Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Efektívna reprezentácia množiny krátkych reťazcov

Bakalárska práca

2016
Jaroslav Petrucha

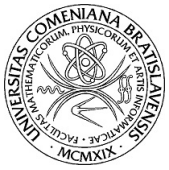
Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Efektívna reprezentácia množiny krátkych reťazcov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	Mgr. Tomáš Vinař, PhD.

Bratislava, 2016
Jaroslav Petrucha



39182234

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

PRIHLÁŠKA NA ZÁVEREČNÚ PRÁCU

Meno a priezvisko študenta: Jaroslav Petrucha
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Efektívna reprezentácia množiny krátkych reťazcov

Cieľ: Pri sekvenovaní DNA vzniká množstvo krátkych reťazcov, ktoré je potrebné reprezentovať v rámci efektívnych dátových štruktúr. Jednou z možných reprezentácií je nadslovo, ktoré obsahuje všetky podreťazce dĺžky k týchto reťazcov. Úlohou je navrhnúť praktické algoritmy na vytvorenie takéhoto nadslova a vyhodnotiť úspešnosť ich aplikácie na sekvenovacie dáta druhej generácie.

Vedúci: Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum schválenia: 28.10.2015

podpis študenta

Poďakovanie:

Abstrakt

Abstrakt

Kľúčové slová: Kľučové slovo

Abstract

Abstract

Keywords: Keywords

Obsah

Zoznam obrázkov

Úvod

Kapitola 1

Základné pojmy, motivácia a súvisiace práce

V tejto kapitole čiatateľa oboznámime so základnými biologickými a informatickými pojmami, s ktorými sa môže stretnúť v tejto práci, s motiváciou pre riešenie tohto problému a s inými, súvisiacimi prácami v tejto oblasti.

Kapitola 2

Požiadavky na riešenie

Problém, ktorý riešime v tejto práci sa dá rozdeliť na tri základné úrovne podľa toho, ako veľmi berieme do úvahy požadovanú funkcionalitu a povahu reálnych dát. Pôjde o:

1. Hľadanie k-nadslova
2. Indexovanie do k-nadslova
3. Tolerancia chýb v dátach

V tejto kapitole sa pozrieme na každú z týchto úrovní. Popíšeme spôsob, akým zasahujú do riešenia, ako ovplyvňujú ťažkosť a zložitosť a popíšeme si, ako problém danej úrovne budeme riešiť.

2.1 Hľadanie k-nadslova

Prvým a najmenej zložitým problémom je hľadanie najkratšieho k-nadslova. Na pripomenuie, ide o nadslovo, ktoré obsahuje všetky podslová dĺžky k zo zadaných slov.

Mohlo by sa zdať, že ide o slabšiu verziu problému hľadania spoločného nadslova, keďže ľubovoľné nadslovo nejakej množiny reťazcov bude zároveň k-nadslovom tejto množiny reťazcov, ale ľahko vidno, že naopak to neplatí. Ukazuje sa však, že rovnako, ako problém hľadania najkratšieho nadslova, je aj problém hľadania najkratšieho k-nadslova $\mathcal{NP}$-ťažký.

2.1.1 $\mathcal{NP}$-ťažkosť problému

Rozhodovacia verzia problému najkratšieho spoločného nadslova je pomocou redukcie z problému Hamiltonovskej cesty v orientovanom grafe s obmedzením na stupne vrcholov a následnej redukcie na tento pomocný problém z problému Hamiltonovskej cesty v jednoduchom grafe, $\mathcal{NP}$-úplný, pokiaľ je splnená aspoň jedna z nasledujúcich podmienok[2]:

- Veľkosť abecedy, nad ktorou máme zadané slovo je neobmedzená
- Veľkosť abecedy je aspoň 2 a existuje $h \geq 1$ také, že každý vstupný reťazec má dĺžku $\lceil h \cdot log_2(S) \rceil$, kde S je celková dĺžka vstupných slov.

Pokiaľ by sme vedeli efektívne riešiť rozhodovací problém najkratšieho k-nadslova, na vstupe s rovnako dlhými vstupnými reťazcami a k rovným dĺžke týchto reťazcov by bolo riešenie zhodné s riešením rozhodovacieho problému najkratšieho spoločného nadslova. Preto ak by sme mali polynomiálny algoritmus, ktorý vie riešiť rozhodovací problém najkratšieho k-nadslova, mali by sme zároveň polynomiálny algoritmus riešiaci problém Hamiltonovskej cesty. Trieda takýchto vstupov spĺňa druhú zo spomenutých podmienok.

Z toho vyplýva $\mathcal{NP}$-ťažkosť rozhodovacieho problému najkratšieho k-nadslova a tým pádom aj $\mathcal{NP}$-ťažkosť problému hľadania najkratšieho k-nadslova.

2.1.2 Abstrakcia

Po zdôvodnení a dokázaní ťažkosti riešeného problému sa pozrieme na spôsob, akým ho budeme riešiť. Ako prvé si objasníme abstrakciu problému, na ktorej postavíme celé riešenie.

Podobne ako pri dokazovaní ťažkosti problému si pomôžeme grafovou teóriou. Celú sadu vstupných slov si budeme reprezentovať ako mierne upravený ohodnotený de Bruijnov graf stupňa $k-1$ nad abecedou $\{A, C, G, T\}$. Každej hrane symbolicky priradíme reťazec, ktorého dĺžka bude vždy zároveň hodnotou hrany. Naše riešenie, čiže nejaké spoločné k-nadslovo, budeme reprezentovať ako sled v tomto grafe.

Vrcholmi v tomto grafe budú všetky $(k-1)$-tice písmen ktoré sa vyskytujú v aspoň jednom slove. Z vrchola $(x_1x_2x_3 \ldots x_{k-1})$ bude do vrchola $(x_2x_3 \ldots x_k)$ viesť hrana, ktorú označíme ako *nutnú*, práve vtedy, ak slovo $x_1x_2x_3 \ldots x_k$ je podslovom niektorého

zo vstupných slov. Každej *nutnej* hrane priradíme jednoprvkový reťazec pozostávajúci z x_k, čiže posledného znaku vo vrchole, do ktorého smeruje.

Môžeme si všimnúť, že v takomto grafe budú všetky k-tice písmen, ktoré sú ako podslovo niektorého vstupného slova, reprezentované ako *nutné* hrany. Ďalej potrebujeme zaviesť pojem *nepovinnej* hrany.

Ak z vrchola S_1 nevedie *nutná* hrana do vrchola S_2, tak z vrchola S_1 vedie do vrchola S_2 *nepovinná* hrana. Pre zadefinovanie hodnoty hrany a priradeného reťazca potrebujeme ešte jeden pojem.

Definícia 1. *Nech s_1 a s_2 sú reťazce znakov a nech z je najdlhší reťazec taký, že*

$$\exists x, y \in \{A, C, G, T\}^* : s_1 = xz \wedge s_2 = zy$$

Potom prekryvom s_1 a s_2 označíme reťazec z a dokončením s_1 v s_2 označíme taký reťazec y, ktorý spĺňa $s_2 = zy$.

Nepovinnej hrane z vrchola S_1 do vrchola S_2 priradíme dokončenie S_1 v S_2. Pripomíname, že hodnotou tejto hrany bude dĺžka priradeného reťazca. Takto skonštruovaný graf k množine vstupných slov S budeme označovať ako ***k*-graf množiny *S***.

Slovo prislúchajúce sledu v k-grafe množiny S získame ako zreťazenie zr, kde z je označenie prvého vrchola sledu a r sú pospájané reťazce priradené hranám v takom poradí, v akom sa tieto hrany nachádzajú v slede.

Formálnejšie zapísané, nech $slovo(V)$ označuje reťazec dĺžky $k-1$, označenie vrchola V a $slovo(e)$ označuje reťazec priradený hrane e. Potom $slovo(w)$, slovo prislúchajúce sledu $w = V_1 e_1 V_2 e_2 \dots e_{n-1} V_n$ skonštruujeme ako

$$slovo(w) = slovo(V) \cdot slovo(e1) \cdot slovo(e2) \cdots slovo(e_{n-1}).$$

Sled v k-grafe množiny S budeme volať *korektný*, ak obsahuje všetky *nutné* hrany. Teraz si dokážeme, že slovo prislúchajúce *korektnému* sledu v k-grafe množiny S je k-nadslovom množiny S. K tomu budeme potrebovať ešte jednu pomocnú lemu.

Lema 2. *Nech $V_1 e_1 V_2 e_2 \dots e_{n-1} V_n$ je sled v k-grafe nejakej množiny S. Potom sa reťazec r prislúchajúci tomuto sledu končí $(k-1)$-ticou znakov zhodnou s označením vrchola V_n.*

Dôkaz. Matematickou indukciou na n, dĺžku sledu.

1^0 : Ak n je rovné 1, sled obsahuje iba jeden vrchol. Podľa konštrukcie slova priradeného sledu bude toto slovo totožné s označením prvého a zároveň aj posledného vrchola v slede.

2^0 : Nech $w_1 = V_1 e_1 \dots V_{j-1}$ a $w_2 = V_1 e_1 \dots V_j$. Podľa konštrukcie slova $slovo(w_2)$ platí

$$slovo(w_2) = slovo(V_1) \cdot slovo(e_1) \cdots slovo(e_{j-1}) = slovo(w_1) \cdot slovo(e_{j-1})$$

Ďalej potrebujeme rozobrať dva prípady:

1. Ak je hrana e_{j-1} *nutná*, tak platí

   $$\exists x_1, x_2 \dots x_k \in \{A, C, G, T\} : V_{j-1} = (x_1, \dots, x_{k-1}) \wedge V_j = (x_2, \dots, x_k)$$

   a podľa konštrukcie hrany a priradenia jej reťazca, $slovo(e_{j-1}) = x_n$. Z indukčného predpokladu vyplýva, že $\exists q \in \{A, C, G, T\}^* : slovo(w_1) = q \cdot x_1 \cdot x_2 \cdots x_{k-1}$. Keď to spojíme dohromady, dostaneme

   $$slovo(w_2) = slovo(w_1) \cdot slovo(e_{j-1}) = q \cdot x_1 \cdots x_{k-1} \cdot slovo(e_{j-1}) =$$
   $$= q \cdot x_1 \cdots x_{k-1} \cdot x_k = q' \cdot x_2 \cdots x_k = q' \cdot slovo(V_j),$$

   kde $q' = qx_1$.

2. Ak je hrana e_{j-1} *nepovinná*, tak z konštrukcie *nepovinnej* hrany a jej priradeného reťazca vyplýva:

   $$\exists x, y, z \in \{A, C, G, T\}^* : slovo(V_{j-1}) = xz \wedge slovo(V_j) = zy \wedge slovo(e_{j-1}) = y.$$

   Opäť z indukčného predpokladu vyplýva $\exists q \in \{A, C, G, T\}^* : slovo(w_1) = q \cdot slovo(V_{j-1}) = q \cdot xz$. Keď to poskladáme dohromady, dostaneme

   $$slovo(w_2) = slovo(w_1) \cdot slovo(e_{j-1}) = q \cdot x \cdot z \cdot slovo(e_{j-1}) = q \cdot x \cdot z \cdot y =$$
   $$= q \cdot x \cdot slovo(V_j).$$ □

Veta 3. *Nech W je* korektný *sled v k-grafe množiny S. Potom $slovo(W)$ je k-nadslovom množiny slov S.*

Dôkaz. Vezmime si ľubovoľné slovo w také, že $|w| = k$ a w je podslovom nejakého slova z S. Nech $x_1, x_2, \cdots x_k$ sú znaky tohto slova v poradí. Potom sa v grafe vyskytujú vrcholy $V_1 = (x_1, x_2, \cdots, x_{k-1})$ a $V_2 = (x_2, x_3, \cdots x_k)$ také, že z V_1 ide hrana e do V_2 a e je *nutná*. Z toho vyplýva, že $W = U_1 V_1 e V_2 U_2$, kde U_1, U_2 sú nejaké časti sledu. Z predošlej lemy potom vyplýva, že $\exists q \in \{A, C, G, T\}^* : slovo(U_1 V_1) = q \cdot x_1 \cdots x_{k-1}$ a teda

$$slovo(W) = slovo(U_1 V_1 e V_2 U_2) = q \cdot x_1 \cdots x_{k-1} \cdot slovo(e) \cdot slovo(V_2 U_2) = q \cdot w \cdot slovo(V_2 U_2).$$

□

2.1.3 Vlastnosti abstrakcie

Zatiaľčo každému korektnému sledu v abstrakcii prislúcha práve jedno k-nadslovo, nie každé k-nadslovo vieme reprezentovať ako sled v našom grafe.

Ak si napríklad vezmeme ako vstupné slová `ACG` a `CGA` a k rovné trom, budeme mať v našom grafe tri vrcholy zodpovedajúce dvojiciam `(AC)`, `(CG)` a `(GA)`. Hrany aj s označením, ktoré budeme používať ďalej:

- *nutná* hrana e_1 z vrchola `(AC)` do `(CG)` s priradeným reťazcom `G`,
- *nutná* hrana e_2 z vrchola `(CG)` do `(GA)` s priradeným reťazcom `A`,
- *nepovinná* hrana e_3 z vrchola `(AC)` do `(GA)` s reťazcom `GA`,
- *nepovinná* hrana e_4 z vrchola `(CG)` do `(AC)` s reťazcom `AC`,
- *nepovinná* hrana e_5 z vrchola `(GA)` do `(AC)` s reťazcom `C`,
- *nepovinná* hrana e_6 z vrchola `(GA)` do `(CG)` s reťazcom `CG`.

Celkom ľahko vidíme, že najkratšie možné k-nadslovo musí mať aspoň 4 znaky. Vhodné k-nadslovo takejto dĺžky naozaj existuje, napríklad `ACGA`. V našom grafe ho vieme reprezentovať ako sled

$$\texttt{(AC)}e_1\texttt{(CG)}e_2\texttt{(GA)}$$

Ak sa ale pozrieme napríklad na slovo `TTACGTTTCGATT`, neexistuje žiaden sled, ktorému prislúcha toto k-nadslovo, keďže žiaden vrchol ani hrana neobsahujú znak `T`. Ľahko ale vidíme, že toto slovo vieme skrátiť vynechaním niektorých znakov na stále vyhovujúce k-nadslovo `ACGCGA`. Ďalej si dokážeme, že toto pozorovanie vieme zovšeobecniť a ukážeme si, ako to vplýva na robustnosť našej abstrakcie problému.

Veta 4. *Nech slovo s je k-nadslovom množiny S. Potom ak neexistuje sled W v k-grafe množiny S taký, že slovo$(W) = s$, existuje slovo s' také, že s' je k-nadslovom množiny S a zároveň $|s'| < |s|$.*

Kapitola 3

Výsledky a porovnanie

V tejto kapitole sa pozrieme na výkonnosť nášho riešenia v porovnaní s predošlou prácou v tejto oblasti.

Záver

Na záver už len odporúčania k samotnej kapitole Záver v bakalárskej práci podľa smernice [**?**]: „V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom. Rozsah záveru je minimálne dve strany. Záver ako kapitola sa nečísluje.“

Všimnite si správne písanie slovenských úvodzoviek okolo predchádzajúceho citátu, ktoré sme dosiahli príkazmi `\glqq` a `\grqq`.

Literatúra

[1] Vladimír Boža, Jakub Jursa, Broňa Brejová, and Tomáš Vinař. Fishing in read collections: Memory efficient indexing for sequence assembly. In *String Processing and Information Retrieval*, pages 188–198. Springer, 2015.

[2] John Gallant, David Maier, and James Astorer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50–58, 1980.