, Volume 237, Issues1-2, 28 April 2000; <https://www.sciencedirect.com/science/article/pii/S0304397599003333>

[3] Approximate string matching, Wikipedia. Preuzeto 13. siječnja 2020. <https://en.wikipedia.org/wiki/Approximate_string_matching>

[4] Needleman-Wunsch algorithm, Wikipedia. Preuzeto 13. siječnja 2020. <https://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm>

[5] The GFA Format Specification. Preuzeto 13. siječnja 2020. <http://gfa-spec.github.io/GFA-spec/GFA1.html>

[6] FASTQ format, Wikipedia. Preuzeto 13. siječnja 2020. <https://en.wikipedia.org/wiki/FASTQ_format>