Prioritné fronty

Operácie: Insert(key[,val]), ExtractMin or ExtractMax

Implementácia	Insert	ExtractMin	Poznámka
Netriedené pole	$\Theta(1)$	$\Theta(n)$	
Triedené pole	$\Theta(n)$	$\Theta(1)$	
Počítadlá	$\Theta(1)$	$\Theta(1)$	malá doména (napr. $[1, 1000]$)
Halda	$\Theta(\log n)$	$\Theta(\log n)$	

V nadväzujúcich predmetoch:

Fibonacciho halda	$\Theta(1)$	$O(\log n)$	amortizovaná

Slovníky

Operácie: Insert(key,val), Search(key), Delete(key)

Porovnávanie kľúčov pomocou rovnosti:

Metóda	Insert	Search	Delete	Typ analýzy
Netriedený sp. zoznam	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	najhorší príp.
Hašovanie s reťazením	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	najhorší príp.
(naplnenosť α)	$\Theta(1)$	$\Theta(1+\alpha)$	$\Theta(1+\alpha)$	očakávaná zlož.
Hašovanie s otvorenou	$\Theta(n)$	$\Theta(n)$	N/A	najhorší prípad
adres. ($\alpha < 1$)	$\Theta(\frac{1}{1-\alpha})$	$\Theta(\frac{1}{\alpha}\ln\frac{1}{1-\alpha})$	N/A	očakávaná zlož.
		alebo $\Theta(\frac{1}{1-\alpha})$		

Porovnávanie kľúčov $=, <, >$:

Metóda	Insert	Search	Delete	Typ analýzy
Triedené pole	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	najhorší prípad
Binárny vyhľ. stromy	$\Theta(n)$ $\Theta(\log n)$	$\Theta(n)$ $\Theta(\log n)$	$\Theta(n)$ $\Theta(\log n)$	najhorší prípad priemerný príp.
AVL stromy	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	najhorší prípad
Scapegoat stromy	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	amortizovaná
Trie (reťazec dĺžky k)	$\Theta(k)$	$\Theta(k)$	$\Theta(k)$	najhorší príp.

Disjunktné množiny

Operácie: MakeSet(x), Union(x,y), FindSet(x)

Metóda	MakeSet	Union	FindSet	Typ analýzy
Stromy	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$	amortizovaná
. . . so skracovaním	$\Theta(1)$	$\Theta(\alpha(n))$	$\Theta(\alpha(n))$	amortizovaná

Reprezentácia grafov

Operácia	Matica susedností	Zoznamy susedov
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\text{outdeg}(u))$
Zoznam hrán z u	$\Theta(n)$	$\Theta(\text{outdeg}(u))$
Pamäť	$\Theta(n^2)$	$\Theta(m + n)$

Základné grafové algoritmy:

– Prehľadávanie (do šírky, do hĺbky)

– Najkratšie cesty (Floyd-Warshall, Dijkstra)

– Najlacnejšie kostry (Kruskal, Prim)

Pokročilé triediace algoritmy

Algoritmus	Časová zlož.	Analýza
Merge Sort	$\Theta(n \log n)$	najhorší príp.
Heap Sort	$\Theta(n \log n)$	najhorší príp.
Znáhodnený Quick Sort	$\Theta(n^2)$ $\Theta(n \log n)$	najhorší príp. očakávaná zlož.
Counting Sort čísla z $[0, k]$	$\Theta(n + k)$	najhorší príp.
Radix Sort d-ciferné čísla	$\Theta(dn)$	najhorší príp.

Vyhľadávanie v texte

Algoritmus	Predspracovanie	Vyhľadanie	
Naivný algoritmus		$O(mn)$	najhorší príp.
Rabin-Karp		$O(m+n+\frac{mn}{q})$ q =veľkosť domény hašovacej funkcie	očakávaná zlož.
Konečný automat	$O(m^3\|\Sigma\|)$	$O(n)$	najhorší príp.
Knuth-Morris-Pratt	$O(m)$	$O(n)$	najhorší príp.
Sufixové stromy	$O(n)$	$O(m+k)$ $k=$ počet výskytov	najhorší príp.

Kompresia textu

LZW kompresie	Huffmanovo kódovanieg
nahraď tokeny variabilnej dĺžky kódovými slovami fixnej dĺžky	nahraď tokeny fixnerj dĺžky kódovými slovami variabilnej dĺžky
heuristika	optimálny kód
jednoprechodový algoritmus	dvojprechodový algoritmus
ukladá sa iba komprimovaný reťazec	ukladá sa kódovací strom aj komprimovaný reťazec
45% kompresia	65% kompresia
GIF, Unix compress, transmisia dát; niektoré varianty PDF, postscript, NTFS	zip, posledný krok v JPEG, MP3 kompresii

Greedy: "Vzor" dôkazu správnosti greedy algoritmu

Lema: Predpokladajme, že greedy algoritmus vráti riešenie G. Potom existuje optimálne riešenie, ktoré sa s riešením G zhoduje na prvých k voľbách.

Dôkaz: Matematickou indukciou podľa k.

Báza indukcie. Pre $k = 0$ – ľubovoľné optimálne riešenie.

Indukčný krok. (Prepokladajme, že sme neurobili chybu pri prvých k voľbách, potom aj $(k+1)$-vá voľba je OK.)

- Predpokladajme, že existuje optimálne riešenie OPT, ktoré sa zhoduje s G na prvých k voľbách.
- Vyrobíme riešenie OPT':
  - OPT' má rovnakú hodnotu ako OPT (a preto je tiež optimálne)
  - OPT' súhlasí s G na jednej ďalšej $(k+1)$-vej voľbe.

Dynamické programovanie

1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.** Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **Bázové podproblémy.** Ktoré políčka nemožno vypočítať pomocou vzťahov z predchádzajúceho kroku? Aké hodnoty by mali obsahovať?
4. **Vyberieme poradie vypĺňania.** V akom poradí musíme maticu vypĺňať tak, aby sme v každom kroku mali vypočítané všetky políčka, ktoré potrebujeme na výpočet daného políčka?

Rozdeľuj a panuj: Master theorem

Nech $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$. Nech $k = \log_b a$. Potom:

1. Ak $\boxed{f(n) \in O(n^{k-\varepsilon})}$ pre niektoré $\varepsilon > 0$, potom $\boxed{T(n) \in \Theta(n^k)}$.
2. Ak $\boxed{f(n) \in \Theta(n^k)}$, potom $\boxed{T(n) \in \Theta(f(n) \log n)}$.
3. Ak $\boxed{f(n) \in \Omega(n^{k+\varepsilon})}$ pre niektoré $\varepsilon > 0$ a platí podmienka regularity, potom $\boxed{T(n) \in \Theta(f(n))}$.

Podmienka regularity: Existuje $c < 1$ také, že pre všetky dostatočne veľké n platí $af(n/b) \leq cf(n)$.

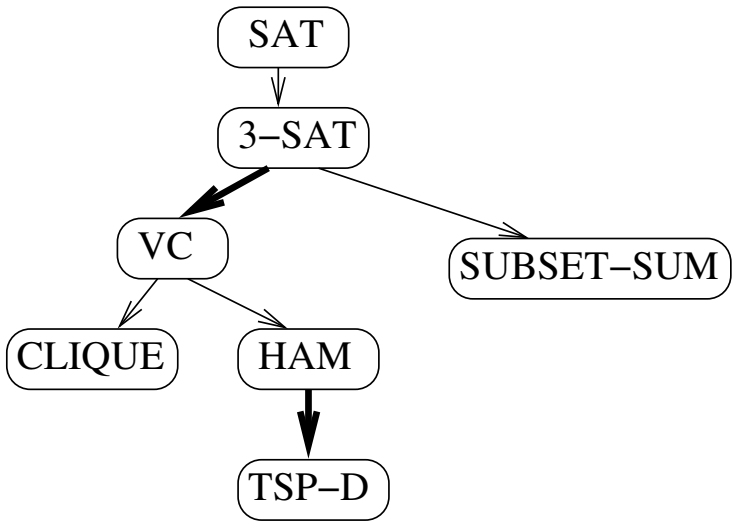
Poznámka: Veta platí aj v prípade rozumných usporiadaní dolných a horných celých častí - viď napr. CLRS2 4.4.2.

Nedeterministický algoritmus pre riešenie TSP-D

```
function TSP-D
  visited[i]:=false for all vertices;
  last_visited:=1; visited[1]:=true;
  length:=0;
  repeat n-1 times
    choose next_visited between 1 and n;
    if visited[next_visited] then reject;
    //we cannot visit a single vertex twice
    visited[next_visited]:=true;
    length:=length+w(last_visited,next_visited);
    last_visited:=next_visited;
  length:=length+w(last_visited,1);
  if length<=B then accept;
               else reject;
```

Ako dokázať, že problém Q je NP-ťažký?

1. Vyberme si problém N o ktorom už vieme, že je NP-úplný
2. Ukážeme $N \leq_P Q$:
   - Navrhneme polynomiálny algoritmus, ktorý prerobí vstup x pre problém N na vstup $f(x)$ pre problém Q.
   - Dokážeme: Ak je x pozitívny vstup pre N, potom
     $f(x)$ je pozitívny vstup pre Q
   - Dokážeme: Ak je x negatívny vstup pre N, potom
     $f(x)$ je negatívny vstup pre Q
     —ALEBO—
     Ak $f(x)$ je pozitívny vstup pre Q, potom
     x je pozitívny vstup pre N
3. Keďže N je NP-úplný, Q musí byť NP-ťažký.



A ⟶ B means reduction A to B

➡ means reduction shown already

Nadväzujúce predmety

Bakalárske štúdium:

- 1-AIN-211: Úvod do teoretickej informatiky
- 1-AIN-412: Grafy, grafové algoritmy a optimalizácia
- 2-INF-174: Teória grafov
- 1-INF-167: Výpočtová zložitosť a vypočítateľnosť

Magisterské štúdium:

- 2-AIN-205: Algoritmické riešenie ťažkých problémov
- 2-INF-237: Vybrané partie z dátových štruktúr
- 2-INF-231: Efektívne paralelné algorimy