PŘIHRÁDKOVÉ TŘÍDENÍ

Binsort, Bucketsort, Radixsort

nepoužívají porovnávání prvků

třídí v lineárním čase

netřídí na místě

1) třídím permutaci $\{1, \dots, n\}$

A... původní pole

B... výsledné pole

for $i := 1$ **to** n **do**

$B(A(i)) := A(i)$

enddo

složitost: čas $O(n)$

prostor $O(n)$

2) třídím n navzájem různých čísel (celých):

vypočtu min a max (porovnáváním, čas $O(n)$)

vytvořím pole $B[min..max]$

dosazuji $B(A(i)) := A(i)$

předosadím neprázdné položky z pole B do A

(nebo pole B zkomprimuji)

<u>složitost</u>:

$O(n)$, je-li max - min řádu $O(n)$

$O(n^2)$, — " — $O(n^2)$

může být i horší

Doporučení:

spočítat ještě druhou nejmenší a druhou největší hodnotu

3) třídím n čísel, která nabývají hodnot $1, \dots, m$

předpoklad: $m \leq n$

a) vytvořím m prázdných seznamů

b) **for** $i := 1$ **to** n **do**

vložím $A(i)$ do seznamu $k = A(i)$

enddo

c) spojím seznamy $1, \dots, m$

složitost: a) $O(m)$

b) $O(n)$

c) $O(m)$ pokud udržuji ukazatel na začátek a konec seznamu

celkem: $O(n+m) = O(n)$

Pro $m > n$ se to nevyplatí.

4) Třídění probíhá v cyklu přes jednotlivé cifry, nejméně významnou počínaje.

Příklad: 26, 9, 0, 251, 1, 49, 640, 16, 81, 4

1. fáze – podle jednotek:

0: 0, 640
1: 251, 1, 81
2:
3:
4: 4
5:
6: 26, 16
7:
8:
9: 9, 49

2. fáze – podle desítek:

0: 0, 1, 4, 9
1: 16
2: 26
3:
4: 640, 49
5: 251
6:
7:
8: 81
9:

3. fáze – pro stovky:

0: 0, 1, 4, 9, 16, 26, 49, 81
1:
2: 251
3:
4:
5:
6: 640
7:
8:
9:

Výsledek: 0, 1, 4, 9, 16, 26, 49, 81, 251, 640

Po 1. fázi je posloupnost setříděná podle posledních cifer

Po 2. ——— " ——— posledních 2 cifer

Po 3. ——— " — podle všech

Kratší čísla se dají postupně vynechávat.

<u>Důležité</u>: prvky přidávat na konec seznamu

v každé fázi rozdělovat posloupnost přesně v tom pořadí, do kterého ji uspořádala fáze předchozí

<u>Složitost</u>: $O(k\cdot(n+m))$

k = maximální počet cifer, m = rozsah hodnot cifer

$O(n)$ pokud chápeme k a m jako konstanty

$O(n\log n)$ když třídíme permutaci čísel $\{1, \dots, n\}$ v binárním zápise

$(m = 0,1;\ k = \log n)$

HYBRIDSORT

Meijer, Akl (1980)

kombinuje prvky přihrádkového a porovnávacího třídění

Příklad: třídím maximálně trojciferná čísla rovnoměrně rozdělená

rozdělím je do přihrádek po stovkách

$\lfloor \frac{x_i}{100} \rfloor$ ... číslo přihrádky pro x_i

každou přihrádku setřídím – Quicksortem, Heapsortem ...

spojím setříděné přihrádky

Složitost: $O(m)$ vytvoření přihrádek (předp. konstantní)

$O(b_i^2)$ nebo $O(b_i \log b_i)$ setřídění přihrádky (b_i prvků)

$O(m)$ spojení přihrádek

celkem: $O(m + \sum_{i=1}^{m} b_i^2)$ nebo $O(m + \sum_{i=1}^{m} b_i \log b_i)$

tj. $O(n^2)$ nebo $O(n \log n)$

průměrný případ: $O(n \log n)$

může být $O(n)$ při vhodně volených přihrádkách

Předpoklady:

třídíme reálná čísla z $\langle 0,1 \rangle$

rovnoměrně rozdělená

rozdělíme je do k bucketů, kde $k = \alpha n$, $0 < \alpha < 1$

x_i přijde do bucketu $\lceil x_i \cdot k \rceil$

Příklad: 0,1; 0,85; 0,6; 0,34; 0,2; 0,3; 0,15;

0,42; 0,25; 0,7; 0,11; 0,35; 0,17; 0,29; 0,66;

0,9; 0,37; 0,19; 0,87; 0,5

$n = 20$, $\alpha = 0,2$, $k = 4$

buckety:

1: 0,1; 0,2; 0,15; 0,25; 0,11; 0,17; 0,19

2: 0,34; 0,3; 0,42; 0,35; 0,29; 0,37; 0,5

3: 0,6; 0,7; 0,66

4: 0,85; 0,9; 0,87

Každý bucket setřídíme Heapsortem

buckety pospojujeme

Očekávaný čas

vytvoření bucketů : $O(k) = O(dn)$
rozdělení prvků : $O(n)$
spojení bucketů : $O(k) = O(dn)$

$\Big\}\ O(n)$

třídění : $E\left(\sum_{j=1}^{k} b_j \log b_j\right)$

$$P(b_j = i) = \binom{n}{i}\left(\frac{1}{k}\right)^i\left(1-\frac{1}{k}\right)^{n-i} \quad \text{pro všechna } j$$

$$E\left(\sum_{j=1}^{k} b_j \log b_j\right) = \sum_{j=1}^{k} E\, b_j \log b_j = \sum_{j=1}^{k}\sum_{i=2}^{n} i \log i\, P(b_j = i) =$$

$$= k \sum_{i=2}^{n} i \log i \binom{n}{i}\left(\frac{1}{k}\right)^i\left(1-\frac{1}{k}\right)^{n-i} \le$$

$$(*) \quad \le k \sum_{i=1}^{n} i^2 \binom{n}{i}\left(\frac{1}{k}\right)^i\left(1-\frac{1}{k}\right)^{n-i} =$$

$$= k\left(\frac{n(n-1)}{k^2} + \frac{n}{k}\right) = \frac{n(n-1)}{k} + n =$$

$$= \frac{n(n-1)}{dn} + n = \frac{n-1}{d} + n = O(n)$$

(*) $\Rightarrow$ k třídění bucketů lze použít algoritmus s kvadratickou složitostí

$$\sum_{i=1}^{n} i^2 \binom{n}{i} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-i} = \sum_{i=1}^{n} i(i-1) \binom{n}{i} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-i}$$

$$+ \sum_{i=1}^{n} i \binom{n}{i} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-i}$$

2. součet $= \displaystyle\sum_{i=1}^{n} i \frac{n!}{i!(n-i)!} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-i} =$

$$= \frac{n}{k} \sum_{i=1}^{n} \frac{(n-1)!}{(i-1)!(n-i)!} \left(\frac{1}{k}\right)^{i-1} \left(1-\frac{1}{k}\right)^{n-i} =$$

$$= \frac{n}{k} \sum_{i=0}^{n-1} \binom{n-1}{i} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-1-i} =$$

$$= \frac{n}{k} \left(\frac{1}{k} + 1 - \frac{1}{k}\right)^{n-1} = \frac{n}{k}$$

1. součet $= \displaystyle\frac{n(n-1)}{k^2} \sum_{i=0}^{n-2} \binom{n-2}{i} \left(\frac{1}{k}\right)^i \left(1-\frac{1}{k}\right)^{n-2-i} =$

$$= \frac{n(n-1)}{k^2}$$

Modifikace pro nerovnoměrná rozdělení:

Platí: Když náhodná veličina X má distribuční funkci F, pak $Y = F(X)$ má rozdělení $R(0,1)$.

Dk: $P(Y < y) = P(F(X) < y) = P(X < F^{-1}(y)) =$

$$= F(F^{-1}(y)) = y$$

x_i přijde do bucketu $\lceil F(x_i) \cdot k \rceil$

Nefunguje to pro úplně všechna rozdělení:

podmínky: rozdělení musí být soustředěno na konečném intervalu

(tj: $P(|X| > M) = 0$ pro nějaké M)

$F(x_i)$ se musí dát rychle spočítat

DISTRIBUTIVE PARTITIONING

Dobosiewicz (1978)

třídím n prvků (ne nutně permutace $\{1, \dots, n\}$)

najdu: min, max, medián – čas $O(n)$

$\langle$min, medián$\rangle$ rozdělím na $\frac{n}{2}$ stejně dlouhých intervalů

$\langle$medián, max$\rangle$ — " —

pro $x \leq$ medián: x přijde do intervalu

$$j = \left\lfloor \frac{x-\min}{\text{med}-\min} \cdot \frac{n-2}{2} + 1 \right\rfloor$$

pro $x >$ medián: $j = \left\lceil \frac{x-\text{med}}{\max-\text{med}} \cdot \frac{n-1}{2} + \frac{n+1}{2} \right\rceil$

spočtu prvky v každém intervalu, pokud je jich víc než 1, postup se pro tento interval opakuje

<u>časová složitost</u>

nejhorší případ: $O(n \log n)$

nejhorší případ: v každé fázi $\frac{n}{2}$ prvků padne do jednoho bucketu

$$T(n) = cn + 2T\left(\frac{n}{2}\right)$$

$$T(n) = O(n \log n)$$

průměrný případ:

$$T(n) = cn + n\sum_{i=1}^{n} P_i T(i)$$

$P_i = P(\text{bucket obsahuje } i \text{ prvků}) =$

$$= \binom{n}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{n-i}$$

$\sum_i P_i T(i) =$ očekávaný čas na setřídění bucketu

řešení: $T(n) = O(n)$

ověření: dosadím $T(i) = i$

$$T(n) = cn + n\underbrace{\sum_{i=1}^{n} i\binom{n}{i}\left(\frac{1}{n}\right)^i\left(1-\frac{1}{n}\right)^{n-i}}_{n\cdot\frac{1}{n}} = O(n)$$

LINEAR PROBING SORT

Gonnet-Munro (1981), Dobosiewicz (1991)

třídím n-prvkové pole A

vytvořím pole B délky $m > n$

$\forall x \in A$ spočtu pomocí nějaké interpolační fce
jeho přibližnou polohu v poli B

je-li toto místo v poli B volné, dosadím tam x

není-li volné (je tam y):

pokud $x < y$, dosadím x a y posunu doprava –
tak dlouho, až se narazí na volné místo

($x > y$ symetricky)

je to podobné hašování s lineárním přidáváním
(včetně analýzy)

poslední krok: komprese pole B

Problémy

1) volba m (velikost pole B)

2) volba interpolační fce

3) kolik se provede porovnání, než se prvek definitivně umístí

4) pole B může přetéct

Řešení

2) za předpokladu rovnoměrného rozdělení lineární interpolace, tj. $j = 1 + \lfloor m \frac{x - min}{max - min} \rfloor$

1) m se volí v závislosti na n a na tom, čeho chci dosáhnout:

minimalizovat počet dodatečných porovnání,

— " — paměťovou náročnost,

atd.

při rovnoměrném rozdělení platí:

a) pro $\frac{n}{m} < 0{,}8$ je pravděpodobnost přetečení o 36 prvků (a více) menší než 10^{-7}

b) pro $\frac{n}{m} = 2-\sqrt{2} \doteq 0{,}5857$ je očekávaný počet

operací (přístupů, porovnání, dosazení)

roven $(2+\sqrt{2})n + O(1)$

c) součin prostor × čas je minimální pro

$\frac{n}{m} = \sqrt{3}-1 \doteq 0{,}732$

jeho očekávaná hodnota je $\frac{3\sqrt{3}+5}{2} n^2 + O(n) \doteq$

$\doteq 5{,}098 n^2 + O(n)$

d) doporučení: $m = \lfloor 1{,}25 n \rfloor$

4) pole B se navíc rozšíří o 40 pozic

3) výsledky jsou podobné jako v hašování s

lineárním přidáváním (liší se jen o konstantu)

Časová složitost

nejhorší případ $O(n^2)$

průměrný případ $O(n)$

GROUPSORT - Bucketsort na místě

Agarwal a spol. (1997)

Algoritmus:

1) obor hodnot min - max se rozdělí na k stejně dlouhých intervalů

2) pole se jednou projde

podle hodnot prvků se napočítá, kolik jich bude patřit do každého buckety

výsledky se uloží do pomocného pole velikosti k

3) prvky se přemístí do svých bucketů

buckety jsou přímo v tříděném poli

hranice jsou uloženy v pomocném poli

celkový čas na přemístění - lineární

4) každý bucket se setřídí Quicksortem

Počet bucketů:

konstantní – třídí se na místě
čas není lineární (ani očekávaný)

$k = \alpha n$, $0 < \alpha < 1$ – za předpokladu rovnoměrného rozdělení bude očekávaný čas $O(n)$
prostorová složitost není konstantní

Implementace bodu 2):

do pole k uložím navíc průběžnou pozici v jednotlivých bucketech

pole začnu procházet odleva:

posouvám průběžnou pozici v 1. bucketu, dokud nenarazím na prvek, který tam nepatří

patří-li tento prvek do j-tého bucketu, začnu prohledávat j-tý bucket (od průběžné pozice), dokud nenarazím na prvek, který tam nepatří (patří do bucketu k)

dosadím na jeho místo prvek z 1. bucketu

prohledávám bucket k

atd.

Příklad: 75, 32, 199, 98, 1, 125, 137, 88, 170, 25, 165, 188

obor hodnot: 1 - 199

rozdělím na 4 intervaly:

1. 1 - 49 velikost: 3
2. 50 - 99 3
3. 100 - 149 2
4. 150 - 199 4

označím: zdola konce bucketů

shora průběžné pozice

75, 32, 199, 98, 1, 125, 137, 88, 170, 25, 165, 188

zleva: 75 nepatří do bucketu 1, patří do 2

98 patří do 2

1 nepatří do 2 (patří do 1) => 75 půjde na místo 1

zařadím 1 - půjde na místo 1

1, 32, 199, 98, 75, 125, 137, 88, 170, 25, 165, 188 pokračuji od 32 - patří do bucketu 1

199 - nepatří (patří do 4)

170 patří do 4

25 nepatří do 4 (patří do 1) =>

199 půjde na místo 25

zařadím 25 - půjde na místo 199

1, 32, 25, 98, 75, 125, 137, 88, 170, 199, 165, 188

průběžná pozice v 1. bucketu přelezla hranici

pokračuje se od 125 - nepatří do 2 (patří do 3)

137 patří do 3

88 nepatří do 3 (patří do 2) => 125 půjde na místo 88

zařadím 88 - půjde na místo 125

1, 32, 25, 98, 75, 88, 137, 125, 170, 199, 165, 188

doprohlédnu poslední bucket

celková složitost - lineární