SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Navarrov algoritam za približno uspoređivanje teksta

Jana Juroš i Josip Skender
Voditelj: *Krešimir Križanović*

Zagreb, svibanj 2021.

SADRŽAJ

1. Uvod

Približno uspoređivanje teksta predstavlja problem pronalaska sekvence koja sadrži neki traženi uzorak, uzimajući u obzir *"udaljenost"*. Postoji mnogo algoritama koji obrađuju navedenu problematiku, jedan od njih je *Navarrov algoritam*. U ovom radu opisali smo njegovu funkcionalnost te ga na koncu usporedili s *bit parallel sequence-to-graph alignment* algoritmom.

2. Navarrov algoritam

Navarrov algoritam oslanja se na *hypertextu* kao strukturi po kojoj pretražujemo najmanju *udaljenost* nekog uzorka.
Najmanja *udaljenost* predstavlja najmanji broj operacija potreban da se neka dva uzorka izjednače. Moguće operacije su: *dodavanje, brisanje i substitucija.*
Hypertext u obliku usmjerenog grafa pohranjuje sadržaj kao čvorove koji u sebi nose *char* informaciju. Svaki čvor sadrži referencu na svoje prethodne i sljedeće čvorove.
Neka je p urozak koji tražimo i G graf po kojem se vrši pretraga. Za provedbu algoritma potrebno je izvršiti i iteracija po svakom čvoru od G, gdje je i duljina od p. Pri iteriranju svaki čvor N poprima vrijednost C koja je jednaka najmanjoj $udaljenosti$ bilokojeg puta u G koji završava u N. Efektivno za određivanje čvorova s najmanjom C vrijednošću za p kroz cijeli algoritam u bilokojem koraku potrebno je samo imati informaciju od prethodne iteracije te pomoću nje izračunavamo trenutnu vrijednost.
U slučaju grafova koji ne sadrže cikluse, obilazak čvorova po redoslijedu ne predstavlja nikakav problem. Kada slijedno obrađujemo čvorove grafa može se desiti da krivo izračunamo C vrijednost čvora u kojem započinje neki ciklus. Ukoliko bi taj čvor imao manju C vrijednost kada bi se uzele u obzir vrijednosti s kraja ciklusa (kasnije vrijednosti po topološkom poretku) dolazi do krivog rješenja.
Da bi se spriječila ova pojava, uklanjamo mogućnost *dodavanja* kod izračuna C te stvaramo novu rekurzivnu funkciju koja za svaki čvor kojem je potrebna redukcija provodi navedenu.

i $\Rightarrow$ *broj iteracije*
Nu $\Rightarrow$ *Skup svih čvorova koji ulaze u N*
E $\Rightarrow$ *skup svih rubova grafa G*

Funkcija za određivanje C vrijednosti:
C = f(N,i) =
    if (p[i] = N.value):

return min(C ∈ Nu, i-1)

else:

return 1 + min(C ∈ Nu, C vrijednost od N u prethodnoj iteraciji)

Rekurzivna funkcija Redukcije:

Reduciraj ($u \in Nu$, N):

if C_N>$1 + C_u$:

$C_N = 1 + C_u$

for all x/(N, x ∈svi sljedbenici od N):

Reduciraj(x, N)

Gonzalo Navarro algoritam:

GonzaloNavarro (G, p):

for all $N \in G \leftarrow C_N = 0$

for $i = 1$ to $m = len(p)$

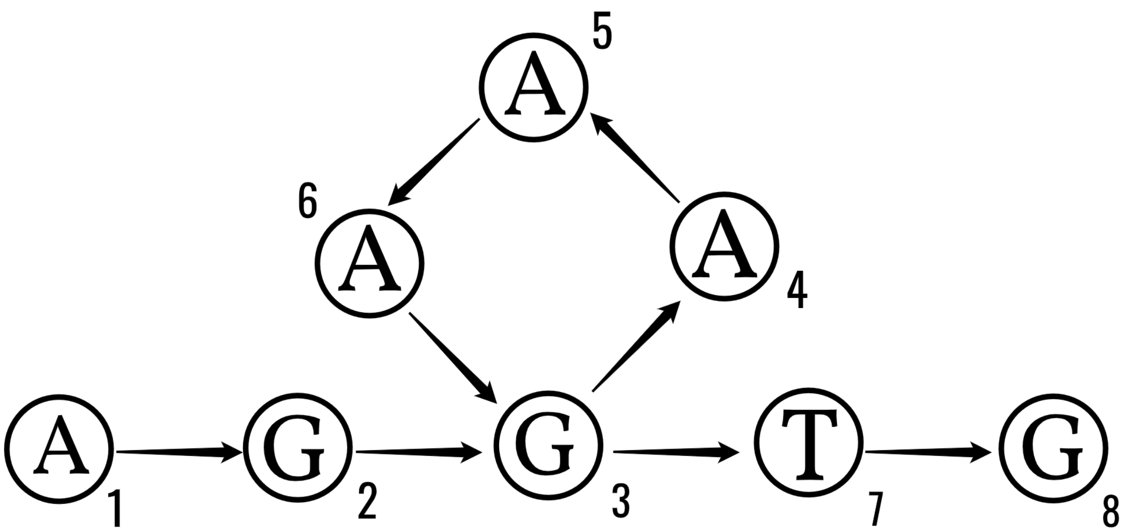
for all $N \in G$, $C'_n \leftarrow f(N, i)$

for all $N \in G$, $C_n \leftarrow C'_n$

for all $(N_u, N) \in G$, Reduciraj(N_u, N)

Primjer funkcionalnosti algoritma:

Usmjereni graf G:



Uzorak p = '**AAAA**'

		1	2	3	4	5	6	7	8
-	C'	0	0	0	0	0	0	0	0
	C	0	0	0	0	0	0	0	0
A	C'	0	1	1	0	0	0	1	1
	C	0	1	1	0	0	0	1	1
A	C'	1	1	1	1	0	0	2	2
	C	1	1	1	1	0	0	2	2
A	C'	2	2	1	1	1	0	2	3
	C	2	2	1	1	1	0	2	3
A	C'	3	3	1	1	1	1	2	3
	C	3	3	1	1	1	1	2	3

Navedena tablica prikazuje svaki korak iteracije algoritma.
Stupci tablice prikazuju vrijednosti C' i C vrijednosti svakog čvora grafa.
C' vrijednost izračuna se po redu za svaki čvor grafa te se tek na kraju iteracije uvrsti u C.
Nakon izračunate C vrijednosti za svaki čvor vrši se provjera **Redukcije** te njena daljnja propagacija.
Tijek izvođenja vrši se po retcima tablice.
C vrijednost u bilokojem koraku prikazuje nam najmanju *Udaljenost* nekog puta koja završava u čvoru s tom vrijednošću.
Ako gledamo 3. redak ispunjene tablice, možemo uočiti da najmanju vrijednost $C = 0$ poprima čvor 6. što nam znači da uspoređvanje uzorka **'AAA'** s G najbolje pristaje nekom putu koji završava u čvoru 6, u ovom slučaju to bi bio put **4** $\rightarrow$ **5** $\rightarrow$ **6**. Čije vrijednosti čvorova su upravo jednake '**AAA**'.
Uzmimo sad cijeli p, promatranjem 4. retka tablice uočavamo da je najmanja vrijednost $C = 1$ te nju poprimaju čvorovi **3, 4, 5 i 6**. Ti čvorovi su dio ciklusa grafa G, ovisno koji čvor odaberemo kao kraj puta, vrijednosti čvorova iznose: **AAAG**, **AAGA**, **AGAA** i **GAAA**. Odavde uočavamo da je za svaki od ovih puteva potrebna samo jedna operacije substitucije $G \rightarrow A$ i time potvrđujemo izračunatu C vrijednost.

Vremenska složenost ovog algoritma je $O(m(n + e))$.
Prostorna složenost s obrizom da je potrepno znati samo prethodnu iteraciju jednaka je $O(n)$.
Gdje je m duljina od p, n broj svih čvorova G te e broj svih rubova E.

3. Usporedba s bitparallel algoritmom

Za razliku od *Navarrovog algortima*, *bitparallel algoritam* provodi se u smjeru stupaca gdje su stupci čvorovi grafa a retci vrijednost uzoraka koji uspoređujemo. Svaki red matrice u *Bitparallel algoritmu* sastoji se od dvaju **bitvectora** (*pozitivan bitvectorP i negativan bitvectorN*) i S_{end} varijable koja čuva rezultat.

Usporedba rezultata naše implementacije **Navarrovog algoritma** i **Bitvector algoritma**:

4. Zaključak

Navarrov algoritam u konačnosti pronađe ispravno poravnanje za svaki od ispitanih grafova. Uspoređujući **Navarrov algoritam** i **Bitparallel algoritam** jasno je vidljv njegov nedostatak u brzini. S obzirom na ulazni skup podataka Navarrov algoritam je u prosjeku X sporiji od *bitvector algoritma.*

5. Literatura

[1] Gonzalo Navarro, Improved approximate pattern matching on hypertext, Theoretical Computer Science 237, 2000, pp. 455-463

[2] M.Rautiainen, V.Mäkinen, T.Marschall, Bit-parallel sequence-to-graph alignment, Bioinformatics, Volume 35, Issue 19, 2019, pp. 3599-3607