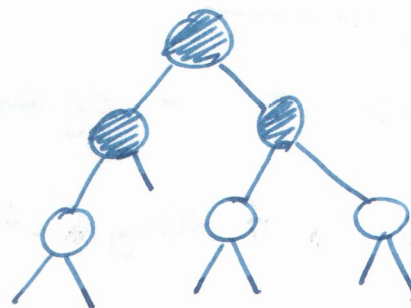
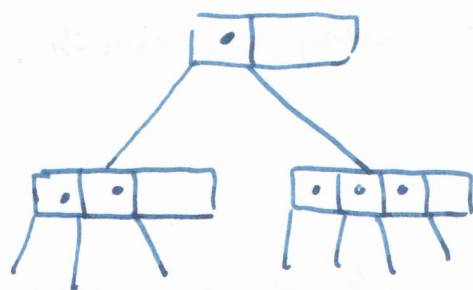


Przykład:



B-drewa były motywowane do drzew czerwonych. Teraz te drzewa ~~do~~ oparte z drzew wyglądają się regularne patrząc na B-drewo

ZŁĄCZALNE KOLEJKI PRIORYTETOWE

Będziemy mówić o kopcach dwumianowych w wersjach lewej i prawej, a potem przejdziemy do kopca Fibonacciego. Na razie kopce dwumianowe w wersji gorzej.

DRZEWIA DWUMIANOWE

B_0 : •

B_1 : |

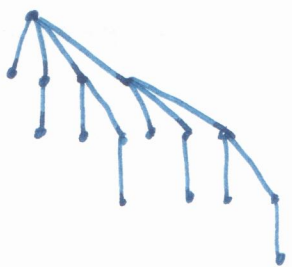
B_2 :



B_3 :



B_4 :



1
4
6
4
1

Wierchołty

B_{i+1} :



W kopcu dwumianowym mamy \log drzew dwumianowych. Wierchołty tych drzew pamiętają sobie w porządku kopcowym.
* Fakt: Drewo B_i ma 2^i wierchołtów.
(pamiętamy skomplikowany dowód indukcyjny).

Będziemy chcieli wykonywać następujące operacje na zgrupowanych kolejkach priorytetowych:

- insert
- min
- delete min
- meld

Termin: rząd danej # synów korzenia ($nsd(B_i) = i$)

Wersja eager kopca dwumianowego:

W kopcu nie ma dwóch drzew o tym samym rzędzie

H - wektor : 

$H[i] = \begin{cases} \text{wskaźnik na korzeń } B_i \text{ (o ile } B_i \text{ jest w kopcu)} \\ \text{NULL} \end{cases}$

MIN_H - wskaźnik na korzeń zawierający minimalny element

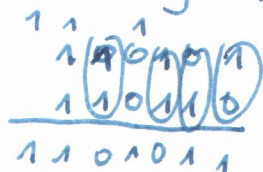
Operacje:

(.) makeheap(k) - tworzenie kopca z danej liczby k $\Theta(1)$

(..) insert(k, H) :

$H_i \leftarrow \text{makeheap}(k)$
 $H \leftarrow \text{meld}(H, H_i)$

(...) Mamy dwa wektory H i dodajemy binarnie :



Koszt $O(\log n)$, gdzie n to #liści drzew. H

W implementacji wykorzystujemy operacje join : łączymy dwa drzewa tych samych rzędów. Koszt $O(1)$

(•••) Deletemin

Niech B_i - drzewo zawierające min

Usuwamy koniec z B_i

Synowie konenia: drzewa B_0, B_1, \dots, B_{i-1}

$H_2 \leftarrow$ kopieje z tych drzew

$H \leftarrow$ meld(H_1, H_2) Koszt $O(\log n)$

Wersja lazy kopca dwumianowego:

- lista drzew dwumianowych
- może występować >1 drzewo o danym rdzie

(•) meld - połączenie list $\Theta(1)$

(••) min - banalnie $O(1)$

(•••) insert - też banalnie $O(1)$

(••••) deletemin - po usunięciu jednego trzeba znaleźć nowe min
(to może kosztować $\Omega(n)$)

Analiza kosztu amortyzowanego:

- ciąg operacji
- interesuje nas średni koszt operacji

Linnik binarny:

... 0 0 0 0 1 0

inc - zwiększenie o 1

ciąg operacji: inc, inc, inc, inc, ..., inc

1 2 1
—————
 n
2 2 2 2
—————
 n

- na końcu każdego komórek linka, które posiada cyfrę 1 jest 1 jednostka kredytowa

• przydzielamy po 2 jednostki kredytowej operacji inc

0, 0, 0, 1 $\xrightarrow{\text{inc}}$ 0, 0, 1, 0 $\xrightarrow{\text{inc}}$ 0, 0, 1, 1

Wracamy teraz do kopce:

Nierozmiennik kredytowy:

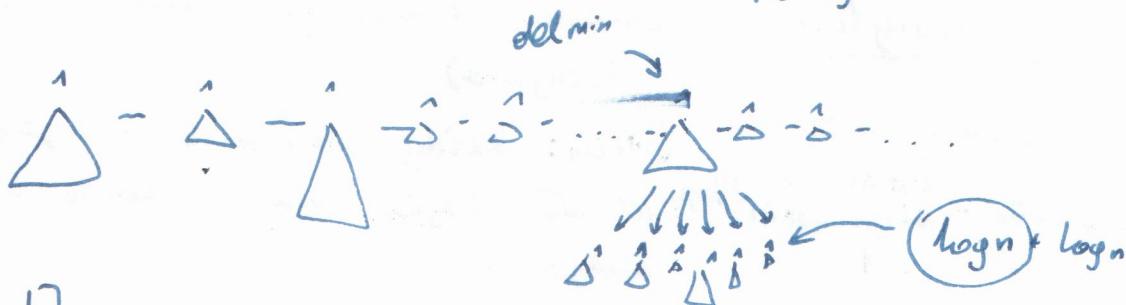
Na koncie każdego dnia kopca jest 1 kredyt

Przydatne kredyty:

- insert - 2
- min - 1
- meld - 1
- makeheap - 2
- deletemin - $2 \log n$

deletemin:

- usunięcie wartości zawierającego min
- dotarcie listy synów tego wierzchołka do listy drzew kopca
- redukcja # drzew (tak by $\forall i < 1 \text{ drzew } B_i$)



$\Delta \leftarrow$ by wykonywać na podłogach

Przeprawy listy do listy drzew. Można się
umówić, że dopisywanie będzie jednostkowe i zrobi się
3 $\log n$. Inny sposób jest ust. tak dany jednostkowy, że możemy z każdego
zobaczyć ϵ na przesłanie i na połączenie wynosi $1-\epsilon$, ale w
tym wierzchołku, w którym było min zwiększymy już ϵ na
robienie drzew, więc $1-2\epsilon$ musi być na połączenie. ???

Kopce Fibonacciego:

Wykorzystamy do Dijkstra

W tym kopcu: min, delete-min, decrease-key, insert
"opracji"
 $(n+m) \log n$