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

Koristeći dinamičko programiranje koje se temelji na jednostavnoj rekurziji:

$$D(i, j) = \min \begin{cases} D(i-1, j-1) + (X[i] == Y[j] \ ? \ 0 : 1) & i > 0, j > 0 \\ D(i-1, j) + 1 & i > 0, j \geq 0 \\ D(i, j-1) + 1 & i \geq 0, j > 0 \\ 0 \text{ if } i=0 \text{ and } j=0 & i = 0, j = 0 \end{cases}$$

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.:

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

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

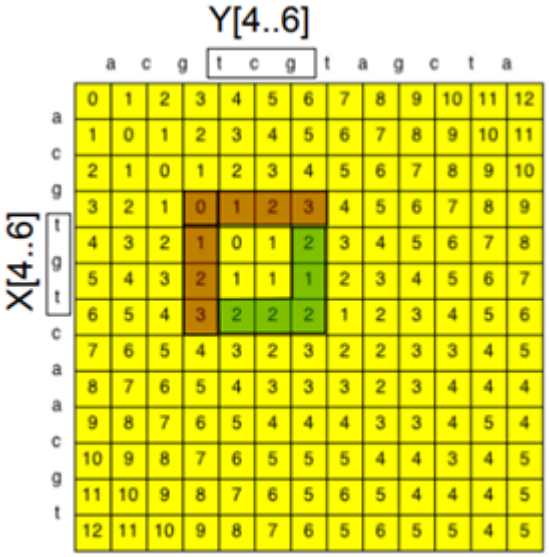
Ispostavilo se da je moguće izračunati udaljenost dvaju nizova i sa manjom složenošću $O(n^2/\log n)$ uz uvjet $n \geq m$, i to koristeći algoritam “4 Russians”[1]. Ovaj algoritam pojavljuje se 1970. kada se primjenjuje za množenje Boolean matrica (Arlazarov, Dinic, Kronrod, Faradzev) te nešto kasnije nalazi primjenu u računanju udaljenosti između nizova znakova(Masek, Paterson).

[1] Algoritam je poznat pod ovim nazivom, iako je samo jedan od autora bio rus.

2. Opis rada algoritma 4 Russians

2.1 Osnovna ideja

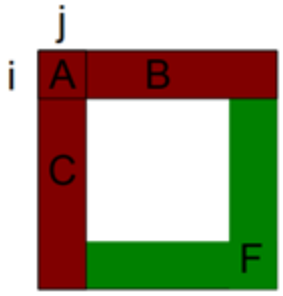
Da bismo bolje razumjeli algoritam uvodimo pojam t-blok. T-blok je blok veličine t x t. Npr. Na slici je u tzv. D-tablici (tablica udaljenosti nizova ispunjena koristeći Levenstheinov algoritam) označen t-blok, za t=3 na poziciji (3,3).



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:

$F = b(A, B, C, X[i+1..i+t], Y[j+1..j+t])$, 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 $O(t^2)$), koraci algoritma su sljedeći

1. Inicijalizacija prvog retka i stupca početne tablice.

		a	c	g	t	c	g	t	a	g	c	t	a
	0	1	2	3	4	5	6	7	8	9	10	11	12
a	1	0	1	2	3	4	5	6	7	8	9	10	11
c	2	1	0	1	2	3	4	5	6	7	8	9	10
g	3	2	1	0	1	2	3	4	5	6	7	8	9
t	4	3	2	1	0	1	2	3	4	5	6	7	8
g	5	4	3	2	1	1	1	2	3	4	5	6	7
t	6	5	4	3	2	2	2	1	2	3	4	5	6
c	7	6	5	4	3	2	3	2	2	3	3	4	5
a	8	7	6	5	4	3	3	3	2	3	4	4	4
a	9	8	7	6	5	4	4	4	3	3	4	5	4
c	10	9	8	7	6	5	5	5	4	4	3	4	5
g	11	10	9	8	7	6	5	6	5	4	4	4	5
t	12	11	10	9	8	7	6	5	6	5	5	4	5

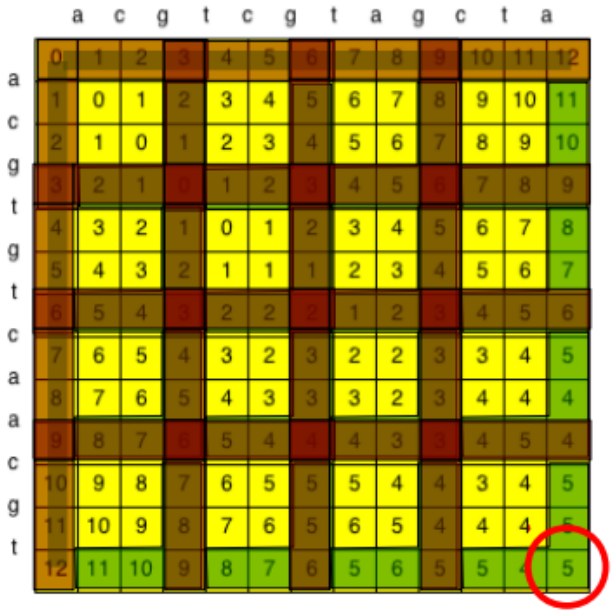
Slika 4: Inicijalizacija prvog retka i stupca u D-tablici.[1]

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.

		a	c	g	t	c	g	t	a	g	c	t	a
	0	1	2	3	4	5	6	7	8	9	10	11	12
a	1	0	1	2	3	4	5	6	7	8	9	10	11
c	2	1	0	1	2	3	4	5	6	7	8	9	10
g	3	2	1	0	1	2	3	4	5	6	7	8	9
t	4	3	2	1	0	1	2	3	4	5	6	7	8
g	5	4	3	2	1	1	1	2	3	4	5	6	7
t	6	5	4	3	2	2	2	1	2	3	4	5	6
c	7	6	5	4	3	2	3	2	2	3	3	4	5
a	8	7	6	5	4	3	3	3	2	3	4	4	4
a	9	8	7	6	5	4	4	4	3	3	4	5	4
c	10	9	8	7	6	5	5	5	4	4	3	4	5
g	11	10	9	8	7	6	5	6	5	4	4	4	5
t	12	11	10	9	8	7	6	5	6	5	5	4	5

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).



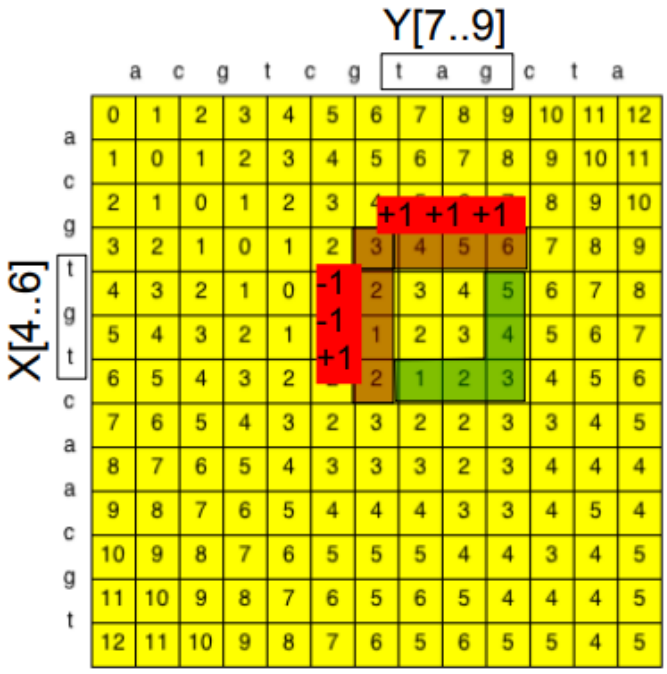
Slika 6: D-tablica s popunjenim t-blokovima te konačnim rezultatom udaljenosti između dva niza zapisanom u polju D(n,m). [1]

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 $O(n^2/t)$, dakle ovisi o duljini niza što može biti loše za iznimno duge nizove.
Ukupna vremenska složenost je $O(n^2/t)$ uz vrijeme koje je potrebno za izračun blok funkcije.[1]

2.2 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).

2. 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:

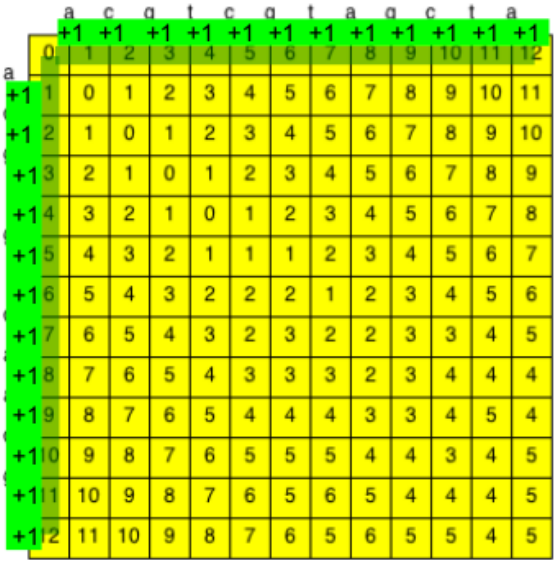
$$b'(A, B', C', x, y) = b'(0, B', C', x, y)$$

Broj mogućih ulaza sada je $3^t\, 3^t\, |\sum|^{2t} = 3^{2t}|\sum|^{2t} = (3|\sum|)^{2t}$ i više ne ovisi o n. Ako je $t=(\log_{3|\sum|}n)/2$ tada ukupna vremenska složenost računanja blok funkcije postaje $(3|\sum|)^{(\log 3|\sum|n)} \times (t^2) = n((\log_{3|\sum|}n)/2)^2$. Dakle, radi se o $O(n\log^2 n)$ složenosti umjesto prijašnje $O(n^2)$. [3]

2.3 Konačan algoritam

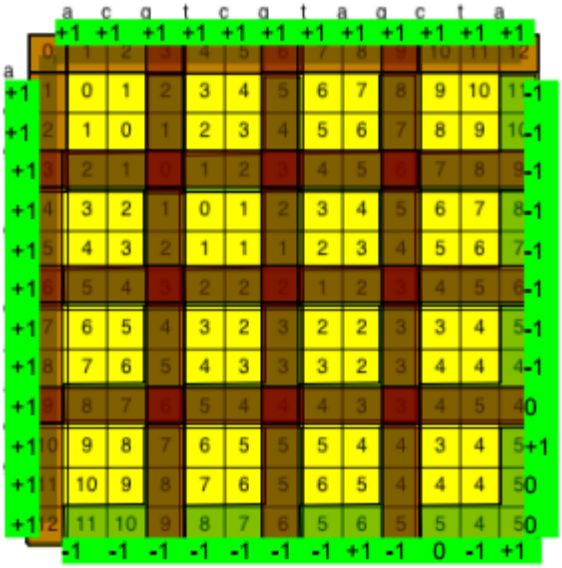
Već navedeni koraci algoritma mijenjaju se, zbog promjene (poboljšanja) blok funkcije, na sljedeći način:

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]

2. 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]

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

3. 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]	[2]	0 [2]
A	+1 [2]	-1 [1]	+1\0 [2]

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

		T	T
	+1 [2]	+1 [3]	+1 [4]
G	0 [2]	[3]	0 [4]
A	0 [2]	+1 [3]	+1\0 [4]

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

		A	C
	+1 [2]	-1 [1]	+1 [2]
T	+1 [3]	[2]	0 [2]
A	+1 [4]	-1 [3]	0\+1 [3]

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

		T	T
	+1\0 2	+1 3	+1 4
T	0 2	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

4. Rezultati

U ovom poglavlju su dani rezultati pokretanja programa na nekoliko primjera sintetskih podataka te na nekoliko primjera stvarnih podataka (Escherichia coli).

Example	Syn 1	Syn 2	Syn 3	Syn 4	Syn 5
String length	204	2004	20004	40004	100004
T-Block size	2	2	2	3	3
Edit distance [s]	57	530	5204	10353	25842
T-Block generation time [s]	0.0426667	0.0447933	0.0449911	4.67616	4.65971
D-Table fill time [s]	0.0011349	0.0789204	7.35662	23.9254	155.854
Edit script generation time [s]	0.0046047	0.0072503	0.0186713	0.0343178	0.0726609
Total time [s]	0.0484063	0.130964	7.42028	28.6359	160.586
Memory usage [MB]	0.968292	2.85657	191.684	475.704	2255.79

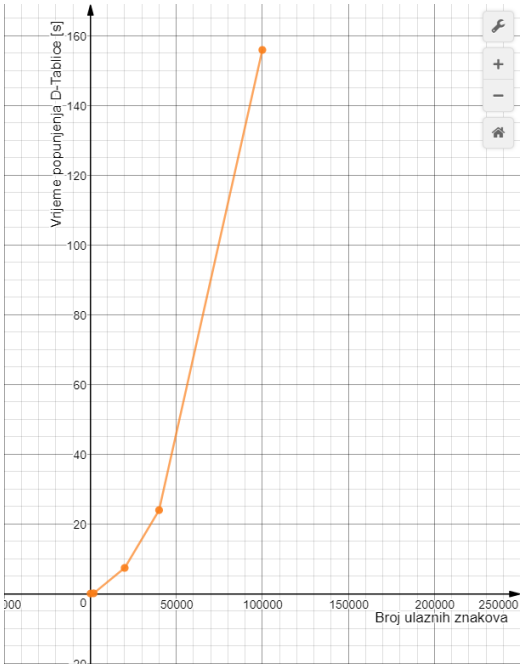
Tablica 1. Rezultati testiranja na sintetskim podacima

Example	E. Coli 1	E. Coli 3	E. Coli 6	E. Coli 7	E. Coli 10
String length	412	2854	11728	20048	28902
T-Block size	2	2	2	2	3
Edit distance [s]	3	151	3117	5433	7580
T-Block generation time [s]	0.0454564	0.0432549	0.0437785	0.0437949	4.60871
D-Table fill time [s]	0.0030135	0.0072067	2.50464	7.33515	12.4178
Edit script generation time [s]	0.0014331	0.0015315	0.00110438	0.0169109	0.234351
Total time [s]	0.049903	0.0519931	2.55946	7.39586	17.2608
Memory usage [MB]	1.02859	1.08621	66.4915	192.524	313.643

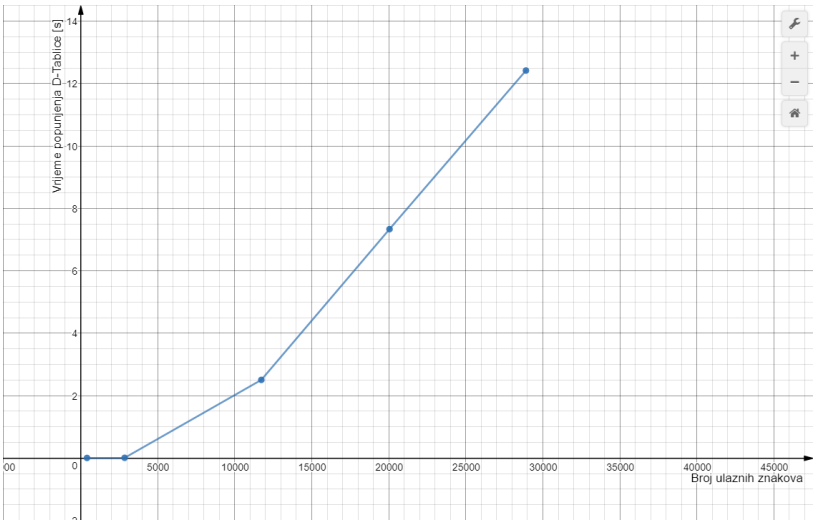
Tablica 2. Rezultati testiranja na stvarnim podacima (Escherichia coli)

Iz prethodno navedenih tablica možemo izvući nekoliko zaključaka:

- Za konstantnu veličinu T-Bloka vrijeme potrebno za generiranje svih blokova je jednako (ukoliko dopustimo malo pogrešku zbog različitog stanja sustava prilikom pokretanja raznih primjera)
- Vrijeme punjenja D-Tablice raste eksponencijalno s brojem znakova ulaznog niza
- Vrijeme za generiranje skripte se neznatno povećava s brojem znakova ulaznog niza



Slika 10. Ovisnost vremena generiranja D-Tablice o duljini ulaznog niza za sintetičke podatke



Slika 11. Ovisnost vremena generiranja D-Tablice o duljini ulaznog niza za stvarne podatke

5. Zaključak

Kroz ovaj projekt smo naučili kako pronalaženje minimalne udaljenosti dvaju nizova te proces poravnavanja istih nije nimalo trivijalan zadatak. Vrlo brzo se može uočiti kako je potrebno provesti mnogo tehnika optimizacija kako bi što više smanjili količinu računalnih resursa koji se koriste unatoč tome što smo implementirali 4 Russians algoritam koji je već optimiran u odnosu na Needelman – Wunsch algoritam na kojem se temelji. Nadalje, kao što je prikazano u prethodnom poglavlju, možemo primijetiti kako je ova implementacija algoritma izuzetno efikasna na nizovima manje duljine te da porastom duljine niza performanse značajno opadaju. Kroz ovaj projekt smo se upoznali s mnogim tehnikama optimizacije te smo naučili koliko je ona uistinu bitna.

6. 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