Fakultet elektrotehnike i računarstva

Bioinformatika

Projektni izvještaj

**Računanje udaljenosti uređivanja algoritmom**

**4 Russians**

Borna Ivanković, Josipa Popović, Iva Topolovac

Nastavnik: Izv. prof. dr. sc. Mile Šikić

Zagreb, siječanj 2018.

Sadržaj

[1. Uvod 3](#_Toc504053167)

[2. Opis rada algoritma 4 Russians 4](#_Toc504053168)

[2.1 Osnovna ideja 4](#_Toc504053169)

[2.2 Smanjenje broja mogućih vrijednosti o kojima ovisi blok funkcija 6](#_Toc504053170)

[2.3 Konačan algoritam 7](#_Toc504053171)

[3. Provedba algoritma na primjeru 9](#_Toc504053172)

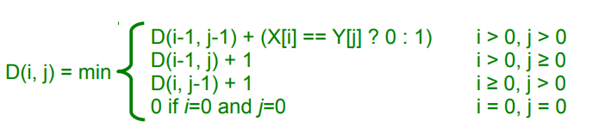
[4. Rezultati 11](#_Toc504053173)

[5. Zaključak 12](#_Toc504053174)

[6. Literatura 13](#_Toc504053175)

# Uvod

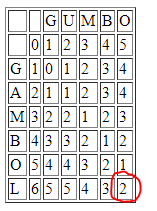
Koristeći dinamičko programiranje koje se temelji na jednostavnoj rekurziji:



Relacija 1: Relacija za računanje udaljenosti između nizova [1]

Moguće je izračunati udaljenost između dva niza znakova X[1..n] i Y[1..m] s vremenskom i prostornom složenošću O(nm). [1]

Npr.:



Slika 1: Primjer izračuna Levenshteinove udaljenosti za nizove „GUMBO“ i „GAMBOL“ [2]

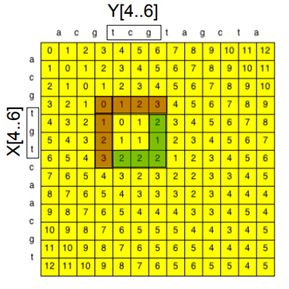
Ispostavilo se da je moguće izračunati udaljenost dvaju nizova i sa manjom složenošću O(n2/logn) uz uvjet n≥m, i to koristeći algoritam “4 Russians”[[1]](#footnote-1). Ovaj algoritam pojavljuje se 1970. kada se primjenjuje za množenje Boolean matrica (Arlazarov, Dinic, Kronrod, Faradzev) te nešto kasnije nalazi primjenu u računanju udaljenosti između nizova znakova(Masek, Paterson).

# Opis rada algoritma 4 Russians

## Osnovna ideja

Da bismo bolje razumjeli algoritam uvodimo pojam t-blok. T-blok je blok veličine t x t.

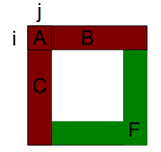
Npr. Na slici je u tzv. D-tablici (tablica udaljenosti nizova ispunjena koristeći Levenstheinov algoritam) označen t-blok, za t=3 na poziciji (3,3).



Slika 2: T-blok u D-tablici [1]

Slika dolje prikazuje da je za t-blok na nekoj poziciji (i,j) izlaz F funkcija ulaza A,B i C te podnizova X[i+1..i+t] i Y[j+1..j+t]. Dakle, funkcija F ima oblik:

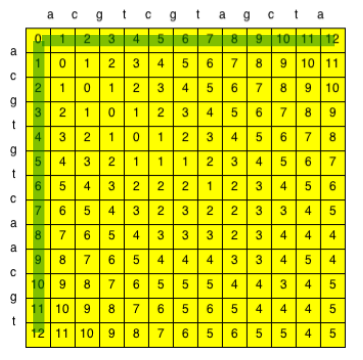
, gdje je *b* tzv. blok funkcija.



Slika 3: T-blok na poziciji (i,j) [1]

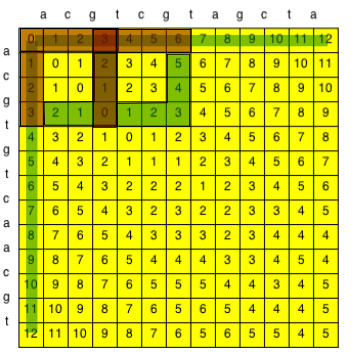
Uz pretpostavku da je blok funkcija već izračunata za sve moguće ulaze (vremenska složenost O(t2)), koraci algoritma su sljedeći

1.       Inicijalizacija prvog retka i stupca početne tablice.



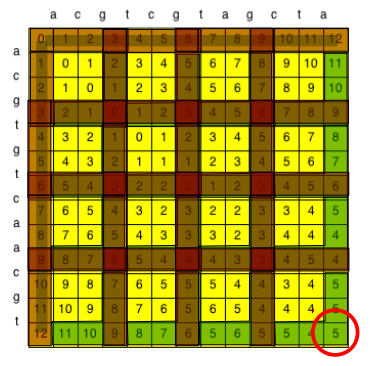
Slika 4: Inicijalizacija prvog retka i stupca u D-tablici.[1]

2.       Podjela tablice na blokove (koji se preklapaju u jednom retku tj. stupcu kao na slici)  i popunjavanje red po red koristeći već izračunatu blok funkciju.



Slika 5: D-tablica s označenim t-blokovima [1]

3.       Ako je D-tablica veličine n x m, konačan rezultat nalazi se u polju D(n,m).



Slika 6: D-tablica s popunjenim t-blokovima te konačnim rezultatom udaljenosti između dva niza zapisanom u polju D(n,m). [1]

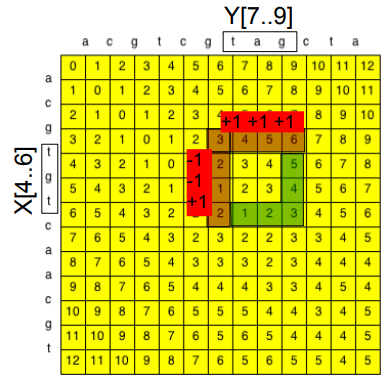
Vremenska složenost prvog koraka je O(n), pri čemu n označava duljinu niza. Treći korak se izvodi u vremenu O(1). Za drugi korak složenost je O(n2/t), dakle ovisi o duljini niza što može biti loše za iznimno duge nizove.

Ukupna vremenska složenost je O(n2/t) uz vrijeme koje je potrebno za izračun blok funkcije.[1]

## Smanjenje broja mogućih vrijednosti o kojima ovisi blok funkcija

Moguće je smanjiti broj mogućih ulaza u blok funkciju koristeći sljedeće zaključke:

1. Razlika susjednih ćelija u popunjenoj D-tablici je najviše jedan. Ovo nam omogućuje da B i C zapišemo kao vektore razlike od A:



Slika 7: D-tablica s označnim vektorima razlike na t-bloku [1]

Za slučaj prikazan na slici: A=3, B=(4,5,6), C=(2,1,2) tj. vektori razlike: B'=(1,1,1) i C'=(-1,-1,1).

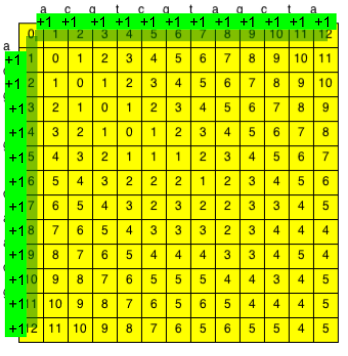
1. Koristeći prethodni zaključak možemo napisati blok funkciju koja ovisi samo o vrijednostima vektora razlike B' i C' te više ne ovisi o vrijednosti parametra A:

Broj mogućih ulaza sada je 3t 3t |⅀|2t = 32t|⅀|2t = (3|⅀|)2t i više ne ovisi o n. Ako je t=(log3|⅀|n)/2 tada ukupna vremenska složenost računanja blok funkcije postaje (3|⅀|)(log3|⅀|n) x (t2) = n((log3|⅀|n)/2)2. Dakle, radi se o O(nlog2n) složenosti umjesto prijašnje O(n2). [3]

## Konačan algoritam

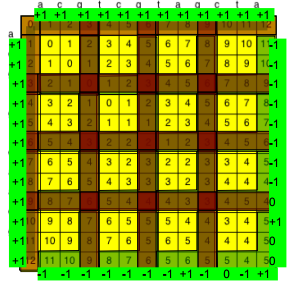
Već navedeni koraci algoritma mijenjaju se, zbog promjene (poboljšanja) blok funkcije, na sljedeći način:

1. Inicijalizacija prvog retka i stupca početne tablice uz računanje razlike između susjednih polja (uvijek +1).



Slika 8: D-tablica s inicijaliziranim vektorima razlike za prvi redak tj. stupac [1]

1. Podjela tablice na blokove (koji se preklapaju u jednom redu i stupcu kao na slici) i popunjavanje red po red koristeći unaprijed izračunatu blok funkciju.



Slika 9: D-tablica s izračunatim konačnim vektorima razlike [1]

1. Rezultat je zbroj dužine niza i svih razlika posljednjeg reda.

# Provedba algoritma na primjeru

Nizovi na kojima će biti proveden algritam su: ACTT i GATA. Veličina t-bloka koju ćemo koristiti u primjenu je 2 (t=2), također, koristi se blok funkcija opisana u poglavlju 2.2.. Slijede koraci algoritma:

* 1. Računanje vektora razlike za prvi par podnizova (AC i GA):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | A | C |
|  | 0 0 | +1 1 | +1 2 |
| G | +1 1 | +1 2 | 0 2 |
| A | +1 2 | -1 1 | +1\0 2 |

* 1. Računanje vektora razlike za drugi par podnizova (TT i GA):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | T | T |
|  | +1 2 | +1 3 | +1 4 |
| G | 0 2 | +1 3 | 0 4 |
| A | 0 2 | +1 3 | +1\0 4 |

* 1. Računanje vektora razlike za treći par podnizova (AC i GA):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | A | C |
|  | +1 2 | -1 1 | +1 2 |
| T | +1 3 | +1 2 | 0 2 |
| A | +1 4 | -1 3 | 0\+1 3 |

* 1. Računanje vektora razlike za četvrti par podnizova (TT i TA):

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | T | T |
|  | +1\0 2 | +1 3 | +1 4 |
| T | 0 2 | +1 2 | -1 3 |
| A | +1 3 | 0 3 | 0\0 3 |

Konačno, cijela tablica:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | A | C | A | C |
|  | 0 0 | +1 1 | +1 2 | +1 3 | +1 4 |
| G | +1 1 |  | 0 2 |  | 0 4 |
| A | +1 2 | -1 1 | +1\0 2 | +1 3 | +1\0 4 |
| G | +1 3 |  | 0 2 |  | -1 3 |
| A | +1 4 | -1 3 | 0\+1 3 | 0 3 | 0\0 3 |

# Rezultati

# Zaključak

# Literatura

[1] K. Shi, “Four Russians” Speed-Up to the Edit Distance Problem, Spring 2016, <https://www.ugrad.cs.ubc.ca/~cs490/2015W2/lectures/Four%20Russians.pdf>

[2] M. Gilleland, Levenshtein Distance, in Three Flavors, <https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm>

[3] Speeding up dynamic programming, The Four Russians, <http://cs.au.dk/~cstorm/courses/AiBS_e12/slides/FourRussians.pdf>

1. Algoritam je poznat pod ovim nazivom, iako je samo jedan od autora bio rus. [↑](#footnote-ref-1)