Sufixové stromy

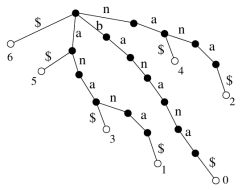
©kuko 18.11.2020

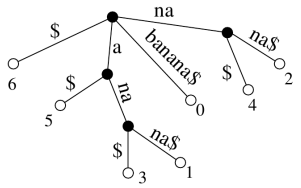
1 Motivácia

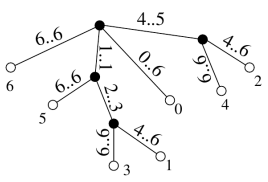
- text T dĺžky n, vzorka (pattern) P dĺžky m; napr. $T =$ ľudská DNA $\approx 3G$ znakov; chceme vyhľadávať P dĺžky 1k–100k
- triviálne vyhľadávanie: $O(m \times n)$; KMP: $O(m)$ predspracovanie vzorky a $O(n)$ vyhľadávanie; praktické algoritmy (Boyer-Moore)
- *indexovanie*: vedeli by sme predspracovať text (namiesto vzorky) a následne vyhľadávať v čase $O(m)$?
- najdlhší spoločný podreťazec – veľa rokov otvorený problém, dá sa v $O(n)$
- úloha: dané texty $T_1, \ldots, T_d$; chceme ich predspracovať tak, aby sme vedeli pre daný reťazec P nájsť všetky texty, ktoré začínajú na P
  - triviálne: bez predspracovania v $O(m \times d)$ (porovnávaním so všetkými textami)
  - zotriedime, následne vyhľadávame binárne: $O(m \times \log d)$
  - písmenkový strom, následne zídeme po ceste P, listy podstromu sú všetky dokumenty začínajúce na P
- vo všeobecnosti je písmenkový strom dobrý na úlohy o prefixoch

2 Sufixové stromy

- sufixový strom – na prvé počutie šialená myšlienka: všetky sufixy textu T vložíme do písmenkového stromu
- každý podreťazec je prefix nejakého sufixu (let that sink in)





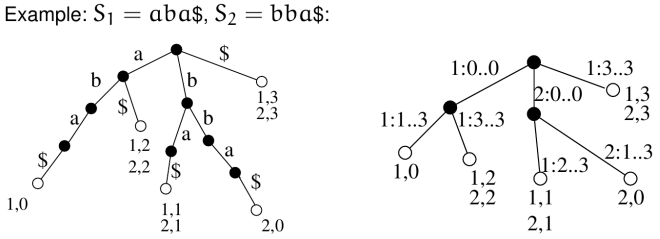


- každý vrchol v zodpovedá podreťazcu na ceste z koreňa do v
- výskyty tohto podreťazca sú listy v podstrome v

- pamäť: všetky sufixy majú spolu dĺžku $\Omega(n^2)$, čo je problém (obr. vľavo)
- riešenie:
- 1) každú cestu, ktorá sa nedelí skontrahujeme na jednu hranu (obr. v strede)
- ostane nám strom, kde má každý vnútorný vrchol stupeň ≥ 2, takže ich počet je $<$ počet listov; spolu bude teda vrcholov $O(n)$
- 2) pre každú hranu uložíme iba indexy do pôvodného textu T, nekopírujeme podreťazce (obr. vpravo)
- takto zaberá každá hrana $O(1)$ pamäte; spolu $O(n)$
- sufixový strom sa dá zostrojiť v čase $O(n)$, konštrukciu vysvetlíme neskôr

- zovšeobecnenie: pre množinu viacero "dokumentov"(textov) $\mathcal{D} = \{T_1, T_2, \ldots, T_d\}$ sufixový strom obsahuje všetky sufixy všetkých dokumentov
- napr. wikipédia $\approx 6M$ článkov, desiatky GB dát

- stačí zostrojiť sufixový strom pre $T_1\#T_2\#T_3\#\cdots T_d\#\$$, kde # a \$ sú dva špeciálne ukončovacie znaky, ktoré sa *nenachádzajú* v textoch $T_1, \ldots, T_d$
- jediný drobný rozdiel je, že listy a hrany musia špecifikovať, o ktorý dokument sa jedná



3 Aplikácie

- nachádza sa P v texte T? nájdi prvý výskyt/všetky výskyty
  - stačí zísť z koreňa pozdĺž cesty P; listy v danom podstrome sú všetky výskyty
  - ak chceme prvý výskyt, predpočítame si pre každý vrchol pointer na list s najmenším číslom sufixu (alebo priamo pozíciu prvého výskytu: prechodom stromu v $O(n)$ zdola nahor (postorder), vo vrchole bude minimum z jeho synov)
  - ak chceme všetky výskyty, stačí prehľadať celý podstrom – ak je k výskytov, podstrom má veľkosť $O(k)$
  - predpočítanie: $O(n)$, prvý výskyt: $O(m)$, všetky výskyty: $O(m+k)$, kde $k = \#$výskytov
- najdlhší opakujúci sa podreťazec
  - vrchol, pod ktorým sú aspoň dva listy reprezentuje reťazec, ktorý sa opakuje (to sú všetky vnútorné vrcholy; #listov = #výskytov)
  - pre každý vrchol môžeme predpočítať "string-depth(v)- počet znakov na ceste od koreňa do v (pozor, to nie je klasická hĺbka vrcholu – nechceme počet hrán, ale dĺžku textu na hranách)
  - výsledok je vnútorný vrchol s maximálnym string-depth – vieme nájsť v $O(n)$
- najdlhší spoločný podreťazec dvoch reťazcov (*longest common substring*)
  - ofarbime listy dvoma farbami – podľa toho, do ktorého reťazca patrí daný sufix
  - potom hľadáme vrchol, ktorý má pod sebou listy oboch farieb (predpočítame prechodom zdola nahor, či má vrchol pod sebou iba listy jednej farby (a ktorej) alebo oboch)
- najkratší podreťazec, ktorý sa vyskytuje len raz / najčastejšie sa vyskytujúci reťazec dĺžky aspoň k – podobne
- maximálne repeaty (chceme také podreťazce, že $T[i \ldots i+k] = T[j \ldots j+k]$, ktoré sa nedajú predĺžiť ani doľava ani doprava: $T[i-1] \neq T[j-1]$ a $T[i+k+1] \neq T[j+k+1]$)
  - stačí si pre každý list zodpovedajúci i-temu sufixu poznačiť znak pred ním (t.j. $T[i-1]$)
- najdlhší spoločný prefix $T[i \ldots]$ a $T[j \ldots]$ (a.k.a. LCP – *longest common prefix*)
  - triviálne v čase $O(k)$, ak $T[i \ldots i+k-1] = T[j \ldots j+k-1]$ ale $T[i+k] \neq T[j+k]$
  - v čase $O(1)$, ak si predpočítame LCA
- približné výskyty s $\leq k$ chybami
  - triviálne v $O(n \times m)$ (priložíme pattern na každú pozíciu a spočítame chyby)
  - lepšie: v $O(n \times k)$ (vytvoríme sufixový strom pre T a P a porovnávanie zrýchlime počítaním najdlhšieho spoločného prefixu: priložíme pattern na každú pozíciu, ako v predchádzajúcom riešení, ale namiesto porovnávania po znakoch sa vieme v $O(1)$ posunúť na najbližšiu chybu)
- počet dokumentov obsahujúcich P (*document counting problem*)
  - predstavme si, že listy zafarbíme rôznymi farbami, podľa toho, v ktorom dokumente je daný sufix; máme d farieb a chceme pre každý vrchol vedieť, koľko rôznych farieb je pod ním

  - triviálne v $O(m+k)$ prehľadaním celého podstromu (bez predpočítania; k = #výskytov) – dá sa lepšie?
  - pre každý vrchol predpočítame množinu farieb pod ním – problém: čas a pamäť $O(n \times d)$
  - lepšie: využijeme LCA; finta: fixnime nejaký konkrétny podstrom; dva vrcholy sú v podstrome práve vtedy, keď je aj ich LCA je v tomto podstrome
  - takže ak je v nejakom podstrome napr. r červených listov, tak $r-1$ po sebe idúcich dvojíc bude mať LCA v danom podstrome
  - namiesto počtu rôznych farieb budeme počítať počet všetkých listov mínus koľkokrát sa nejaká farba opakuje (pričom počet opakovaní = počet vrcholov jednej farby mínus 1)
  - majme pre každú farbu zoradené listy tejto farby zľava doprava
  - postupne pre každú farbu prejdeme listy danej farby; pre každé dva po sebe idúce listy spočítame LCA a započítame, že od tohto vrcholu vyššie máme 1 opakovanie
  - následne rekurzívne pre každý vrchol sčítame tieto hodnoty v podstrome a výsledok odčítame od počtu listov
  - takto v $O(n)$ predpočítame výsledok pre všetky vrcholy
- dokumenty obsahujúce P (*document listing problem*)
  - triviálne v $O(m+k)$ prehľadaním celého podstromu (bez predpočítania) – dá sa lepšie? čo ak je dokumentov veľmi veľa?
  - definujme pole A, kde $A[i]$ =číslo predošlého vrcholu *rovnakej farby*
  - výskyty patternu P budú listy nejakého podstromu, čo zodpovedá intervalu v poli A (povedzme $A[i \ldots j]$)
  - chceme vypísať všetky farby v podstrome; spravíme to tak, že nájdeme tie najľavejšie vrcholy v intervale z každej farby; to sú také, že predchodca rovnakej farby je mimo intervalu $[i \ldots j]$, presnejšie $< i$
  - úlohu teda redukujeme na problém vypísať všetky pozície $i \leq k \leq j$ také, že $A[k] < i$
  - toto vieme v čase úmernom počtu takých pozícií (t.j. v $O(\#\text{dokumentov})$ namiesto $O(\#\text{výskytov})$)
  - predpočítame RMQ na poli A; potom pre daný interval $[i, j]$ nájdeme minimum a kým je $< i$, rekurzívne sa zanoríme vľavo a vpravo