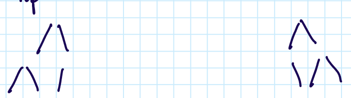
Kopiec

poniedziałek, 27 maja 2024 10:56

Postać drzewa binarnego - w każdym węźle zapisana jedna liczba.

WAŻNE 1! Niepełne kopce muszą mieć pierwsze węzły w lewo

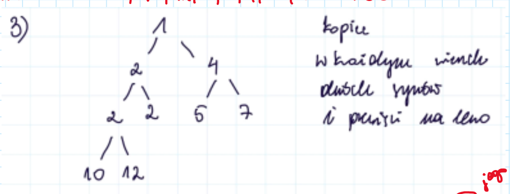
np: a nie ↙ to źle



WAŻNE 2! Wartości w ścieżce od liścia do korzenia tworzą jakiś ustalony ciąg (rosnący / malejący)

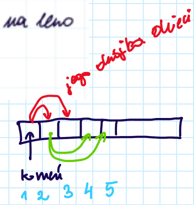
np: (ZADANIE EGZAMIN PRÓBNY):

NARYSUJ POPRAWNY KOPIEC ZAWIERAJĄCY ELEMENTY O NASTĘPUJĄCYCH WARTOŚCIACH: 5,2,7,12,2,1,4,10,2. KOPIEC POWINIEN MIEĆ NA SZCZYCIE EL. MINIMALNY



Zapamiętywanie kopca w tablicy T:

indeksujemy od 1



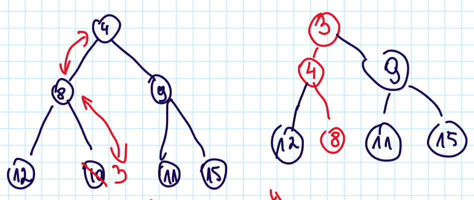
Synowie elementu T[i] są w T[2i] i T[2i+1]

np: T[1] ma synów w T[2] i T[3]

(T[1] – korzeń)

Ojciec elementu T[i] znajduje się w T[i/2] ← dzielenie całkowite

np T[3] ma ojca w T[1]



4	8	9	12	3	11	15
1	2	3	4	5	6	7

(K,5)

k ← 5

j ← 5

operacja "przesuń wyżej" wywołujemy 3.

```
procedure zmień-element (K[1..n], i, u)
    x ← K[i]
    K[i] ← u
    if u < x  then przesuń-niżej (K, i)
              else przesuń-wyżej (K, i)

procedure przesuń-niżej (K[1..n], i)
    k ← i
    repeat
        j ← k
        if 2j ≤ n  and K[2j] > K[k]  then k ← 2j
        if 2j < n  and K[2j + 1] > K[k]  then k ← 2j + 1
        K[j] ↔ K[k]
    until j = k

procedure przesuń-wyżej (K[1..n], i)
    k ← i
    repeat
        j ← k
        if j > 1  and K[j div 2] < K[k]  then k ← j div 2
        K[j] ↔ K[k]
    until j = k
```

szukanie większego z synów by dokonać zamiany

dlatego musimy zrobić dwa porównania bo musimy wziąć z 2 większych synów

złożoność log n

przesuwanie wyżej kosztowne o jedno porównanie mniej

USTALAMY, ŻE W KORZENIU JEST WARTOŚĆ MAKSYMALNA

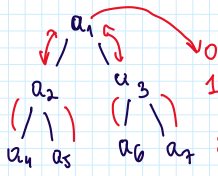
BUDOWA KOPCA

1) możemy budować poprzez sortowanie elementów, ale czas to $n \log n$

2) TROCHĘ LEPIEJ:

wstawiamy pojedyncze elementy i sprawdzamy czy relacja jest zachowana, jak nie to "przesuń wyżej"

KOSZT:



KOSZT jako liczba porównań

0

1 dla 2 wierzchołków

2 dla 4 wierzchołków

3 -||- 8 -||-

PRZYPADEK OGÓLNY:

liczba węzłów: $n = 2^k - 1$

liście: $\lceil \frac{n}{2} \rceil$

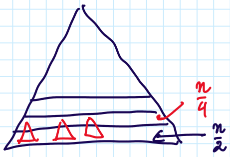
Koszt wstawienia SAMYCH liści: $\lceil \frac{n}{2} \rceil \cdot \log n = \Omega(n \log n)$

3) LEPIEJ

$a_1, \dots, a_n$

n-wierzchołków

$\lceil \frac{n}{2} \rceil$ liści



same liście będą w dobrej relacji kopcowej; później na poziomie $\frac{n}{2}$ trzeba będzie tylko uporządkować koniec z jego dziećmi itd

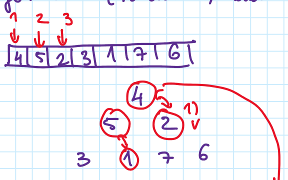
bo przesuń niżej dla każdego będzie wymagało 2 porównań; bo o 2 poziomy; bo o 3 poziomy

$$\frac{n}{2} \cdot 0 + \frac{n}{4} \cdot 2 + \frac{n}{8} \cdot 4 + \frac{n}{16} \cdot 6 + \dots + 1 \cdot 2 \log n =$$

$$= \frac{n}{4} \cdot 2 + \frac{n}{8} \cdot 4 + \frac{n}{16} \cdot 6 + \dots + 1 \cdot 2 \log n = \Theta(n)$$

Czyli

```
procedure buduj-kopiec (K[1...n])
    for i ∈ (n div 2) downto 1  przesuń-niżej (K,i)
```

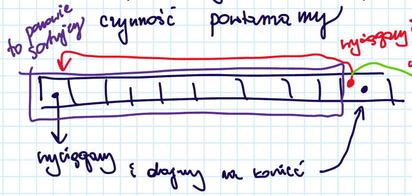


$7 : 2 = 3$

przesuń-niżej na 5. przesuń-niżej na 4

HEAPSORT ⇒ usuwamy element z końca, naprawiamy strukturę kopca i czynność powtarzamy

Bo wiemy, że w korzeniu jest element najmniejszy



ten element z końca był jednym z wszystkich liści, musimy go przenieść niżej aby zachować porządek

```
procedure heapsort (K[1...n])
    buduj-kopiec (K)
    for i ∈ n step -1 to 2 do → n razy
    K[1] <=> K[i]
```

for $i \in n$ step -1 to 2 do → n razy

$K[1] \Leftrightarrow K[i]$

przesuń-niżej $(K[1 \ldots i-1], 1)$ → $\log n$

return K

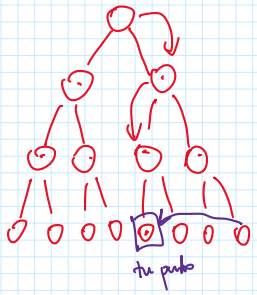
złożoność: $O(n \cdot \log n)$

2 części każdej z poziomów

PRZYSPIESZENIE HEAPSORTA

przerwany "duszę"

Wtedy przeniesie duszę mamy $1 \cdot \log n$



$1 \cdot \log n$ porównań duszy

$+ 1 \cdot \#$ poziomów podciągnięcie wstawianego

kandydata $2 \cdot \log n$

Czyli kopiec to taka bardzo kolejka priorytetowa:

- FIND_MAX, DELETE_MAX $O(1)$, bo to pierwsze wartości w tablicy (przy założeniu, że w końcówce wartości nieistotne)
- INSERT → zwiększamy tablicę o 1, ustawiamy na końcu tę wartość i przy użyciu "przesuń niżej" przesuwamy na odpowiednie miejsce → $O(\log n)$