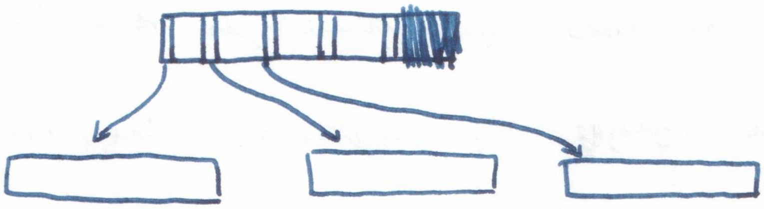
B-DRZEWA

parametr t - ograniczenie na # kluczy w wierzch.

$t \leq \#\text{kluczy} \leq 2t$

$t = 2$ klucze . 13, 15, 7, 19, 18



Cel B-drzew: trzymanie wielkich słowników. Bolą nas w AVLu dostęp do pam. zewnętrznej, który jest bardzo drogi. Ale jak już będziemy mieli w pamięci zewnętrznej to wtedy możemy na niej szybko binsearchować, dostęp jest drogi.

Cechy:

- liście leżą na tym samym poziomie
- drzewo rośnie „do góry"

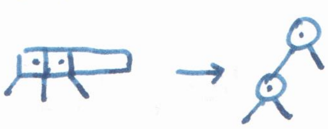
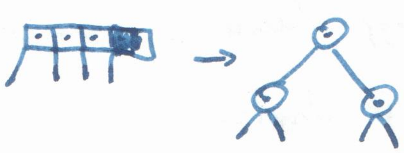
Jeśli t jest duże, to drzewo jest płytkie

Przechodzenie przez wierzchołki jest kosztowne

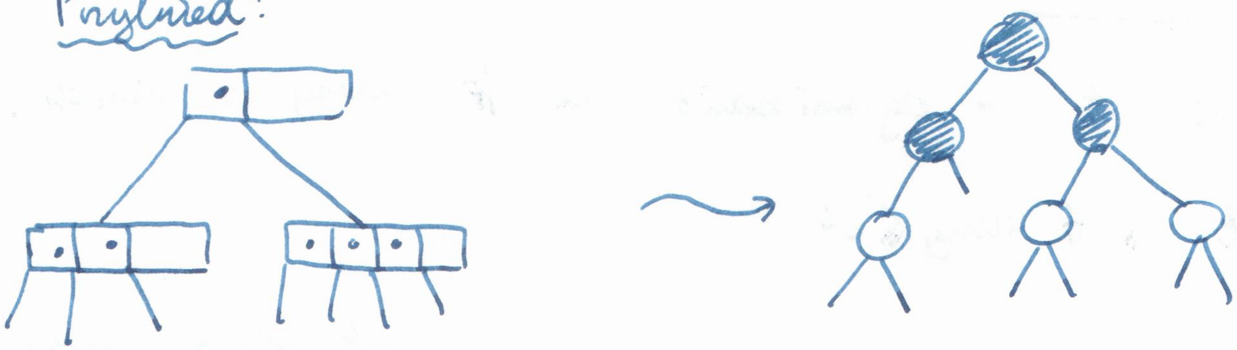
Gdy słownik jest wielki #elementów(n) >> pamięć wew

(2-3) drzewo (1 ≤ kluczy ≤ 3)

wierzchołek

- 1 klucz 
- 2 klucze  lub 
- 3 klucze 

Przykład:

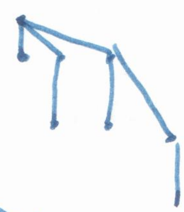


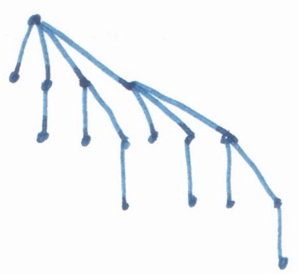
B-drzewa były motywacją do drzew czerwono-czarnych. Teraz te dziwne operacje z drzew wydają się naturalne patrząc na B-drzewa

ZŁĄCZALNE KOLEJKI PRIORYTETOWE

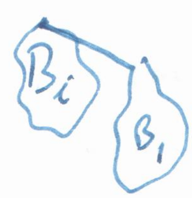
Będziemy mówić o kopcach dwumianowych w wersjach lazy i eager, a potem przejdziemy do kopca Fibonacciego. Na razie kopce dwumianowe w wersji gorliwej.

DRZEWA DWUMIANOWE

B_0: B_1:  B_2:  B_3: 

B_4: 

1
4
6
4
1 } wierzchołków

B_{i+1}: 

W kopcu dwumianowym mamy las drzew dwumianowych. Wierzchołki tych drzew pamiętają klucze w porządku kopcowym.

Fakt: Drzewo B_i ma 2^i wierzchołków.

(pomijamy skomplikowany dowód indukcyjny).