Drzewa Van Emde Bossa

wtorek, 11 czerwca 2024 17:50

Uporządkowane uniwersum U

$U = \{0, \dots, U-1\}$ ← małe liczby naturalne

Operacje:

- insert
- delete
- successor ⇒ następnik
- predecessor ⇒ poprzednik

Chcemy pamiętać $S \subseteq U$

Chcemy aby złożoność tych operacji była mniejsza niż $\log U$
(bo tak można uzyskać w drzewach zbalansowanych)

Dążymy do $O(\log\log n)$!

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

$$T(n) = \begin{cases} T(\sqrt{n}) + \Theta(1) \\ O(1) & \text{dla małych } n \end{cases}$$

log ⇒ zakładamy dane 2^k

loglog ⇒ 2^{2^k}

$\sqrt{2^{2^k}} = 2^{2^{k-1}}$, bo $2^{2^{k-1}} \cdot 2^{2^{k-1}} = 2^{2^1 \cdot (2^{k-1})} = 2^{2^k}$

$n = T(2^{2^k}) = T(2^{2^{k-1}}) + c$

$T(2^{2^{k-r}}) + r \cdot c$, ale k to $\log\log n$

⇝ czyli to będzie $\log\log n \cdot c$

Podejście naiwne:

- w tablicy przechowujemy następnik

Przykład $\{1, 3, 7\}$

1	3	3	7	7	7	7	..		
0	1	2	3	4	5	6	7	....	u-1

Odczyt $O(1)$, ale updaty kosztowały by nas $O(n)$ niesc mamie

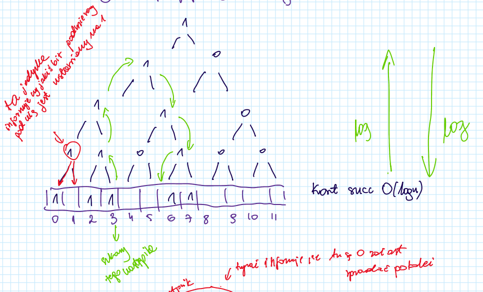
- wektor charakterystyczny

Przykład $\{2, 3, 6, 8, 9\}$

0	0	1	1	0	0	1	0	1	1	...	
0	1	2	3	4	5	6	7	8	9	...	u-1

Odczyt $O(1)$, ale następnik (successor) kłopotliwy, bo musimy szukać pod kolejnymi adresami czy jest 1, $O(n)$ w najgorszym wypadku

Ale możemy to przyspieszyć stosując drzewo OR



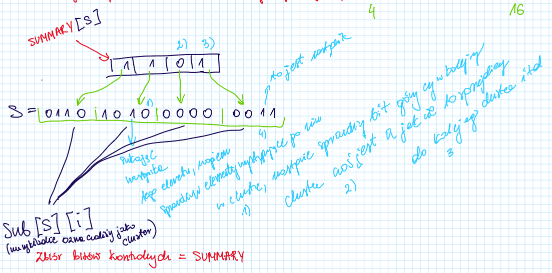
następnik ↙ tracimy informację że tu są 0 zatem sprawdzać potrzebujemy

|0 1 1 0|1 0 1 0|0 0 0 0|0 0 1 1|

- Pomysł: podzielić dane na fragmenty po $\sqrt{n}$ elementów dla n danych
4 16

Cyli sprawdzamy $\sqrt{n}$ elementów w klastrze
Sprawdzamy bity kontrolne których maksymalnie jest $\sqrt{n}$
i potem towarzyszące klastrze o $\sqrt{n}$ elementach by znaleźć następnik

Stąd $\sqrt{n} + \sqrt{n} + \sqrt{n} \to O(\sqrt{n})$



Zbiór bitów kontrolnych = SUMMARY

Kiedy zatem robimy insert, to oprócz wpisu w tablicy robimy update SUMMARY $O(1)$

W delete musimy jeszcze sprawdzić czy nie ma jakichś elementów w tym clustrze, by wiedzieć czy zmienić wartość w SUMMARY, zatem koszt $O(\sqrt{n})$

Ale to i tak za dużo, bo my chcemy loglog

- POMYSŁ JUŻ DOBRY:

$x = |0110|1101|$ $u = 8$

↑ high(x) ↑ low(x)

Czyli poprzednio mogliśmy opisać liczby 2 binarnie od 0 do 15

więc jak mamy liczbę $\underbrace{01}_{high}\underbrace{10}_{low} = 6$

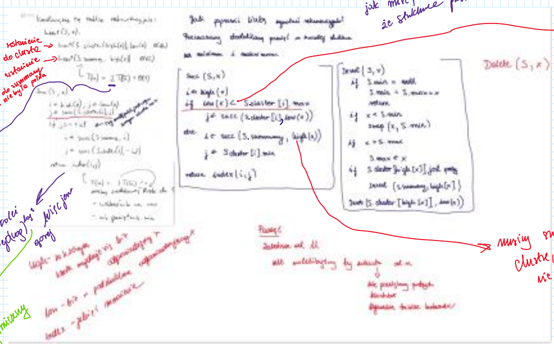
więc jak mamy liczbę $\underbrace{01}_{\text{high}}\underbrace{10}_{\text{low}} = 6$

high = 1 → mówi nam że liczba będzie w pierwszym cluster (numerujemy clustry od 0)

low = 2 → mówi nam że będzie na drugiej pozycji (numerujemy pozycje od 0)

czyli nie będziemy się ograniczać do jednej redukcji ⇒ będzie to rekurencyjne tablice w tablicy

jak min puste to wiemy że struktura pusta



wstawienie do cluster

wstawienie do summary o ile nie była pełna

tu 2 razy bo zależy log logu dolnego

tu z kolei dodajemy log? więc jeszcze gorzej

czas polilogarytmiczny

→ jest sens wchodzić w tą podstrukturę bo faktycznie tą wartość której szukamy następnik jest większe od maxa

→ musimy szukać w innym clusterze, bo w tym na pewno nie będzie następnika dla naszego elementu, bo nie jest mniejszy od maxa