SVEUČILIŠTE U ZAGREBU

**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

**PROJEKT IZ KOLEGIJA BIOINFORMATIKA**

**Računanje udaljenosti uređivanja algoritmom 4 Russians**

Borna Ivanković, Josipa Popović, Iva Topolovac

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

Zagreb, siječanj, 2017.

Sadržaj

[1. Uvod 3](#_Toc503736321)

[2. Opis rada algoritma 4 Russians 4](#_Toc503736322)

[2.1 Osnovna ideja 4](#_Toc503736323)

[2.2 Smanjenje broja mogućih vrijednosti o kojima ovisi blok funkcija 6](#_Toc503736324)

[2.3 Konačan algoritam 7](#_Toc503736325)

[3. Provedba algoritma na primjeru 10](#_Toc503736326)

[4. Rezultati 11](#_Toc503736327)

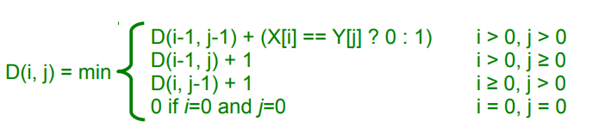
[5. Zaključak 12](#_Toc503736328)

[6. Literatura 13](#_Toc503736329)

[7. Sažetak 14](#_Toc503736330)

# Uvod

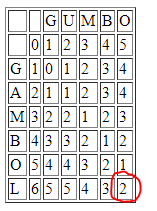
Koristeći dinamičko programiranje koje se temelji na jednostavnoj rekurziji:



Relacija 1: Relacija za računanje udaljenosti između nizova

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

Npr.:



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

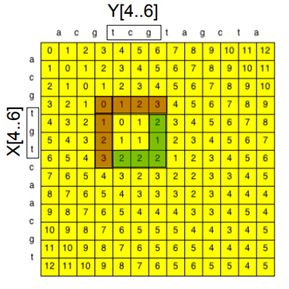
Ispostavilo se da je moguće izračunati udaljenost dvaju nizova i sa manjom složenošću O(n2/logn) uz uvjet n≥m, i to koristeći algoritam “4 Russians”[[1]](#footnote-1). Ovaj algoritam pojavljuje se 1970. kada se primjenjuje za množenje Boolean matrica (Arlazarov, Dinic, Kronrod, Faradzev) te nešto kasnije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

# Opis rada algoritma 4 Russians

## Osnovna ideja

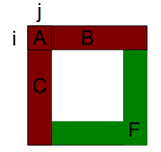
Da bismo bolje razumjeli algoritam uvodimo pojam t-blok. T-blok je blok veličine t x t.

Npr. Na slici je u tzv. D-tablici označen t-blok, za t=3 na poziciji (3,3).



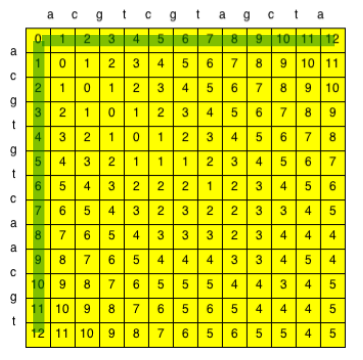
Za t-blok na nekoj poziciji (i,j) izlaz F je funkcija ulaza A,B i C te podnizova X[i+1..i+t] i Y[j+1..j+t] čija je vremenska složenost O(t2). Dakle, funkcija F ima oblik:

F= b(A, B, C, X[i+1..i+1], Y[j+1..j+t]),  gdje je b tzv. blok funkcija.

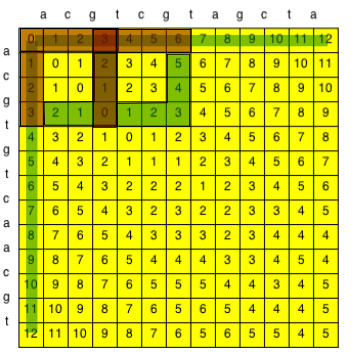


Uz pretpostavku da je blok funkcija već izračunata za sve moguće ulaze, koraci algoritma su sljedeći:

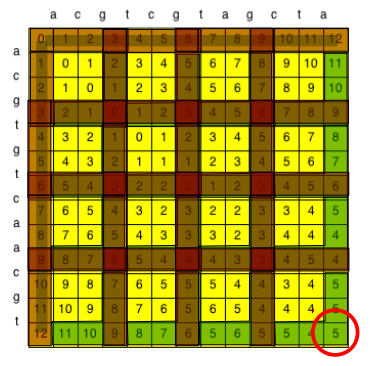
1.       Inicijalizacija prvog retka i stupca početne tablice.



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.



3.       Konačan rezultat je u polju D(n,m).

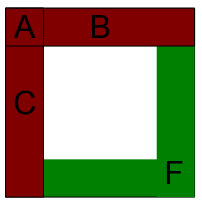


Vremenska složenost prvog koraka je O(n), a trećeg O(1). U drugom koraku vremenska složenost jednaka je O(n2/t2) x O(t) (veličina ulaza i izlaza blok funkcije), dakle O(n2/t).

Ukupna vremenska složenost je O(n2/t) + vrijeme koje je potrebno za izračun blok funkcije.

## Smanjenje broja mogućih vrijednosti o kojima ovisi blok funkcija

Računanje blok funkcije (b(A, B, C, X[i+1..i+1], Y[j+1..j+t]) za svaki ulaz ima vremensku složenost O(t2).



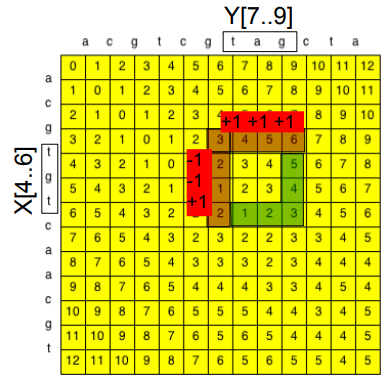
Ako npr. promatramo element D-tablice: 0 ≥ D(i,j) ≥ n:

1. Za A postoji n+1 različitih mogućih vrijednosti
2. Za B i C postoji (n+1)trazličitih mogućih vrijednosti
3. Za x i y postoji |⅀|t različitih mogućih vrijednosti

Što je ukupno:  (n+1)2t+1|⅀| 2t mogućih ulaza. Potrebno je O(t2) vremena kako bi izračunali izlaz za svaki mogući ulaz blok funkcije, što daje  (n+1)2t+1|⅀| 2t x t2   tj. najmanju složenost od O(n2) jer je t najmanje 1.

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

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



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 B' i C': b'(A, B', C', x, y) = b'(0, B', C', x, y) (više ne ovisi o A).

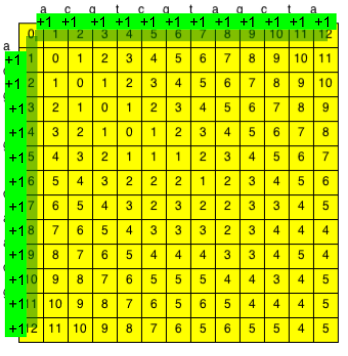
Broj mogućih ulaza sada je 3t 3t |⅀|2t = 32t|⅀|2t = (3|⅀|)2t i više ne ovisi o n. Ako je t=(log3|⅀|n)/2 tada ukupna vremenska složenost računanja blok funkcije postaje (3|⅀|)(log3|⅀|n) x (t2) = n((log3|⅀|n)/2)2. Dakle, radi se o O(nlog2n) složenosti.

Ukupna vremenska složenost sada je O(n2 / log(n)) umjesto prijašnjih O(n2).

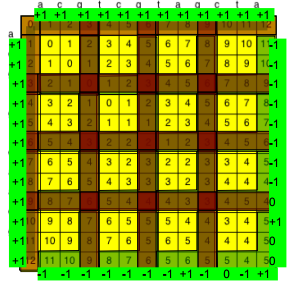
## Konačan algoritam

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

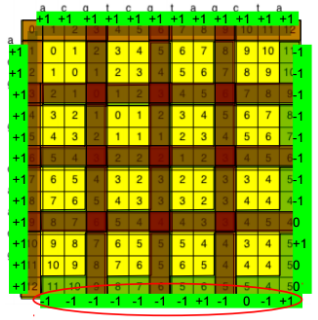
1. Inicijalizacija prvog retka i stupca početne tablice uz računanje razlike između susjednih polja (uvijek +1).



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.



1. Rezultat je n+suma svih razlika posljednjeg reda.



# Provedba algoritma na primjeru

# Rezultati

# Zaključak

# Literatura

# Sažetak

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