Дървета – двоични, за търсене, балансирани

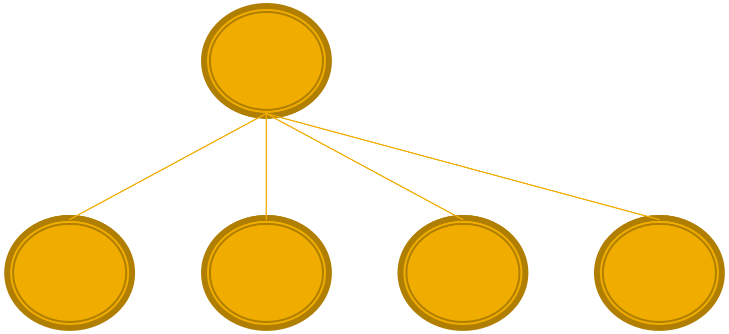
доц. д-р Нора Ангелова

Съдържание

- Дърво
- Двоично дърво
- Двоично наредено дърво (дърво за търсене)
- Балансирано дърво и идеално балансирано дърво

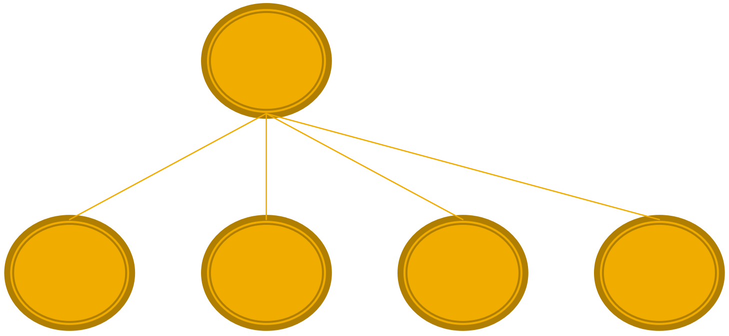
Дърво

- дефиниция – (мат. деф. свързан граф без цикли)
  - празното дърво е дърво
  - дървото с единствен връх е дърво
  - дървото с корен и произволен брой наследници, които са също дървета е дърво
- разклонена хомогенна структура от данни
- елементите на дървото се наричат върхове или възли
- предоставя директен достъп до **корена**
- всеки връх има произволен брой наследници



Дърво

- Свързана реализация
  - Чрез свързан списък/вектор от елементи
    - създаване на дърво с единствен връх – листо
    - добавяне на поддърво в дърво
    - достъп до стойността на корена
    - достъп до наследниците
    - дълбочина на дървото
    - ширина на дървото



Съдържание

- Дърво
- Двоично дърво
- Двоично наредено дърво (дърво за търсене)
- Балансирано дърво и идеално балансирано дърво

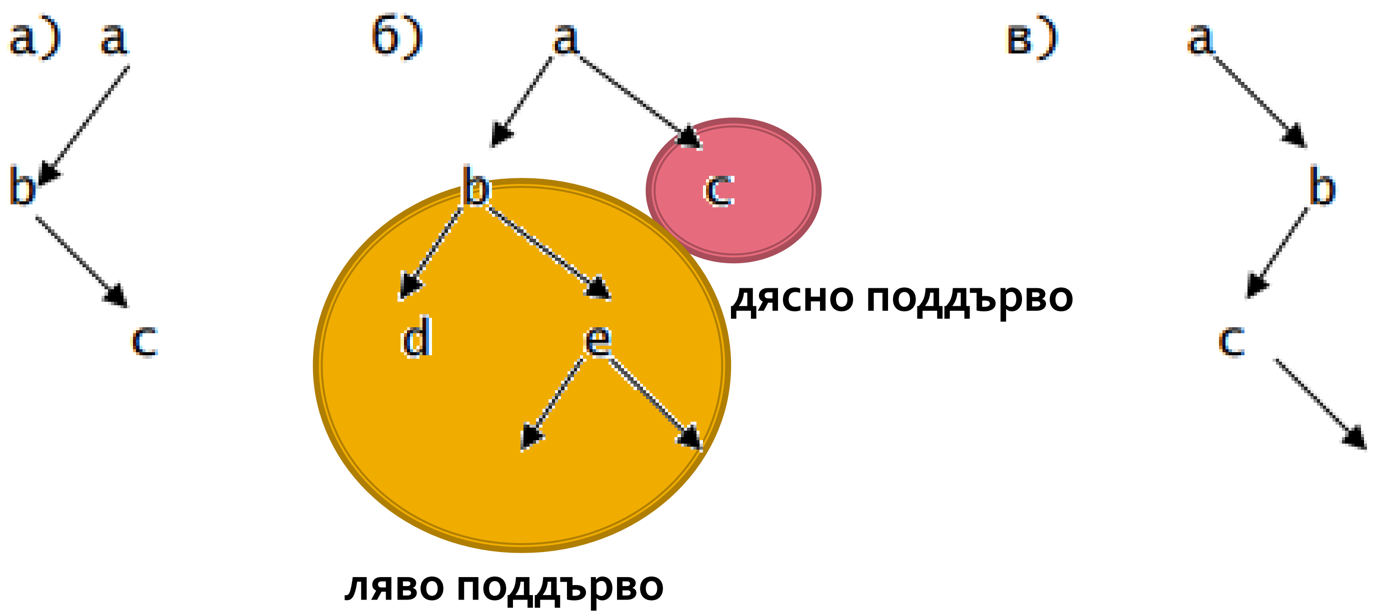
Двоично дърво

Двоично дърво от тип T е рекурсивна структура от данни, която е или празна или е образувана от:

- Данна от тип T, наречена **корен** на двоичното дърво;
- Двоично дърво от тип T, наречено **ляво поддърво** на двоичното дърво;
- Двоично дърво от тип T, наречено **дясно поддърво** на двоичното дърво.

Двоично дърво

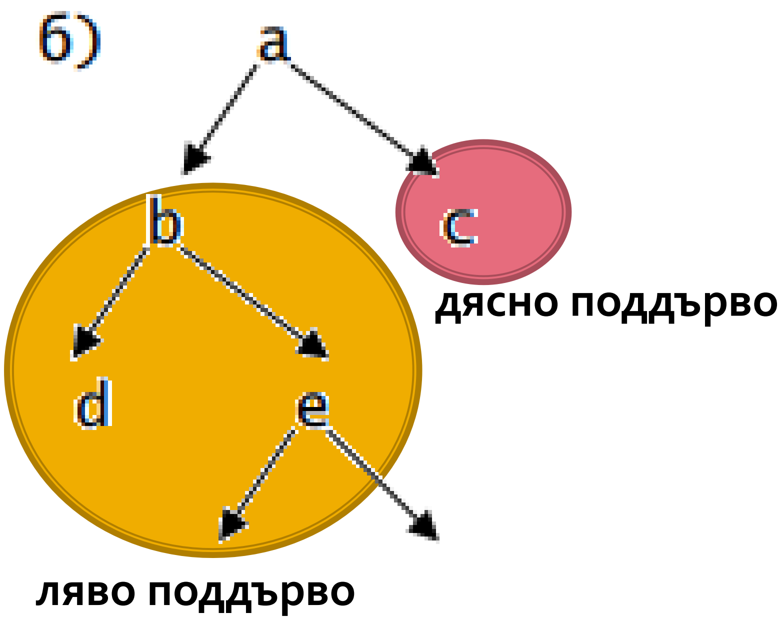
Пример:



Двоично дърво

Множеството на **върховете (възлите)** на едно двоично дърво се определя рекурсивно:

- Празното двоично дърво няма върхове.
- Върховете на непразно дърво са неговият корен и върховете на двете му поддървета.



// a, b, d, e, c

Двоично дърво

- **Листа** – върховете с две празни поддървета.

- **Вътрешни върхове** – върховете, различни от корена и листата.

- **Ляв наследник** на един връх – коренът на лявото му поддърво (ако то е непразно).
- **Десен наследник** на един връх – коренът на дясното му поддърво (ако то е непразно).

- Ако a е наследник на b (ляв или десен), казваме, че b е **родител** (**баща**) на a.

Двоично дърво

- **Ниво** – коренът на дървото има ниво 1 (или 0). Ако един връх има ниво i, то неговите наследници имат ниво i+1.

- **Височина (дълбочина)** – максималното ниво на едно дърво.

Двоично дърво

- **Достъп до връх** – възможен е пряк достъп до корена и непряк достъп до останалите върхове.

- **Операции** – возможни са операциите добавяне и премахване на върхове на произволно място в двоичното дърво, но резултатът трябва отново да е двоично дърво от същия тип. Как ще се извършва добавянето и изтриването?

Двоично дърво

Физическо представяне на двоично дърво:

- Последователно
- Свързано

Двоично дърво

Физическо представяне на двоично дърво:

- Последователно
  - Верижно
  - Списък на бащите

Двоично дърво

Верижно представяне

Използват се три масива – `a[N]`, `b[N]` и `c[N]`.

** N е броят на върховете в дървото*

Върховете са номерирани от 0 до N-1.

- `a[i]` – стойността на i-тия връх на дървото.
- `b[i]` – индексът на левия наследник на i-тия връх (-*1, ако той няма ляв наследник*).
- `c[i]` – индексът на десния наследник на i-тия връх (-*1, ако той няма десен наследник*).
- Индексът на корена – пази се отделно.

Двоично дърво

Чрез списък на бащите

Представя се с един масив `p[N]`.

** N е броят на върховете в дървото*

Върховете са номерирани от 0 до N-1

- `p[i]` - единственият баща на i-тия връх на дървото (-*1, ако този връх е коренът*).

https://www.geeksforgeeks.org/construct-a-binary-tree-from-parent-array-representation/

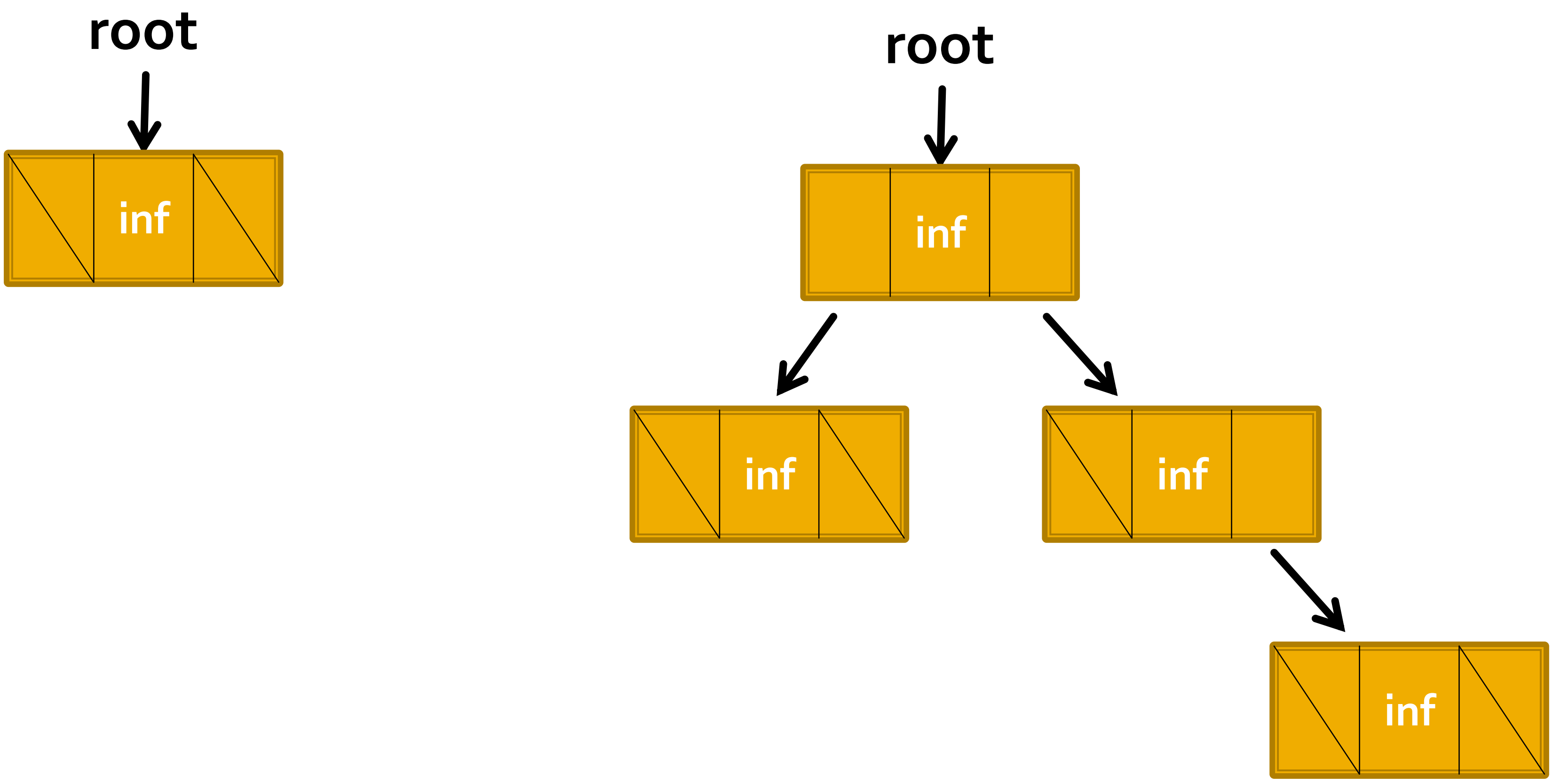
Двоично дърво

Физическо представяне на двоично дърво:

- Последователно
  - Верижно
  - Списък на бащите
- Свързано

Двоично дърво

Свързано представяне

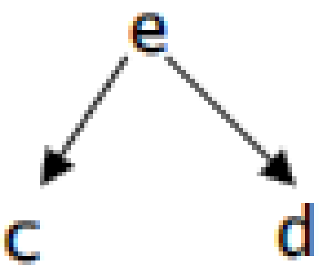


Обхождане на двоично дърво

Обхождането е рекурсивна процедура, която се осъществява чрез изпълнение на следните три действия, в някакъв фиксиран ред:

- обхождане на корена
- обхождане на лявото поддърво
- обхождане на дясното поддърво

Двоично дърво



- **смесеното обхождане (ЛКД)** – c, e, d
- **низходящото обхождане (КЛД)** – e, c, d
- **възходящо обхождане (ЛДК)** – c, d, e

Съществуват още три типа обхождания.

- **КДЛ** – e, d, c
- **ДКЛ** – d, e, c
- **ДЛК** – d, c, e

Операции в двоично дърво

Специфики на основните операции:

- обхождане на дърво
- търсене на елемент
- добавяне/изтриване на елемент
  - къде да се извърши добавено или кой точно елемент да се изтрие?
  - как се избира новия корен, как се осигурява разклонеността на дървото?

Сложност на операциите

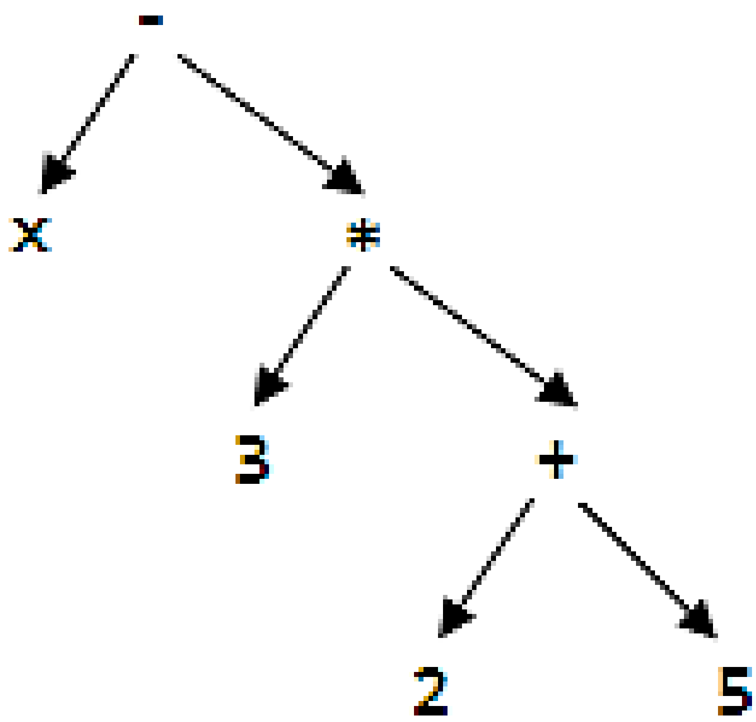
- Средна сложност на операцията добавяне – `O(log N)`
- Средна сложност на операцията търсене – `O(N)`

Итератор?!?

- Движение в двоично дърво – позволява се движение към левия или към десния наследник на дървото
- За целта може да се реализира абстракция на позиция в дърво
  - скрива информацията за вътрешното представяне и възлите в дървото
  - позволява пренасочване на връзките в дървото – при операция добавяне или премахване на елемент
  - дава възможност за работа с позиции при всички вътрешни и външни операции

Приложения на двоично дърво

Представяне на изрази
x – 3 * (2 + 5)



Приложения на двоично дърво

Представяне на изрази

- Възходящия обход (ЛДК) – обратен полски запис

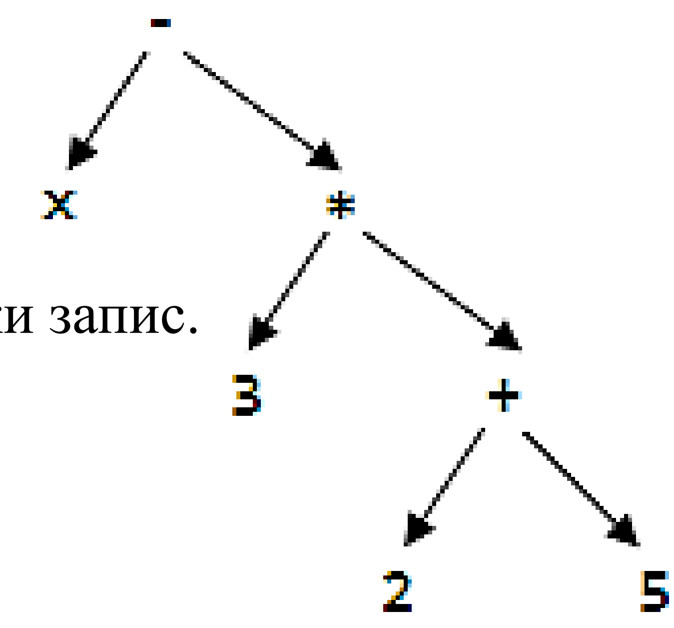
Пример:

x 3 2 5 + * -

- Смесеният обход (ЛКД) – инфиксен запис на аритметичния израз *(без скобите)*

Пример:

x – 3 * 2 + 5



- Низходящият обход (КЛД) – прав полски запис.

Пример:

– x * 3 + 2 5

Съдържание

- Дърво
- Двоично дърво
- Двоично наредено дърво (дърво за търсене)
- Балансирано дърво и идеално балансирано дърво

Двоично наредено дърво (дърво за търсене)

Двоично дърво

- Сложност на операциите добавяне и търсене - O(log N) и O(N)
- Оптимизация

Двоично наредено дърво

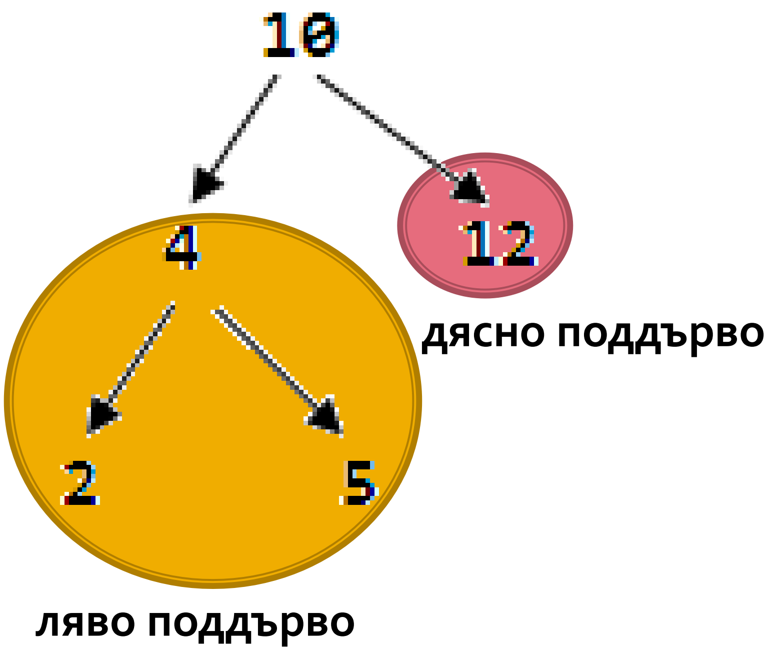
Двоично наредено дърво от тип Т е рекурсивна структура от данни и се дефинира по следния начин:

- Празното двоично дърво е двоично наредено дърво.

- Непразно двоично дърво, върховете на лявото поддърво на което са по-малки от корена, върховете на дясното поддърво са по-големи от корена и лявото, и дясното поддърво са двоично наредени дървета от тип Т.

Двоично наредено дърво

Пример:



Двоично наредено дърво

Свойства

- Смесеното обхождане (ЛКД) сортира върховете във възходящ ред.

- Обхождането по метода (ДКЛ) сортира върховете в низходящ ред.

Двоично наредено дърво

Търсене на елемент

Нека tree е двоично наредено дърво от тип Т.

Търсене елемента *a* от тип Т в tree се осъществява по следния начин :

- Извличаме корена
- Ако елементът съвпада с него, елементът е намерен
- Ако елементът е по-малък от корена, търсим в лявото поддърво
- Ако елементът е по-голям от корена, търсим в дясното поддърво

Двоично наредено дърво

Включване на елемент

Нека tree е двоично наредено дърво от тип T.

Включване на елемента *newElemData* от тип T в tree се осъществява по следния начин:

- Не можем да включим елемент със същата стойност
- Ако tree е празното двоично дърво, новото двоично наредено дърво е с корен елемента *newElemData* и празни ляво и дясно поддървета.
- Ако tree не е празно и *newElemData* е по-малко от корена му, елементът *newElemData* се включва в лявото поддърво на tree.
- Ако tree не е празно и *newElemData* е не по-малко от корена му, елементът *newElemData* се включва в дясното поддърво на tree.

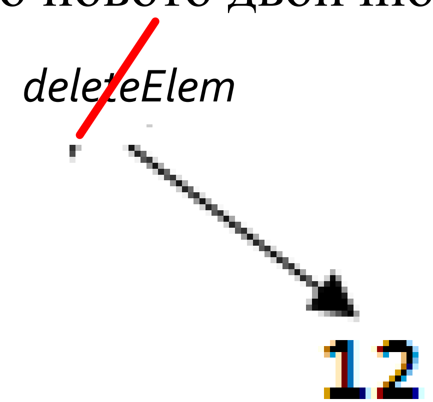
Двоично наредено дърво

Изтриване на елемент

Нека tree е двоично наредено дърво от тип T.

Изтриване на елемента *deleteElem* от tree се осъществява по следния начин:

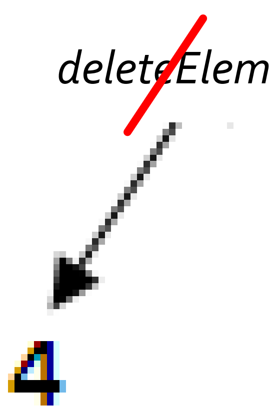
- Ако *deleteElem* не е в дървото, не може да се извърши изтриване
- Ако *deleteElem* е корен на tree с празно ЛПД, то новото двоично наредено дърво е ДПД на tree.



Двоично наредено дърво

Изтриване на елемент

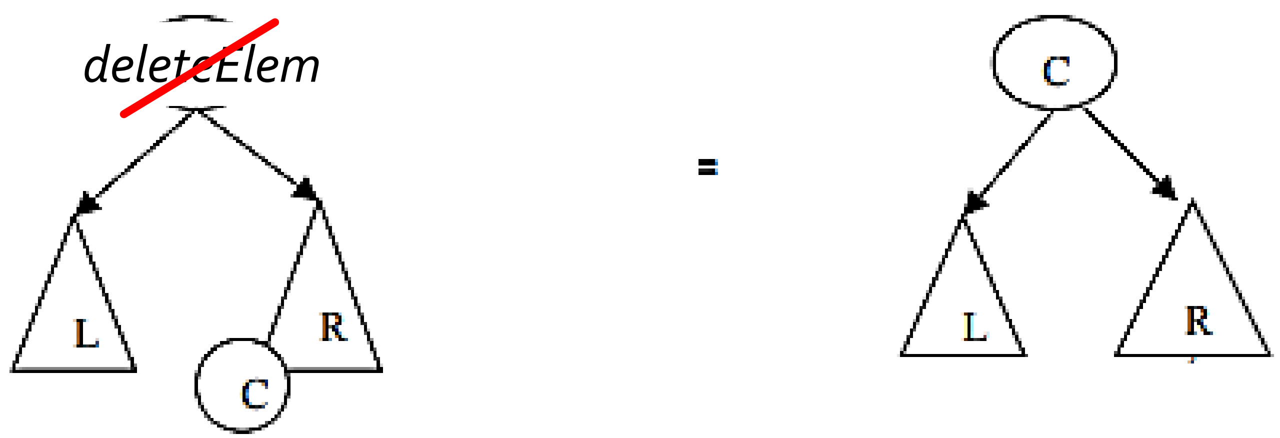
- Ако *deleteElem* е корен на tree с празно ДПД, то новото двоично наредено дърво е ЛПД на tree.



Двоично наредено дърво

Изтриване на елемент

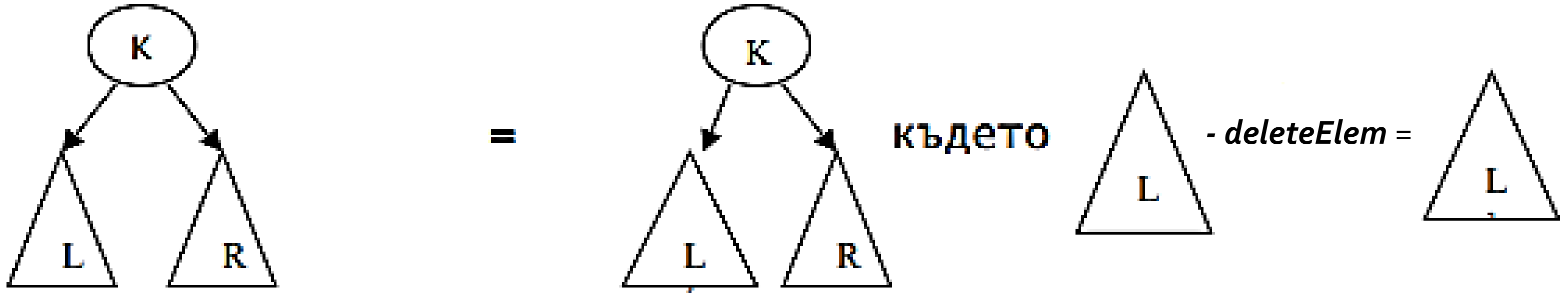
- Нека *deleteElem* е корен на tree с непразни ляво и дясно поддървета и *c* е най-лявото листо от ДПД на tree. Новото двоично наредено дърво има корен елемента *c*, ЛПД е ЛПД на tree и ДПД е двоичното наредено дърво, получено от ДПД на tree след изключване на елемента *c*.



Двоично наредено дърво

Изтриване на елемент

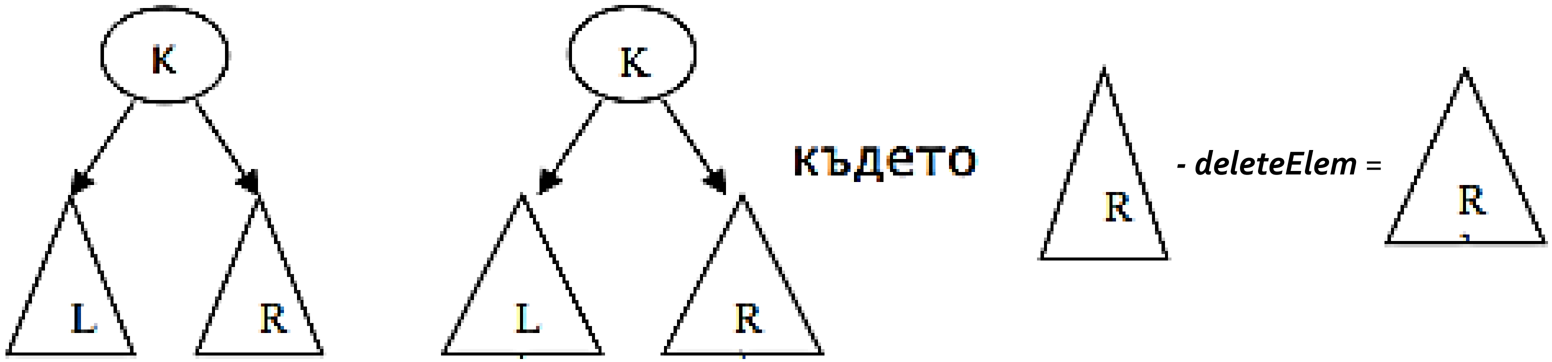
- Ако ***к*** е корен на `tree` с непразни ляво и дясно поддървета и стойността на *deleteElem* е по-малка от стойността на ***к***, то новото двоично наредено дърво има корен ***к***, ЛПД е ЛПД на `tree`, от което е изключен елемента *deleteElem*, и ДПД е ДПД на `tree`;



Двоично наредено дърво

Изтриване на елемент

- Ако ***к*** е корен на `tree` с непразни ляво и дясно поддървета и стойността на *deleteElem* е по-голяма от стойността на ***к***, то новото двоично наредено дърво има корен ***к***, ЛПД е ЛПД на `tree` и ДПД е ДПД на `tree`, от което е изключен елемента *deleteElem*.



Двоично наредено дърво

Помощни методи за добавяне и/или изтриване на елемент

- Всички случаи в дърво за търсене могат да се разгледат посредством операцията търсене
  - проверка дали елемент вече се среща в дървото
  - ако елементът не се среща в дървото, функцията връща информация за мястото, където той би бил добавен
  - ако елементът се среща в дървото, функцията връща точната му позиция

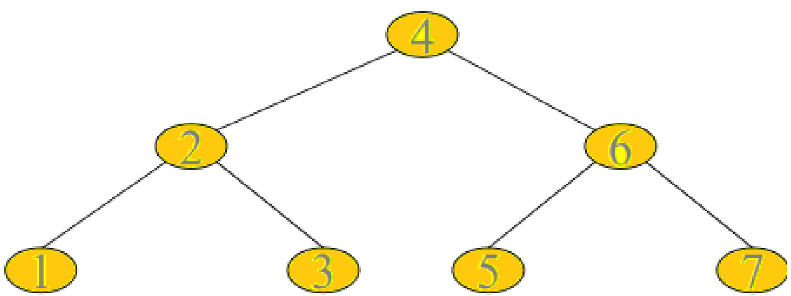
Съдържание

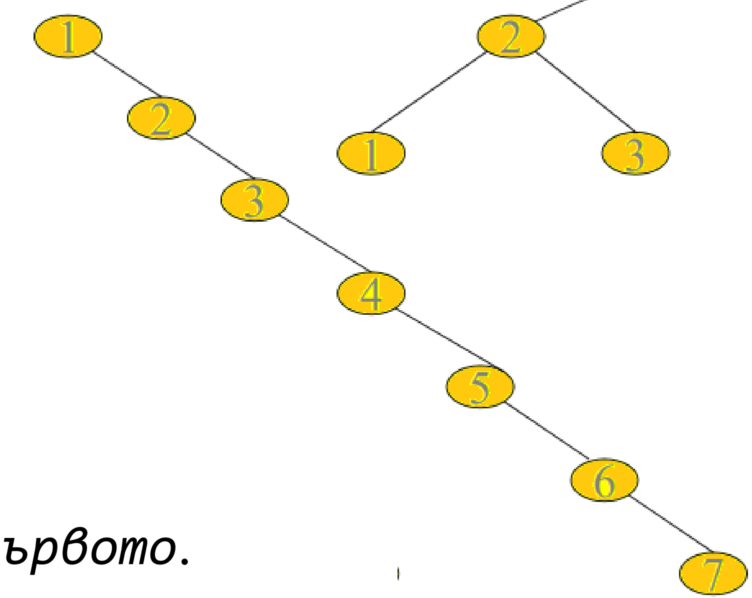
- Дърво
- Двоично дърво
- Двоично наредено дърво (дърво за търсене)
- Балансирано и идеално балансирано дърво

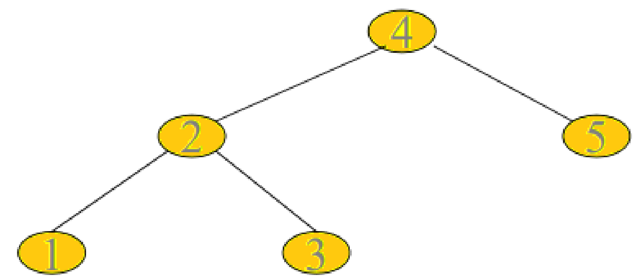
Балансиране на дърво

За дърветата за търсене се знае, че

- Средна сложност на операциите добавяне и търсене - `O(log N)`.
- Добавяне на елементите подредени по големина - `O(N)`.







** N – брой на върховете в дървото.*

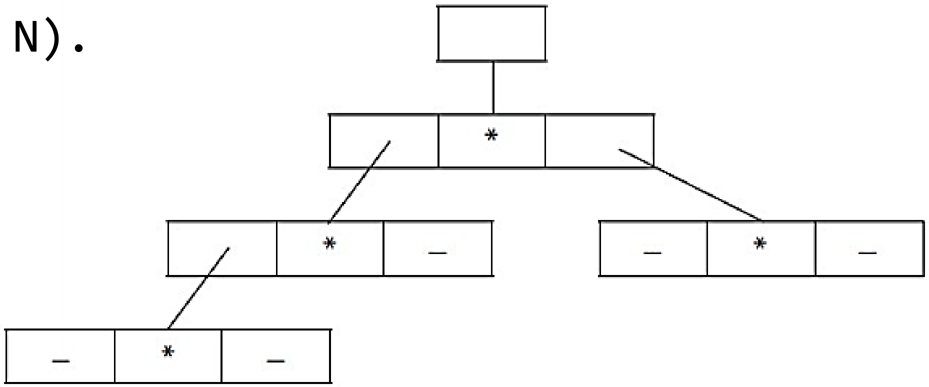
Идеално балансирано дърво

Дефиниция
Двоично наредено дърво се нарича идеално (перфектно) балансирано, ако всеки негов връх има ляво и дясно поддърво, в които **броят на възлите** се различава най-много с 1.

Двоично наредено дърво със следните свойства:

- Броят на възлите в лявото и дясното поддърво се различава най-много с 1.
- Лявото и дясното поддървета са идеално балансирани двоично наредени дървета.

Сложност на операциите - O(log N).



** N – брой на върховете в дървото.*

Балансирано дърво (AVL)

Дефиниция

Двоично наредено дърво се нарича двоично дърво с балансирана височина или само балансирано дърво, ако за всеки негов връх **височините** (дълбочините) на лявото и дясното му поддървета се различават най-много с 1.

Двоично наредено дърво със следните свойства:

- Височините на лявото и дясното поддърво се различават най-много с 1.
- Лявото и дясното поддървета са балансирани двоично наредени дървета.

Сложност на операциите - O(log N).

** N – брой на върховете в дървото.*

Балансирано и идеално балансирано дърво

Свойства:

- Всяко идеално балансирано дърво е балансирано дърво, но обратното не е вярно.
- Алгоритмите за добавяне и премахване на връх от двоично наредено дърво не запазват балансираността и добрите сложности.
- Съществуват алгоритми, които запазват балансираността.
- Съществува прост алгоритъм за създаване на идеално балансирано двоично наредено дърво при определени ограничения.

Балансирано и идеално балансирано дърво

Алгоритъм за създаване на идеално балансирано дърво

- Елементите, които ще се включват към празното двоично наредено дърво се подават в нарастващ ред.
- Предварително е известен броят на върховете на дървото.

```
n - брой на върховете в дървото.
n = nLeft + nRight + 1 && |nLeft - nRight| <= 1
```

Балансирано и идеално балансирано дърво

Алгоритъм за създаване на идеално балансирано дърво

```
n - брой на върховете в дървото.
n = nLeft + nRight + 1 && |nLeft - nRight| <= 1
```

- Конструира ляво поддърво с `nLeft` върха.
- Създава корен на дървото.
- Конструира дясно поддърво с `nRight` върха.

Балансирано дърво

Физическо представяне

Четворна кутия – добавя се още един параметър, означаващ коефициент на балансираност, задаващ разликата на височините на дясното и лявото поддърво на съответния връх.

Балансирано дърво

- **Ротации** - операции, които пренареждат част от елементите на дървото при добавяне или при премахване на елемент от него, за да се избегне дисбалансът му.

- Ротациите зависят от реализацията на конкретната структура от данни. Примери за такива структури:
  - **червено-черно** дърво
  - **AVL** - дърво
  - **AA** - дърво и др.

Балансирано и идеално балансирано дърво

AVL дърво
https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

- **Търсене** – осъществява се по същия начин както при обикновено небалансирано двоично наредено дърво.

Най-лош сценарий:
Трябва да се обходи от корена до най-далечното листо.
Време: пропорционално на височината на дървото – `O(log n)`.

Балансирано дърво

AVL дърво

- **Обхождане** – осъществява се по същия начин както при обикновено небалансирано двоично наредено дърво.

Обхождат се всички върхове – `O(n)`.

Балансирано дърво

AVL дърво

Добавяне на връх

Операцията включване на елемент в балансирано двоично наредено дърво се осъществява по аналогичен начин на включването на елемент в двоично наредено дърво. Разликата е, че при всяко включване, което променя балансираността на дървото трябва да се изпълнят едно или две завъртания с цел възстановяване на балансираността.

Балансирано дърво

AVL дърво

Добавяне на връх:

- Върхът се добавя като листо.

- Повторно обхождане **`(retracing)`** - проверка за съответветствие с инвариантите на AVL дърво. Започва се от родителя на добавения връх и се стига до корена.

Реализира се чрез пресмятане на баланс фактор за всеки връх, който се дефинира като разликата във височините на дясното и лявото поддървета.

`balance – цяло число в интервала [-1 1].`

`(инварианти на AVL дърво)`

Балансирано дърво

AVL дърво

Добавяне на връх:

- Връх с `balance ∈ [-1 1]` - поддървото е балансирано и не е необходима ротация.

- Връх с `balance ∈ [-2 2]` – поддървото е небалансирано и е необходима ротация.

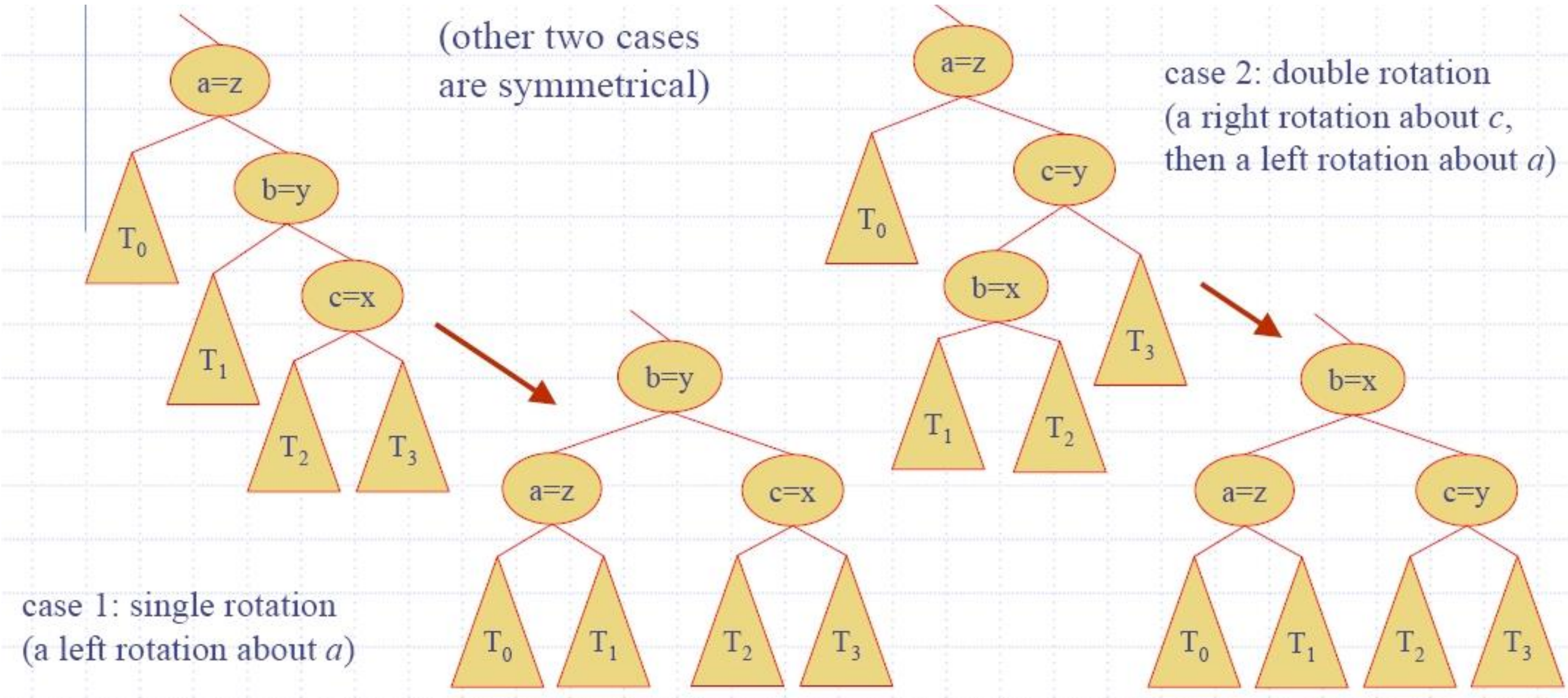
Балансирано дърво

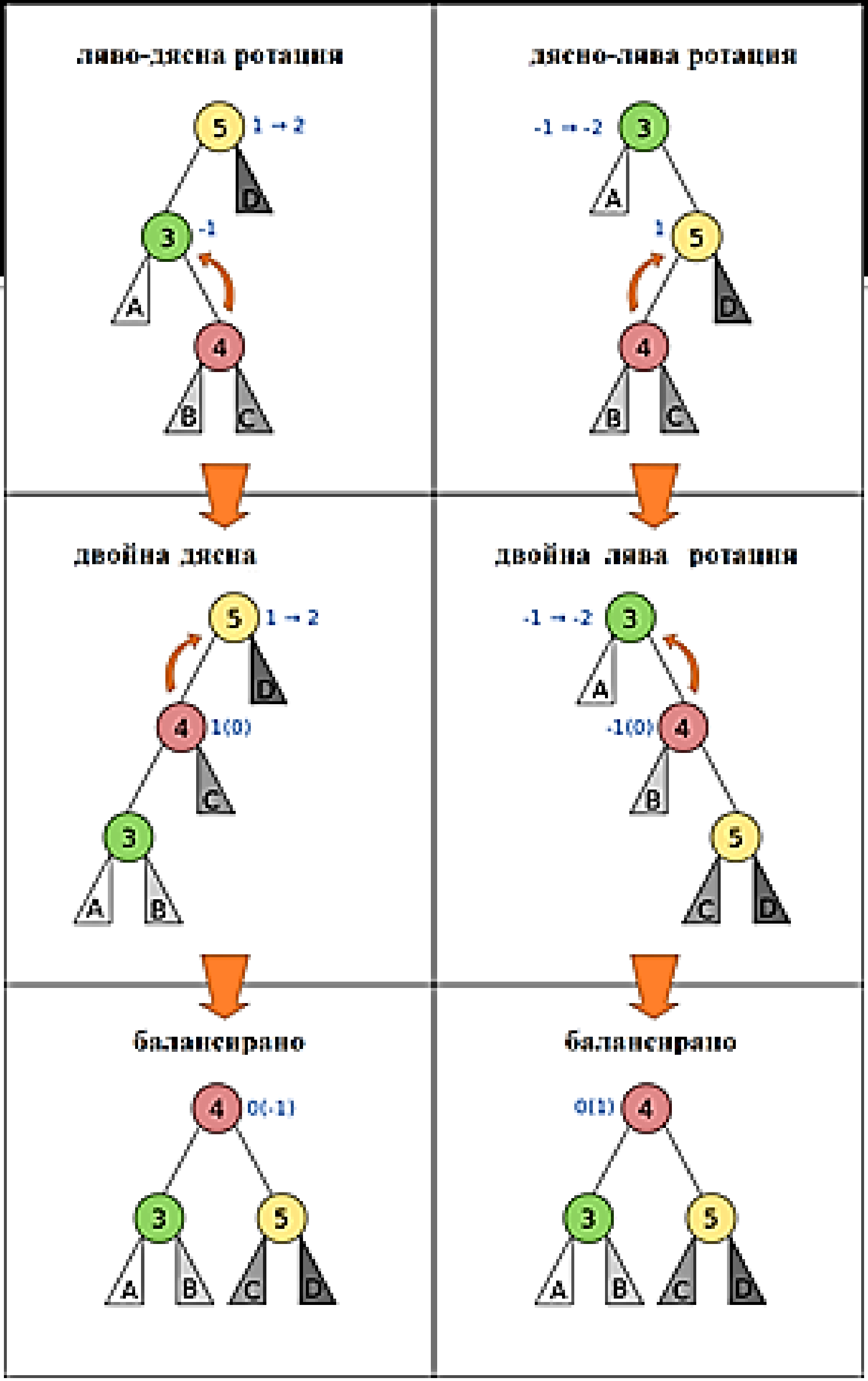
AVL дърво

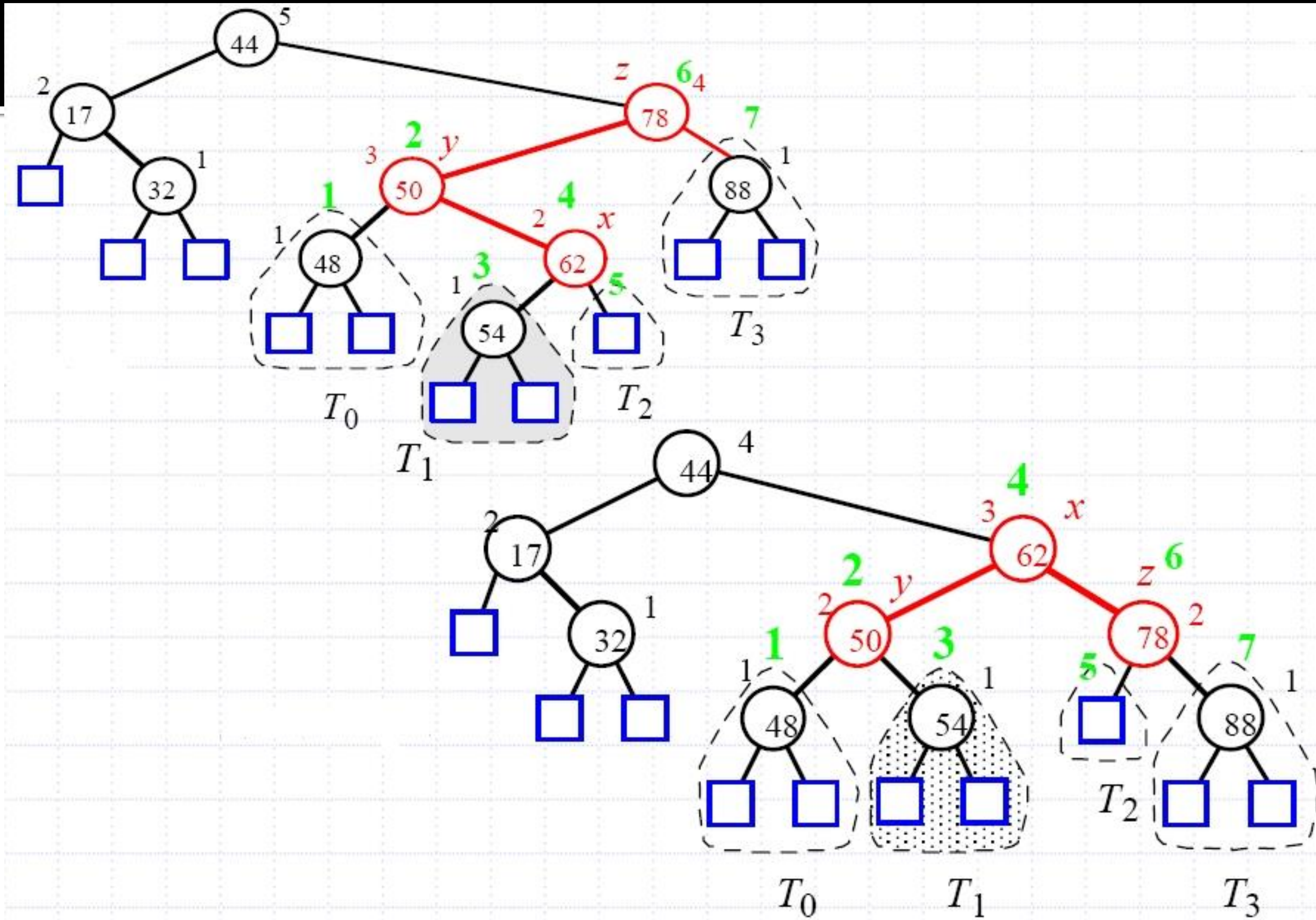
Ротационни функции:

- Дясна ротация.
- Лява ротация.
- Двойна ляво-дясна ротация (състои се от лява ротация, последвана от дясна).
- Двойна дясно-лява ротация (състои се от дясна ротация, последвана от лява).

Балансирано дърво




ляво-дясна ротация
дясно-лява ротация
двойна дясна
двойна лява ротация
балансирано
балансирано



44
17
32
50
48
62
54
78
88
z
y
x
T0
T1
T2
T3

Балансирано дърво

AVL дърво

Изтриване на връх:

- Изтриване на връх от дървото.

- Повторно обхождане **`(retracing)`** - проверка за съответветствие с инвариантите на AVL дърво. Започва се от родителя на изтрития връх и се стига до корена.

Следва продължение...