Heaps, 30.11

# Термини

## Строго двоично дърво

* Абсолютно всеки един възел има 2 или 0 деца

## Пълно двоично дърво

* Всичките листа трябва да са на едно и също ниво (трябва да присъстват всички елементи на азбуката)
* Всеки възел трябва да има 2 или 0 деца.

## Завършено двоично дърво

* Всички листа трябва да са на максимум едно ниво разлика. (На последното или предпоследното ниво)

## Двоично дърво за търсене

* Двоично дърво
* Всеки един node има стойност
* Ключовете трябва да бъдат сравними (<, >=)
* X корен, всички елементи на ляво от Х да са по-малки и всички надясно от Х да са по-големи.

## Триене от BST

* Възела е листо
* Възел с едно дете
* Възел с две деца

## Недостатъци на BST

* Линейна сложност като не е балансирано.
* Много зависи от реда, в който му подаваме данните

## Balanced BST

* Добро дърво – височина logn
* Лошо дърво – височина n

## AVL trees

* При един или два възела не можем да намерим insight, но при 3 възела може да се получи пръчка или дърво.
* 3, 2, 1 – пръчка
* 3, 1, 2 – пръчка
* 1, 3, 2 – пръчка
* 1, 2, 3 – пръчка
* 2, 1, 3 – балансирано
* 2, 3, 1 – балансирано

### Rotations

* 3, 2, 1 -> 2, 1, 3 / 2, 3, 1 L L rotation
* 1, 2, 3 -> 2, 1, 3 / 2, 3, 1 R R rotation
* 3, 1, 2 -> разменяме 2 и 1 и после L L / R R rotation

### Как да разберем, че сме счупили дървото

* Balance factor
* Височината на лявото – височината на дясното под дърво
* Едно дърво е балансирано, ако всеки node има balance factor -1, 0, 1
* Когато баланса на един node е в дисбаланс, правим ротация спрямо него
* Памет за височините на конкретните node-ове

## Scape goat tree

* Не изисква памет, пресмята по формула дали е балансирано или не.

## Splay tree

* Последно търсения елемент става root
* Garbage collection
* Cache-ing