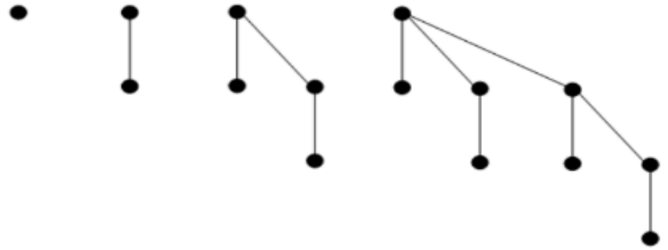
Kopce dwumianowe

Kopce dwumianowe są strukturą danych umożliwiającą łatwe wykonywanie zwykłych operacji kopcowych (*insert, makeheap, findmin* i *deletemin*) a ponadto operacji *meld* łączenia kopców.

Definicja 1 Drzewa dwumianowe *zdefiniowane są indukcyjnie: i-te drzewo dwumianowe B_i składa się z korzenia oraz i poddrzew: $B_0, B_1, \ldots, B_{i-1}$.*



Rysunek 1: Cztery pierwsze drzewa dwumianowe

Fakt 1 *Drzewo B_i zawiera 2^i wierzchołków.*

Definicja 2 Kopiec dwumianowy *to zbiór drzew dwumianowych, które pamiętają elementy z uporządkowanego uniwersum zgodnie z porządkiem kopcowym.*

Definicja 3 *Niech T będzie drzewem.* Rzędem *wierzchołka w T nazywamy liczbę jego dzieci.* Rzędem drzewa T *jest rząd jego korzenia.*

SZCZEGÓŁ IMPLEMENTACYJNY: Aby umożliwić szybką realizację operacji na kopcu dwumianowym, będziemy zakładać, że dzieci każdego wierzchołka zorganizowane są w cykliczną listę dwukierunkową, a ojciec pamięta wskaźnik do jednego z nich (np. do dziecka o najmniejszym rzędzie).

2.3 Operacja *findmin*

Z każdym kopcem dwumianowym wiążemy wskaźnik *MIN* wskazujący na minimalny element. Operacja *findmin* polega na odczytaniu tego elementu. Stąd jej koszt wynosi $O(1)$.

2.1 Łączenie drzew dwumianowych - operacja *join*

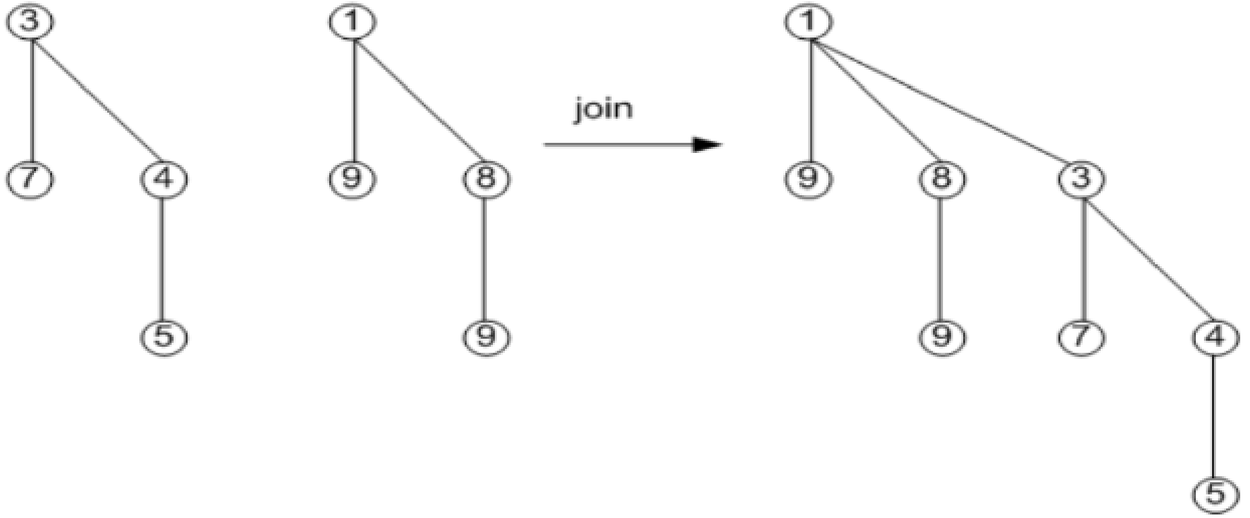
Dwa drzewa B_i łączymy ze sobą tak, że korzeń jednego drzewa staje się synem korzenia drugiego drzewa. W ten sposób otrzymujemy drzewo B_{i+1}.

UWAGI:

(a) Nigdy nie będziemy łączyć drzew o różnych rzędach.

(b) Zawsze podłączamy to drzewo, którego korzeń pamięta większą wartość do tego, którego korzeń pamięta mniejszą wartość.

Koszt operacji *link*: $O(1)$.



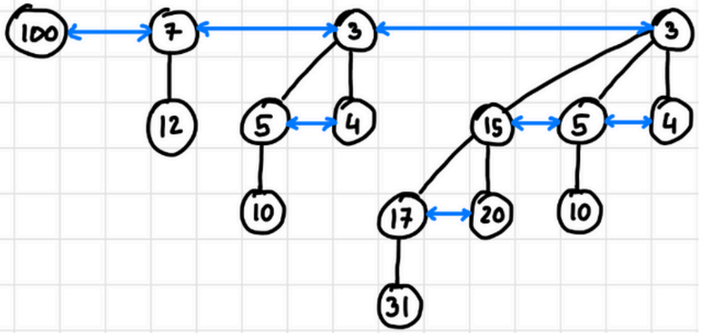
Przejrzenie prezentacji i zadań z hacka te dwa slajdy mi się usunęły i nie chciało mi się powtarzać pracy (brak zadań z ćwiczeń powiązanych z tym tematem

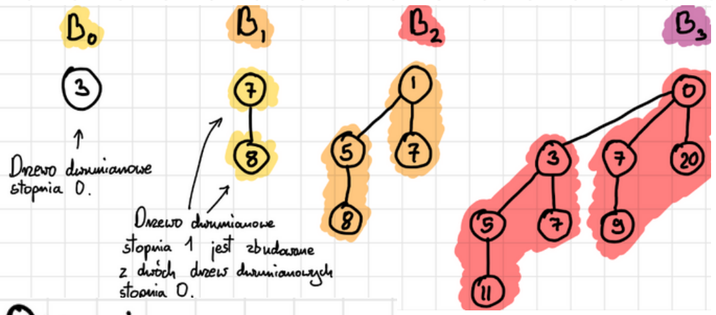
(GORLIWE)

Kopiec dwumianowy to taki kopiec na którym każde dziecko tego samego wierzchołka jak i wszystkie korzenie są połączone razem za pomocą double linked list.

- Każde drzewo dwumianowe w kopcu ma unikalny stopień k
- Stopień korzenia wynosi k (ma k dzieci)
- Wysokość drzewa B_k wynosi k.
- Jest 2^k wierzchołków w każdym drzewie.

Budowa kopca dwumianowego



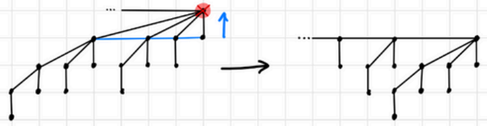


Operacje

min	$O(\log n)$
insert	$O(1)$*
merge	$O(\log n)$
delete min	$O(\log n)$

Delete min

Wykorzystujemy fakt, że dzieci wierzchołka są w liście wiązanej.



Usuwamy wierzchołek i przenosimy dzieci wyżej w odwrotnej kolejności

Delete

Jeśli usuwamy element z drzewa, to przepychamy go na wierzch, a następnie wykonujemy delete min.

Złożoność czasowa zamortyzowana dla insert *

Dodajemy n elementów do pustego kopca.
Policzmy ile scaleń s musimy za każdym razem wykonać

i	1	2	3	4	5	6	7	8	9	10	...
s	0	1	0	2	0	1	0	3	0	1	

Jak możemy zauważyć co drugi element jest chociaż raz scalany, co czwarty kolejny raz itd.

Ma to związek z reprezentacją binarną kopców.

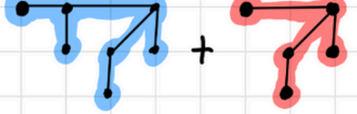
Zatem:

$$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots + \frac{n}{n} \approx 2n \quad \text{dla } n \text{ wywołań}$$

Zatem średnio dla 1 wywołania ok. 2 operacje

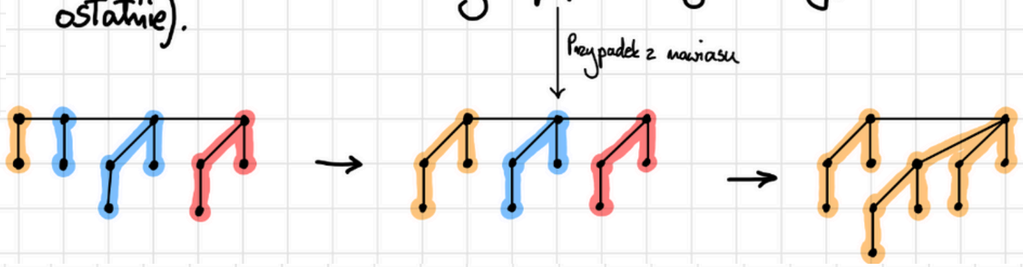
$O(1)$ amortyzowanej złożoności czasowej.

Scalmy



1. Wtasujemy te dwa kopce ze sobą

Iterujemy się od lewej strony i scalamy drzewa gdy napotkamy duplikaty (jeśli pojawią się 3 drzewa jako skutek uboczny z poprzedniej iteracji, to scal dwa ostatnie).

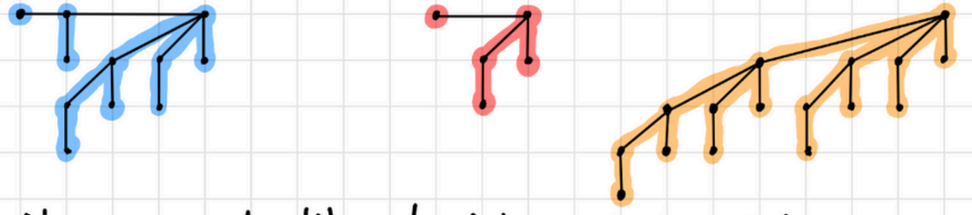


Liczby binarne

Każdy kopiec możemy opisać liczbami binarnymi:

1 1 0 1 (11) + 1 0 1 (5) = 0 0 0 0 1 (16)

B_0 B_1 B_2 B_3 — B_0 B_1 B_2 — B_0 B_1 B_2 B_3 B_4



Każda propagacja bitu, to jedno scalenie pojedyńczego drzewa.