Wracamy teraz do kopca:

Niezmiennik kredytowy:

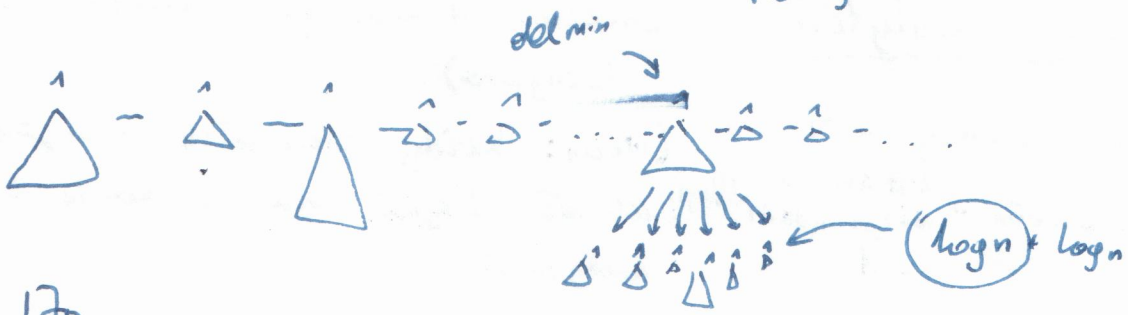
Na kuncie każdego drzewa kopca jest 1 kredyt

Przydział kredytów:

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

deletemin:

- usunięcie wierzchołka zawierającego min
- dołączenie listy synów tego wierzchołka do listy drzew kopca
- redukcja # drzew (tak by $\forall i$ < 1 drzewo B_i)



← tę zużyjemy na podłączenie

Przepisując liście do velitora płacimy. Możemy się umówić, że dopłacimy każdej jednostce 1 i zrobi się 3 log n. Inny opis jest ust. tak dużej stałej, że możemy z kredytu zabrać ε na przepisanie i na połączenie wystarczy $1-\varepsilon$, ale w tym wierzchołku, w którym było min zużyliśmy już ε na zabranie drzewa, więc $1-2\varepsilon$ musi wyst. na połączenie. ???

Kopce Fibonacciego:

Wykorzystamy do Dijkstry

W tym kopcu: min, delete_min, decrease_key, insert

n operacji m operacji

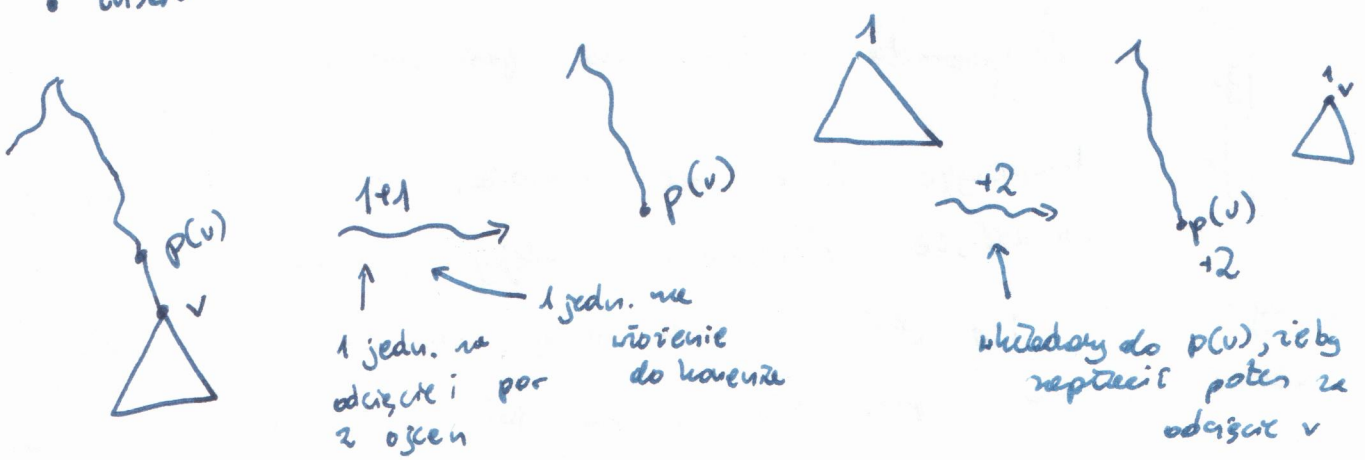
$(n+m)\log n$

kopiec podobnie jak kopiec dwumianowy (wersja luźna).
listee drzew (ale niekoniecznie dwumianowych)

decrease_key (v, Δ)
jeśli nowa wartość klucza v jest mniejsza od
klucza ojca v, to v odcinamy (wraz z poddrzewem)
od ojca (i umieszczamy na liście drzew kopca)
cut (v) ← jeśli v jest drugim dzieckiem, które utracił ojciec
to cut (ojciec (v))

Przydział kredytów: (wiemy: w każdym korzeniu 1 jednostka kredytowa)

- decrease_key - 4 (wiemy: każdy wierzchołek wewnętrzny, który utracił 1 syna ma na koncie 2 jednostki kredytowe)
- delete_min - c·# drzew o różnych rędach $\langle O(\log n) \rangle$
- min - 1
- insert - 2

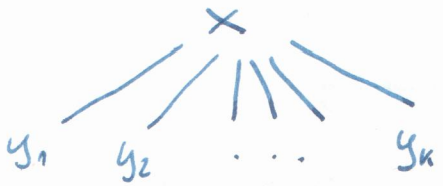


Zatem pozostaje do udowodnienia, że delete_min wymaga $O(\log n)$

FAKT

Każdy wierzchołek o rędzie k w kopcu Fib ma w swoim poddrzewie wykładniczo (wzgl. k) liczbę wierzchołków

Rozważmy dowolny moment



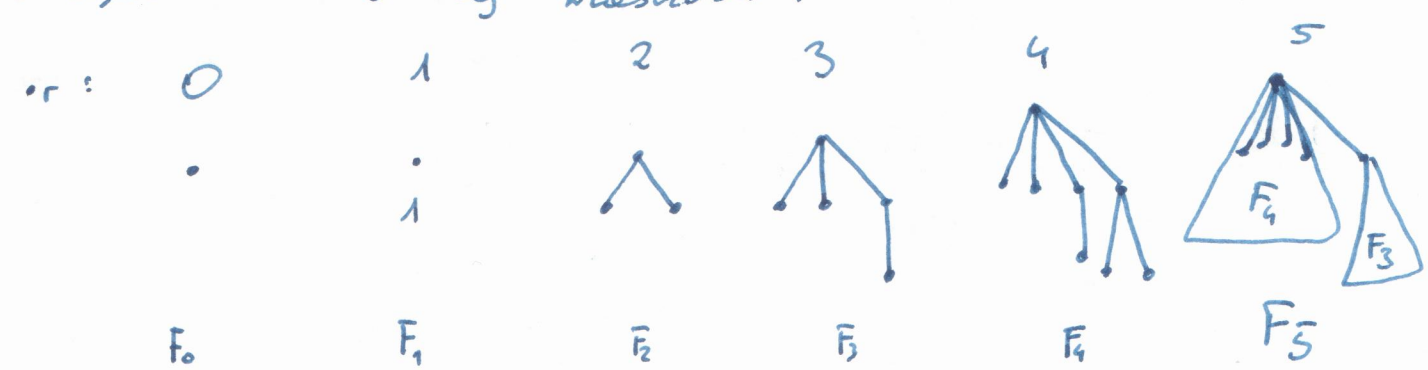
Niech $y_1 \dots y_k$ - synowie x w kolejności podwieszenia ich pod x.

Rozważmy y:

- jaki ngd może mieć y: w momencie podwieszenia pod x?
  - w tym momencie x miał $\geq i-1$ synów
  - stąd y_i tyle samo synów co x_i, a więc też $\geq i-1$
  - czyli teraz y_i ma $\geq i-2$ synów

Tak więc i-ty (w kolejności podwieszania pod x) syn x ma ngd $\geq i-2$.

Rozważmy minimalne (pod względem # wierzchołków) drzewo o ngdzie r o tej własności:

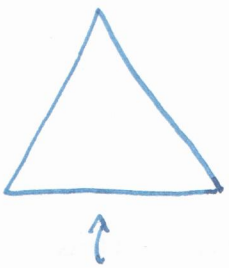


Fibonacci → 1 2 3 5 8 13

$$|F_i| = \varphi^i$$

Wniosek: Maksymalny ngd drzewa w kopcu Fibonacciego jest $O(\log n)$

Dijkstra



n elem

n - op. delete_min

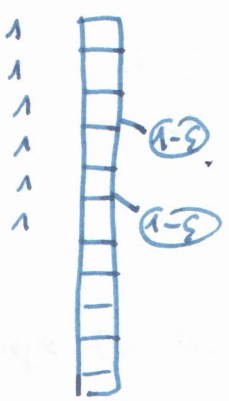
m - op. decrease_key

koszt Dijkstry $(n \log n + m)$

Lepsze wytłumaczenie $2 \log n$ dla delete_min:

$\log n + \log n$

koszt na rozbicie tego co usuwasz



i jak tworzymy nową listę, to musimy przejść całą tablicę i płacimy (1) jeśli o tablicy nie na drzewie B_i, natomiast jeśli mamy (1-ε) to płacimy (1-ε) na przepisanie do listy a (1) dajemy jako kredyt na gorsz. Mamy $2 \log(6)$

DRZEWA SAMOORGANIZUJĄCE SIĘ (drzewo Splay)

drzewo BST;

operacje słownikowe:
- insert(x)
- find(x)
- delete(x)
- Split(x) - rozbicie drzewa T na T_1 & T_2, kiedy klucze $T_1 \leqslant k$, $T_2 \geqslant k$
- join(x) - zał: klucze $T_1 \leqslant$ klucze T_2
  - wynik: drzewo z kluczami $T_1 \cup T_2$

Niezłe operacja Splay(x):

przesuń do korzenia rotacjami (sprytnie)
- x jeśli x jest w drzewie
- w.p.p. poprzednik lub następnik x-a, tj. ostatni v, w którym byliśmy robiąc find