Дървета за търсене

Трифон Трифонов

Структури от данни и програмиране, спец. Компютърни науки, 2 поток, 2024/25 г.

22 декември 2022 г. – 5 януари 2023 г.

"Appletree" от User:MKFF (https://commons.wikimedia.org/wiki/File:Appletree_2.jpg), CC0

Дървета за търсене

- Организация, която позволява бързо намиране на елементи в дървото
- Разчита на **линейна наредба** на елементите
- Основни операции:
  - `create()` — създаване на празно дърво за търсене
  - `insert(x)` — включване на елемент
  - `remove(x)` — изключване на елемент
  - `search(x)` — търсене на елемент
- Обикновено елементите са двойки (ключ, стойност)
- Елементите са наредени относно ключовете си
- Стойностите носят данните на елемента

Двоично дърво за търсене

Дефиниция (Двоично дърво за търсене)

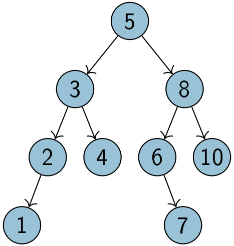
- Празното дърво $\perp$ е ДДТ
- (X, L, R) е ДДТ, ако
  - X е по-голямо от всички елементи в L
  - X е по-малко от всички елементи в R
  - L и R също са ДДТ

Двоично дърво за търсене

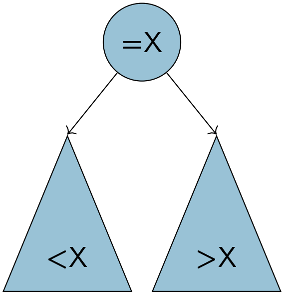
Дефиниция (Двоично дърво за търсене)

- Празното дърво $\perp$ е ДДТ
- (X, L, R) е ДДТ, ако
  - X е по-голямо от всички елементи в L
  - X е по-малко от всички елементи в R
  - L и R също са ДДТ

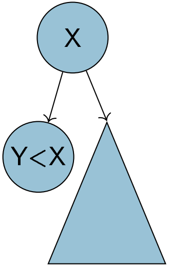
Пример:



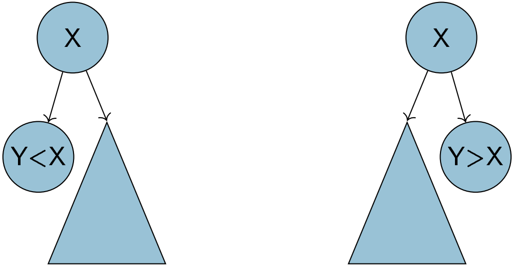
Търсене на елемент



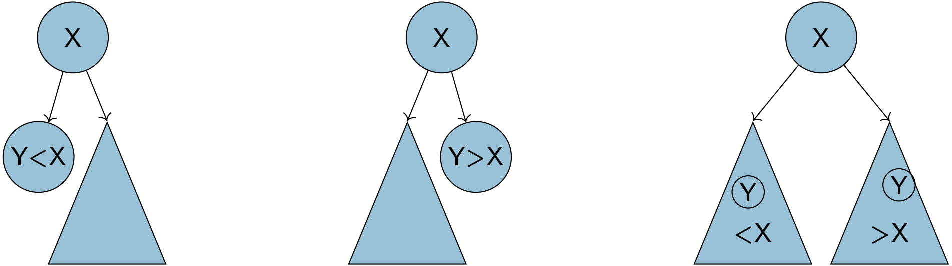
Включване на елемент



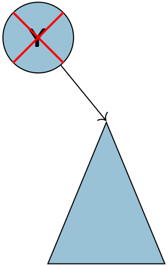
Включване на елемент



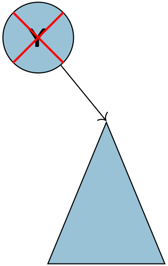
Включване на елемент

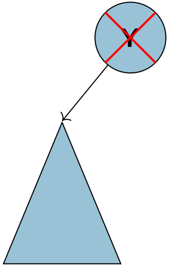


Изключване на елемент

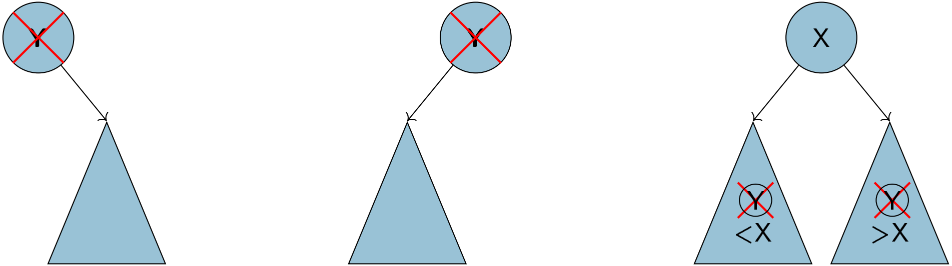


Изключване на елемент

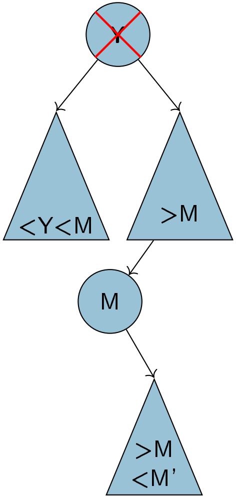




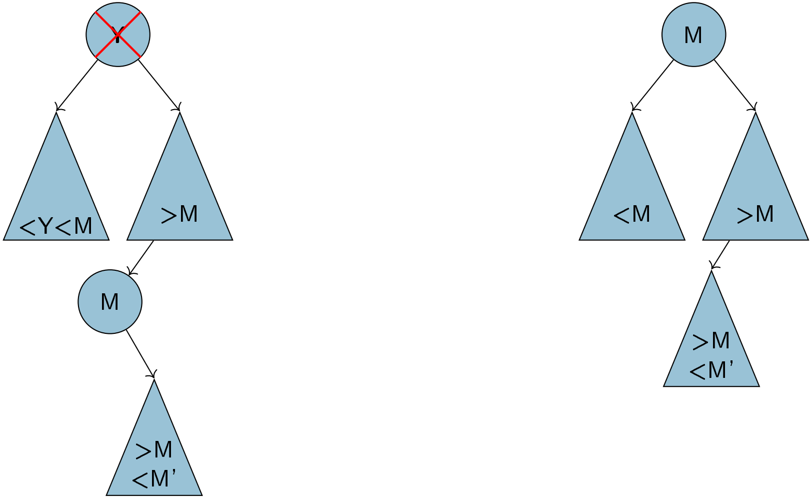
Изключване на елемент



Изключване на елемент — общ случай



Изключване на елемент — общ случай



Оптимална височина на дърво

Сложността на всички операции за двоично дърво до търсене е $O(h)$, където h е височината на дървото.

Знаем, че $\log_2(n+1) \leq h \leq n$.

- $h = n \leftrightarrow$ дървото е изродено до списък
- $h = \log_2(n+1)$, когато дървото е пълно
- само тогава ли?

Балансирано дърво

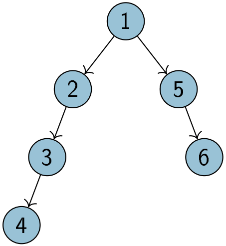
Дефиниция (Балансирано дърво)

- Празното дърво $\perp$ е балансирано
- (X, L, R) е балансирано, ако
  - $|h(L) - h(R)| \leq 1$
  - L и R също са балансирани

Балансирано дърво

Дефиниция (Балансирано дърво)

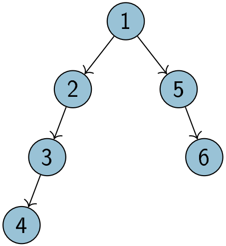
- Празното дърво $\perp$ е балансирано
- (X, L, R) е балансирано, ако
  - $|h(L) - h(R)| \leq 1$
  - L и R също са балансирани

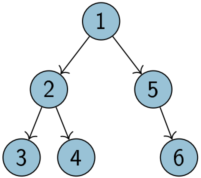


Балансирано дърво

Дефиниция (Балансирано дърво)

- Празното дърво $\perp$ е балансирано
- (X, L, R) е балансирано, ако
  - $|h(L) - h(R)| \leq 1$
  - L и R също са балансирани





Оптимална височина на балансирано дърво

Теорема

За балансирани дървета височината е възможно най-малка

Оптимална височина на балансирано дърво

Теорема

За балансирани дървета височината е възможно най-малка, т.е. $h = \lceil log_2(n+1) \rceil$.

Оптимална височина на балансирано дърво

Теорема

За балансирани дървета височината е възможно най-малка, т.е. $h = \lceil log_2(n+1) \rceil$.

Обратното вярно ли е?

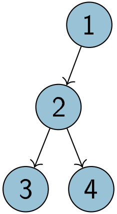
Оптимална височина на балансирано дърво

Теорема

За балансирани дървета височината е възможно най-малка, т.е. $h = \lceil log_2(n+1) \rceil$.

Обратното вярно ли е?

Не!



Идеално балансирано дърво

Дефиниция (Идеално балансирано дърво)

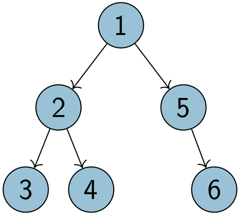
- Празното дърво $\perp$ е идеално балансирано
- (X, L, R) е идеално балансирано, ако
  - $|s(L) - s(R)| \leq 1$, където $s(T)$ означава броя на възлите в T
  - L и R също са идеално балансирани

Идеално балансирано дърво

Дефиниция (Идеално балансирано дърво)

- Празното дърво $\perp$ е идеално балансирано
- (X, L, R) е идеално балансирано, ако
  - $|s(L) - s(R)| \leq 1$, където $s(T)$ означава броя на възлите в T
  - L и R също са идеално балансирани

Пример:



Балансирани и идеално балансирани дървета

Каква е връзката между балансирани и идеално балансирани дървета?

Балансирани и идеално балансирани дървета

Каква е връзката между балансирани и идеално балансирани дървета?

Теорема

Всяко идеално балансирано дърво е балансирано.

Балансирани и идеално балансирани дървета

Каква е връзката между балансирани и идеално балансирани дървета?

Теорема

Всяко идеално балансирано дърво е балансирано.

Доказателство.

Индукция по височината на дървото. □

Балансирани и идеално балансирани дървета

Каква е връзката между балансирани и идеално балансирани дървета?

Теорема

Всяко идеално балансирано дърво е балансирано.

Доказателство.

Индукция по височината на дървото. □

Обратното вярно ли е?

Балансирани и идеално балансирани дървета

Каква е връзката между балансирани и идеално балансирани дървета?

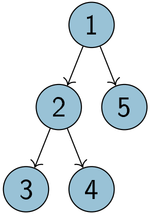
Теорема

Всяко идеално балансирано дърво е балансирано.

Доказателство.

Индукция по височината на дървото. □

Обратното вярно ли е? Не:



Построяване на идеално балансирано дърво

По даден сортиран списък можем да построим идеално балансирано двоично дърво за търсене.

Строим рекурсивно:

- Избираме за корен X “средния” елемент на списъка
- Лявото поддърво строим от подсписъка вляво от “средния” елемент
- Дясното поддърво строим от подсписъка вдясно от “средния” елемент
- Двата подсписъка имат приблизително равни дължини
- Рекурсията ни гарантира идеална балансираност

1	2	3	4	5	6	7	8	9	10	11	12

Самобалансиращи се дървета за търсене

Можем да постигнем сложност $O(\log n)$ на операциите търсене, включване и изключване, ако работим само с балансирани дървета.

Самобалансиращи се дървета за търсене

Можем да постигнем сложност $O(\log n)$ на операциите търсене, включване и изключване, ако работим само с балансирани дървета.

Идея: ако дървото се разбалансира след включване или изключване, да го балансираме наново.

Самобалансиращи се дървета за търсене

Можем да постигнем сложност $O(\log n)$ на операциите търсене, включване и изключване, ако работим само с балансирани дървета.

Идея: ако дървото се разбалансира след включване или изключване, да го балансираме наново.

Има различни вариации на самобалансиращи се дървета:

- 2-3 дърво
- AVL дърво
- червено-черно дърво
- косо дърво (splay tree)
- Декартово дърво (treap)

Самобалансиращи се дървета за търсене

Можем да постигнем сложност $O(\log n)$ на операциите търсене, включване и изключване, ако работим само с балансирани дървета.

Идея: ако дървото се разбалансира след включване или изключване, да го балансираме наново.

Има различни вариации на самобалансиращи се дървета:

- 2-3 дърво
- AVL дърво
- червено-черно дърво
- косо дърво (splay tree)
- Декартово дърво (treap)

AVL дърво

Предложено от Адельсон-Велский и Ландис през 1962 г.

Основна идея: Всяко поддърво $T = (X, L, R)$ поддържа коефициент на баланс:

$$b(T) = h(R) - h(L)$$

AVL дърво

Предложено от Аделсон-Велский и Ландис през 1962 г.

Основна идея: Всяко поддърво $T = (X, L, R)$ поддържа коефициент на баланс:

$$b(T) = h(R) - h(L)$$

Едно AVL дърво T е балансирано

$$\Longleftrightarrow$$

$b(T') \in \{-1, 0, 1\}$ за всяко поддърво T' на T

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$
  - $b(T) = -2$ — лявото поддърво е с 2 нива по-високо от дясното

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$
  - $b(T) = -2$ — лявото поддърво е с 2 нива по-високо от дясното
  - $b(T) = 2$ — дясното поддърво е с 2 нива по-високо от лявото

Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$
  - $b(T) = -2$ — лявото поддърво е с 2 нива по-високо от дясното
  - $b(T) = 2$ — дясното поддърво е с 2 нива по-високо от лявото
- Дефинираме операции за "завъртане", които възстановяват баланса.

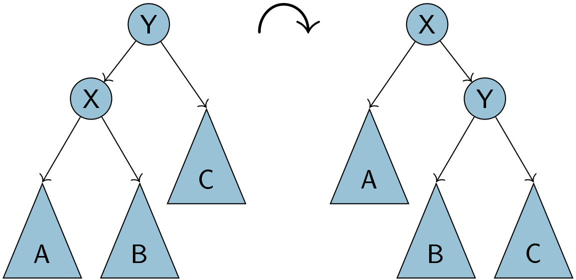
Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$
  - $b(T) = -2$ — лявото поддърво е с 2 нива по-високо от дясното
    - "завъртаме надясно" за да балансираме
  - $b(T) = 2$ — дясното поддърво е с 2 нива по-високо от лявото
- Дефинираме операции за "завъртане", които възстановяват баланса.

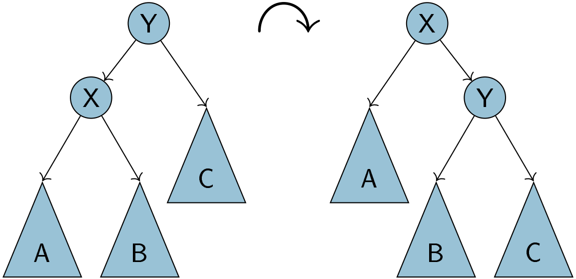
Самобалансиране

- Операциите за включване и изключване може да променят баланса на някой възел!
- Промяната няма как да е с повече от ± 1 (защо?)
- Разбалансиране се получава при $b(T) = \pm 2$
  - $b(T) = -2$ — лявото поддърво е с 2 нива по-високо от дясното
    - "завъртаме надясно" за да балансираме
  - $b(T) = 2$ — дясното поддърво е с 2 нива по-високо от лявото
    - "завъртаме наляво" за да балансираме
- Дефинираме операции за "завъртане", които възстановяват баланса.

Завъртане надясно (zig)

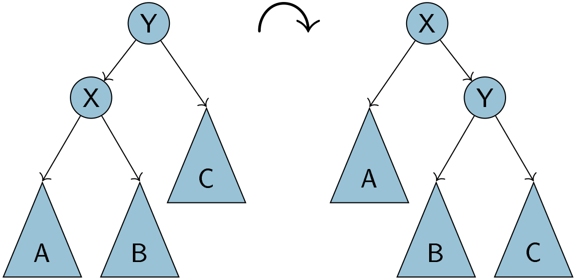


Завъртане надясно (zig)



$$b'(Y) = ?$$

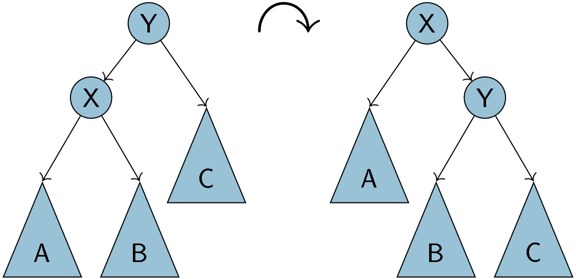
Завъртане надясно (zig)



$$b'(Y) = ?$$

I сл. $b(X) \geq 0$

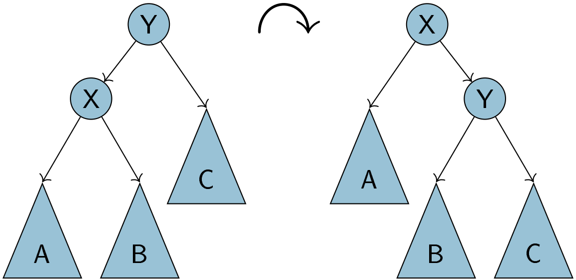
Завъртане надясно (zig)



$$b'(Y) = ?$$

I сл. $b(X) \geq 0 \Rightarrow h(B) \geq h(A)$

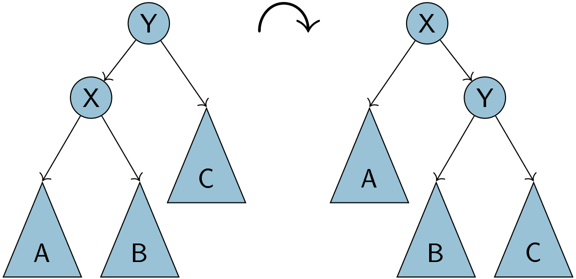
Завъртане надясно (zig)



$$b'(Y) = ?$$

I сл. $b(X) \geq 0 \Rightarrow h(B) \geq h(A) \Rightarrow h(X) = h(B) + 1$

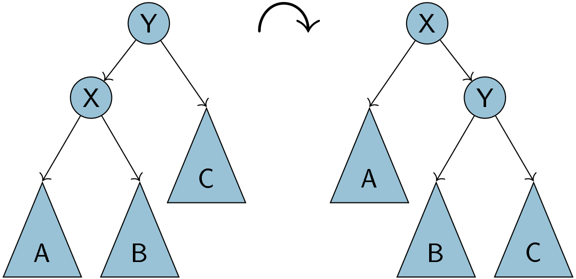
Завъртане надясно (zig)



$$b'(Y) = ?$$

I сл. $b(X) \geq 0 \Rightarrow h(B) \geq h(A) \Rightarrow h(X) = h(B) + 1 \Rightarrow b(Y) = h(C) - h(X)$

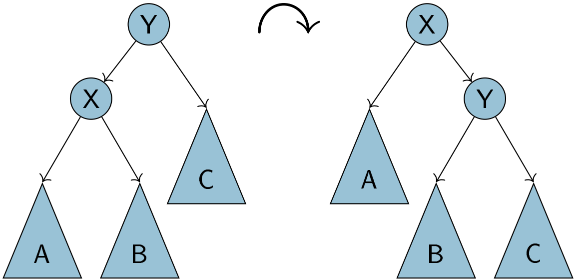
Завъртане надясно (zig)



$$b'(Y) = ?$$

I сл. $b(X) \geq 0 \Rightarrow h(B) \geq h(A) \Rightarrow h(X) = h(B) + 1 \Rightarrow b(Y) = h(C) - h(X) = \underbrace{h(C) - h(B)}_{b'(Y)} - 1$

Завъртане надясно (zig)

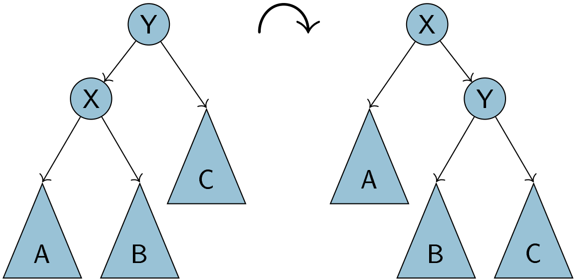


$$b'(Y) = ?$$

I сл. $b(X) \geq 0 \Rightarrow h(B) \geq h(A) \Rightarrow h(X) = h(B) + 1 \Rightarrow b(Y) = h(C) - h(X) = \underbrace{h(C) - h(B)}_{b'(Y)} - 1$

$\Rightarrow b'(Y) = b(Y) + 1$

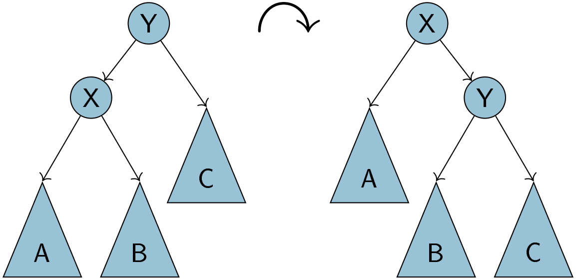
Завъртане надясно (zig)



$$b'(Y) = ?$$

II сл. $b(X) < 0$

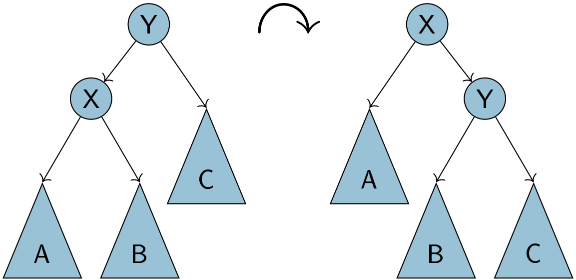
Завъртане надясно (zig)



$$b'(Y) = ?$$

II сл. $b(X) < 0 \Rightarrow h(B) < h(A)$

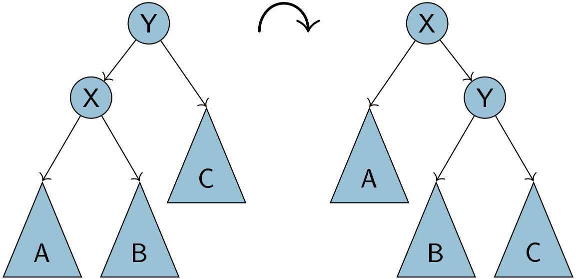
Завъртане надясно (zig)



$$b'(Y) = ?$$

II сл. $b(X) < 0 \Rightarrow h(B) < h(A) \Rightarrow h(X) = h(A) + 1$

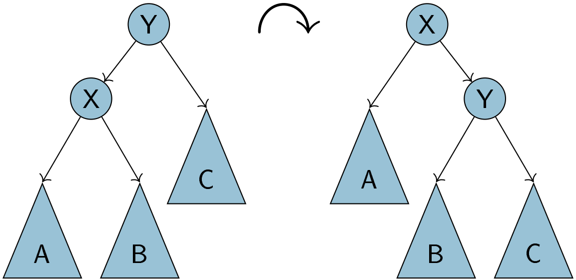
Завъртане надясно (zig)



$$b'(Y) = ?$$

II сл. $b(X) < 0 \Rightarrow h(B) < h(A) \Rightarrow h(X) = h(A) + 1 \Rightarrow b(Y) = h(C) - h(A) - 1$

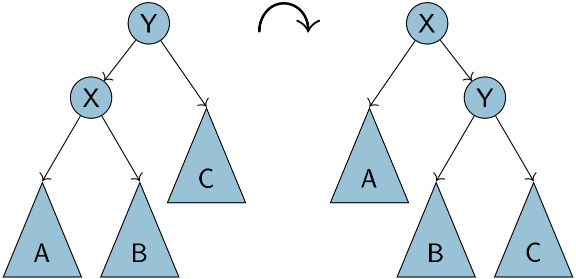
Завъртане надясно (zig)



$$b'(Y) = ?$$

II сл. $b(X) < 0 \Rightarrow h(B) < h(A) \Rightarrow h(X) = h(A) + 1 \Rightarrow b(Y) = h(C) - h(A) - 1$
$= \underbrace{h(C) - h(B)}_{b'(Y)} + \underbrace{h(B) - h(A)}_{b(X)} - 1$

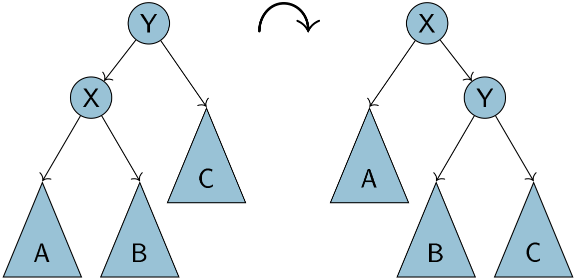
Завъртане надясно (zig)



$$b'(Y) = ?$$

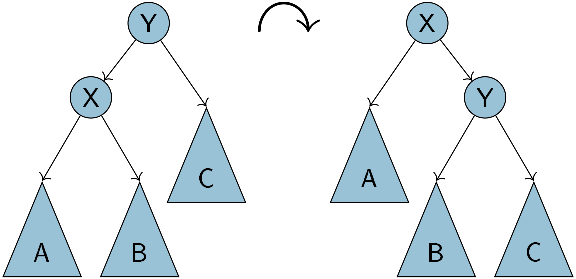
II сл. $b(X) < 0 \Rightarrow h(B) < h(A) \Rightarrow h(X) = h(A) + 1 \Rightarrow b(Y) = h(C) - h(A) - 1$
$= \underbrace{h(C) - h(B)}_{b'(Y)} + \underbrace{h(B) - h(A)}_{b(X)} -1 \Rightarrow b'(Y) = b(Y) - b(X) + 1$

Завъртане надясно (zig)



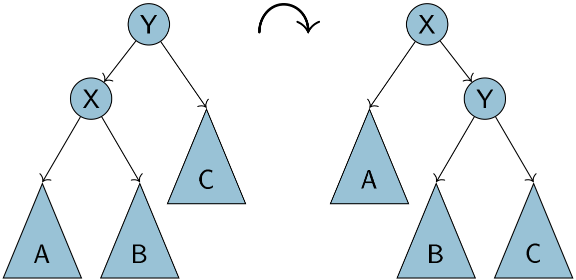
$$b'(Y) = \begin{cases} b(Y) + 1, & \text{ако } b(X) \geq 0, \\ b(Y) - b(X) + 1, & \text{ако } b(X) < 0. \end{cases}$$

Завъртане надясно (zig)



$$b'(X) = ?$$

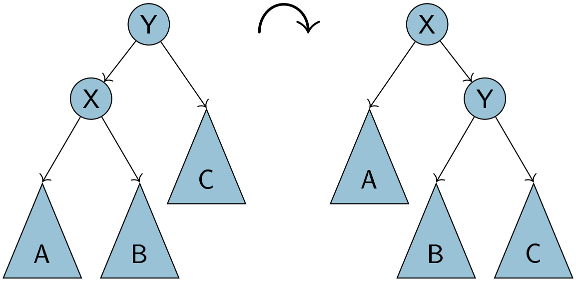
Завъртане надясно (zig)



$$b'(X) = ?$$

I сл. $b'(Y) \leq 0$

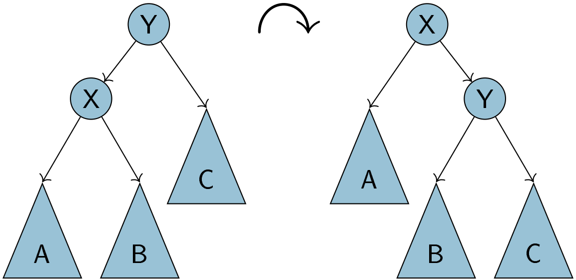
Завъртане надясно (zig)



$$b'(X) = ?$$

I сл. $b'(Y) \leq 0 \Rightarrow h(B) \geq h(C)$

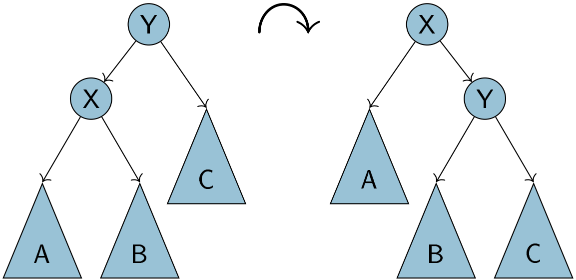
Завъртане надясно (zig)



$$b'(X) = ?$$

I сл. $b'(Y) \leq 0 \Rightarrow h(B) \geq h(C) \Rightarrow h'(Y) = h(B) + 1$

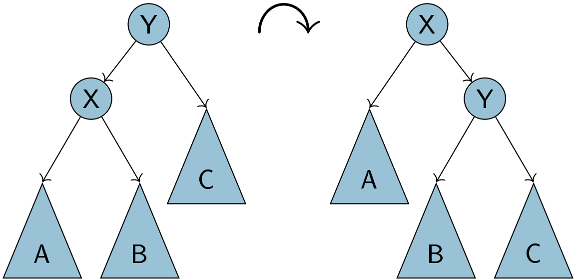
Завъртане надясно (zig)



$$b'(X) = ?$$

I сл. $b'(Y) \leq 0 \Rightarrow h(B) \geq h(C) \Rightarrow h'(Y) = h(B) + 1 \Rightarrow b'(X) = h'(Y) - h(A)$

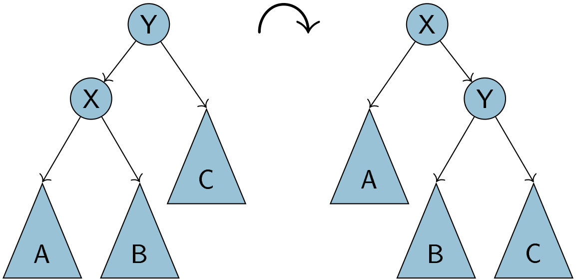
Завъртане надясно (zig)



$$b'(X) = ?$$

I сл. $b'(Y) \leq 0 \Rightarrow h(B) \geq h(C) \Rightarrow h'(Y) = h(B) + 1 \Rightarrow b'(X) = h'(Y) - h(A) = \underbrace{h(B) - h(A)}_{b(X)} + 1$

Завъртане надясно (zig)

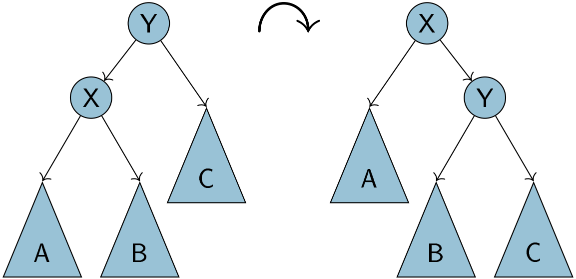


$$b'(X) = ?$$

I сл. $b'(Y) \leq 0 \Rightarrow h(B) \geq h(C) \Rightarrow h'(Y) = h(B) + 1 \Rightarrow b'(X) = h'(Y) - h(A) = \underbrace{h(B) - h(A)}_{b(X)} + 1$

$\Rightarrow b'(X) = b(X) + 1$

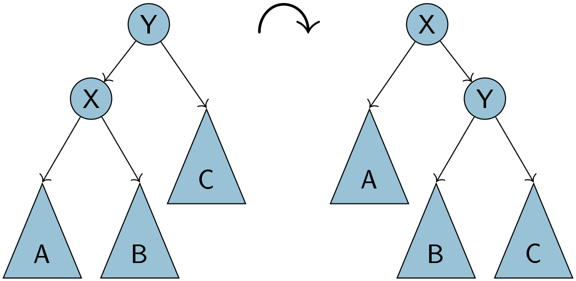
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0$

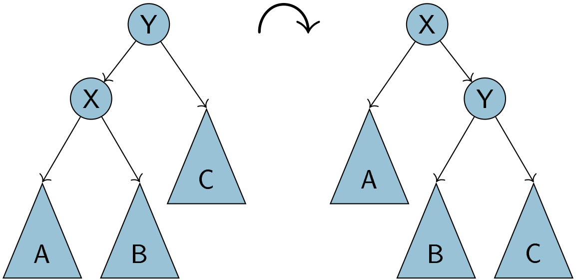
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0 \Rightarrow h(B) < h(C)$

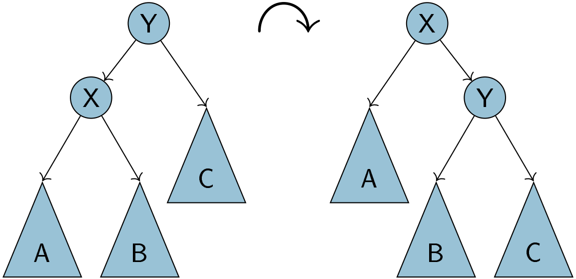
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0 \Rightarrow h(B) < h(C) \Rightarrow h'(Y) = h(C) + 1$

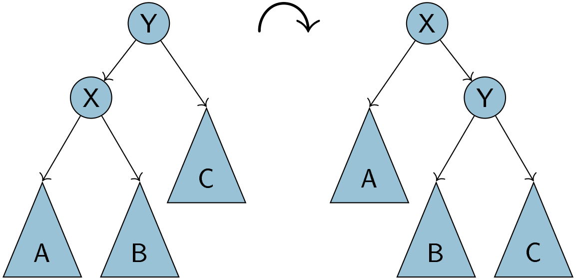
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0 \Rightarrow h(B) < h(C) \Rightarrow h'(Y) = h(C) + 1 \Rightarrow b'(X) = h(C) + 1 - h(A)$

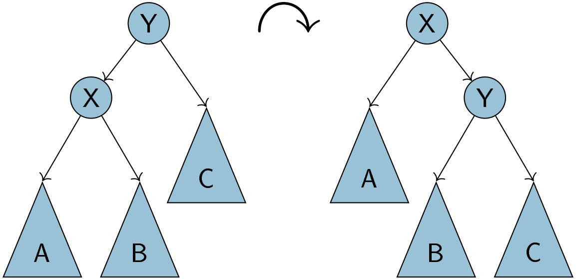
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0 \Rightarrow h(B) < h(C) \Rightarrow h'(Y) = h(C) + 1 \Rightarrow b'(X) = h(C) + 1 - h(A)$
$= \underbrace{h(C) - h(B)}_{b'(Y)} + \underbrace{h(B) - h(A)}_{b(X)} + 1$

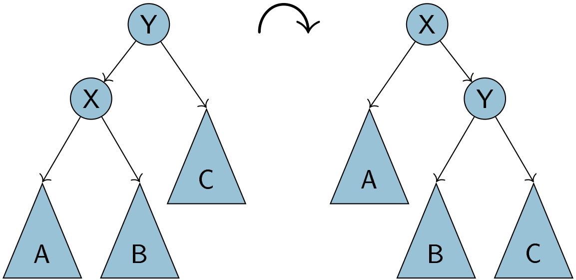
Завъртане надясно (zig)



$$b'(X) = ?$$

II сл. $b'(Y) > 0 \Rightarrow h(B) < h(C) \Rightarrow h'(Y) = h(C) + 1 \Rightarrow b'(X) = h(C) + 1 - h(A)$
$= \underbrace{h(C) - h(B)}_{b'(Y)} + \underbrace{h(B) - h(A)}_{b(X)} + 1 \Rightarrow b'(X) = b(X) + b'(Y) + 1$

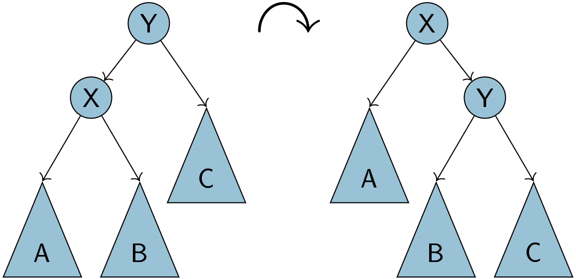
Завъртане надясно (zig)



$$b'(Y) = \begin{cases} b(Y) + 1, & \text{ако } b(X) \geq 0, \\ b(Y) - b(X) + 1, & \text{ако } b(X) < 0. \end{cases}$$

$$b'(X) = \begin{cases} b(X) + 1, & \text{ако } b'(Y) \leq 0, \\ b(X) + b'(Y) + 1, & \text{ако } b'(Y) > 0. \end{cases}$$

Завъртане надясно (zig)

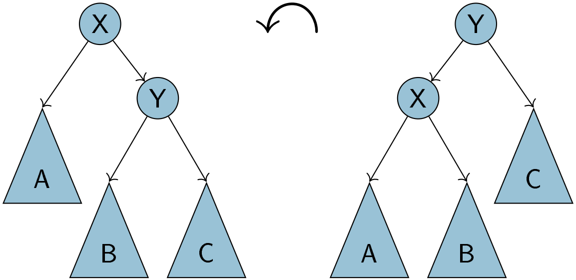


$$b'(Y) = \begin{cases} b(Y) + 1, & \text{ако } b(X) \geq 0, \\ b(Y) - b(X) + 1, & \text{ако } b(X) < 0. \end{cases}$$

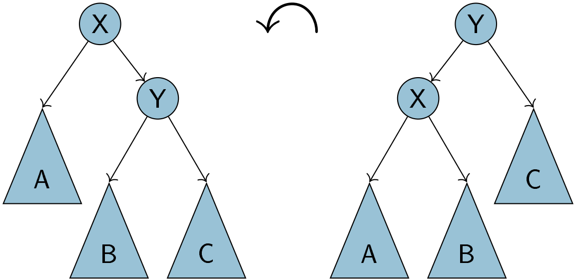
$$b'(X) = \begin{cases} b(X) + 1, & \text{ако } b'(Y) \leq 0, \\ b(X) + b'(Y) + 1, & \text{ако } b'(Y) > 0. \end{cases}$$

$$b'(X) > b(X), \quad b'(Y) > b(Y)$$

Завъртане наляво (zag)



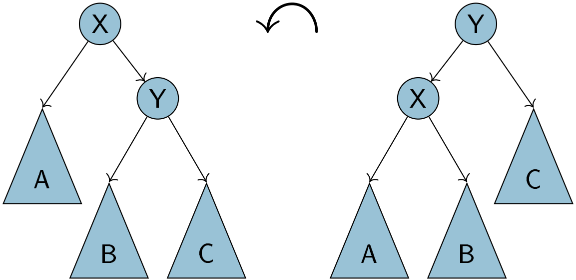
Завъртане наляво (zag)



$$b'(Y) = \begin{cases} b(Y)+1, & \text{ако } b(X) \geq 0, \\ b(Y)-b(X)+1, & \text{ако } b(X) < 0. \end{cases}$$

$$b'(X) = \begin{cases} b(X)+1, & \text{ако } b'(Y) \leq 0, \\ b(X)+b'(Y)+1, & \text{ако } b'(Y) > 0. \end{cases}$$

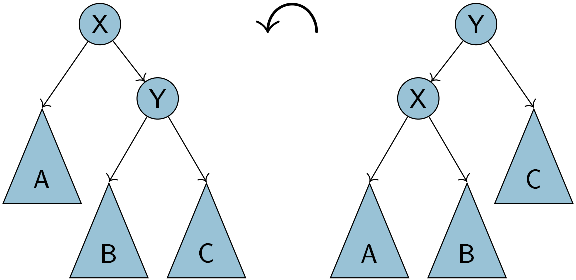
Завъртане наляво (zag)



$$b(Y) = \begin{cases} b'(Y) + 1, & \text{ако } b'(X) \geq 0, \\ b'(Y) - b'(X) + 1, & \text{ако } b'(X) < 0. \end{cases}$$

$$b(X) = \begin{cases} b'(X) + 1, & \text{ако } b(Y) \leq 0, \\ b'(X) + b(Y) + 1, & \text{ако } b(Y) > 0. \end{cases}$$

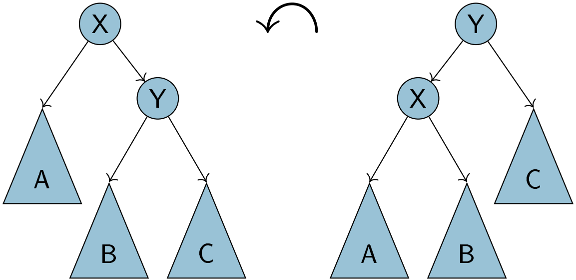
Завъртане наляво (zag)



$$b'(Y) = \begin{cases} b(Y) - 1, & \text{ако } b'(X) \geq 0, \\ b(Y) + b'(X) - 1, & \text{ако } b'(X) < 0. \end{cases}$$

$$b'(X) = \begin{cases} b(X) - 1, & \text{ако } b(Y) \leq 0, \\ b(X) - b(Y) - 1, & \text{ако } b(Y) > 0. \end{cases}$$

Завъртане наляво (zag)

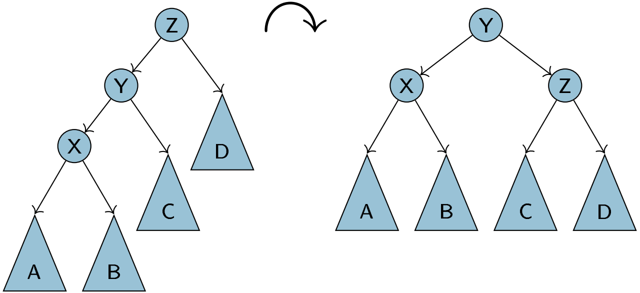


$$b'(Y) = \begin{cases} b(Y) - 1, & \text{ако } b'(X) \geq 0, \\ b(Y) + b'(X) - 1, & \text{ако } b'(X) < 0. \end{cases}$$

$$b'(X) = \begin{cases} b(X) - 1, & \text{ако } b(Y) \leq 0, \\ b(X) - b(Y) - 1, & \text{ако } b(Y) > 0. \end{cases}$$

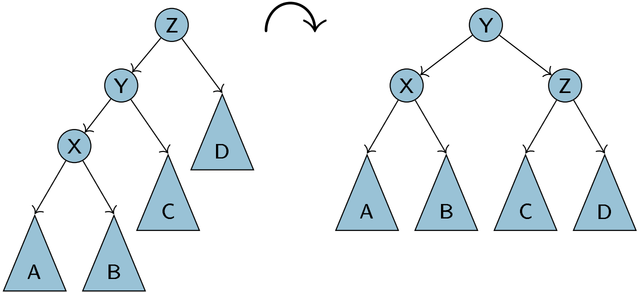
$$b'(X) < b(X), \quad b'(Y) < b(Y)$$

Балансиране надясно



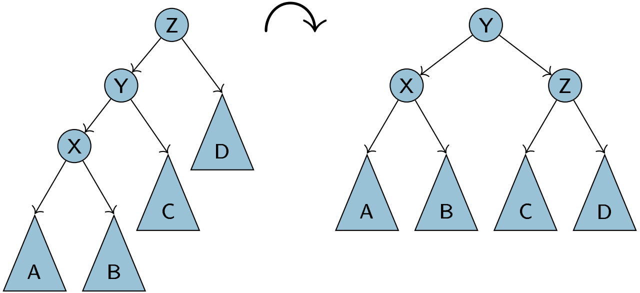
- Въртим надясно, ако $b(Z) = -2$

Балансиране надясно



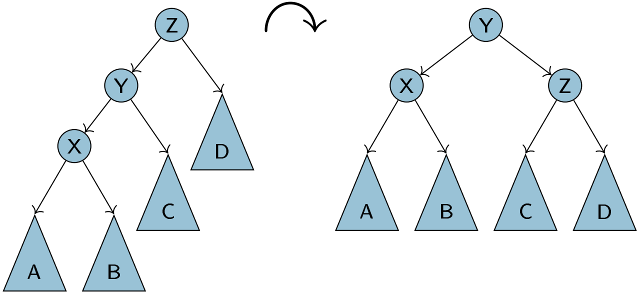
- Въртим надясно, ако $b(Z) = -2$
- Внимание: Ако $b(Y) = 1$, то

Балансиране надясно



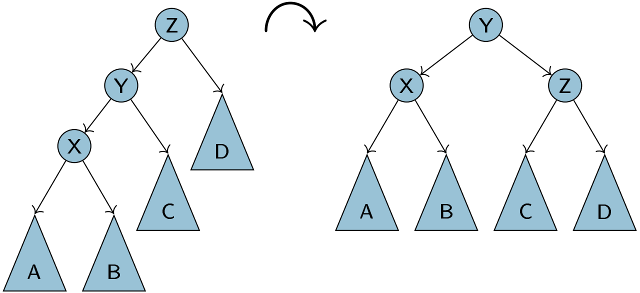
- Въртим надясно, ако $b(Z) = -2$
- Внимание: Ако $b(Y) = 1$, то
    - $b'(Z) = b(Z) + 1 = -1$

Балансиране надясно



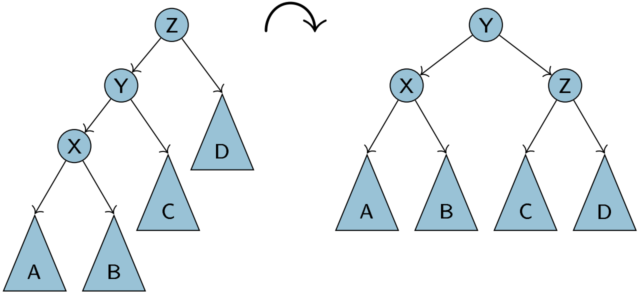
- Въртим надясно, ако $b(Z) = -2$
- Внимание: Ако $b(Y) = 1$, то
  - $b'(Z) = b(Z) + 1 = -1$
  - $b'(Y) = b(Y) + 1 = 2$

Балансиране надясно



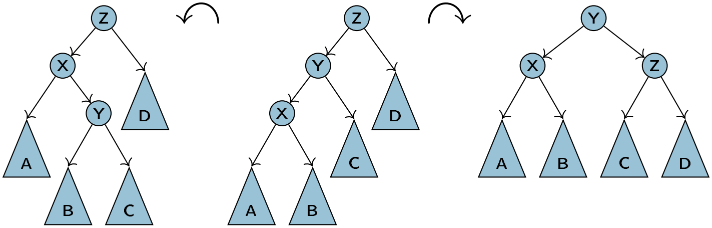
- Въртим надясно, ако $b(Z) = -2$
- Внимание: Ако $b(Y) = 1$, то
  - $b'(Z) = b(Z) + 1 = -1$
  - $b'(Y) = b(Y) + 1 = 2$
- Трябва да подсигурим, че $b(Y) \leq 0$...

Балансиране надясно



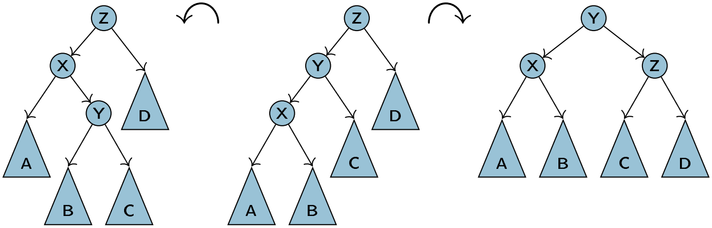
- Въртим надясно, ако $b(Z) = -2$
- Внимание: Ако $b(Y) = 1$, то
  - $b'(Z) = b(Z) + 1 = -1$
  - $b'(Y) = b(Y) + 1 = 2$
- Трябва да подсигурим, че $b(Y) \leq 0$...
- ...с предварително завъртане наляво!

Двойно балансиране надясно (zag-zig)



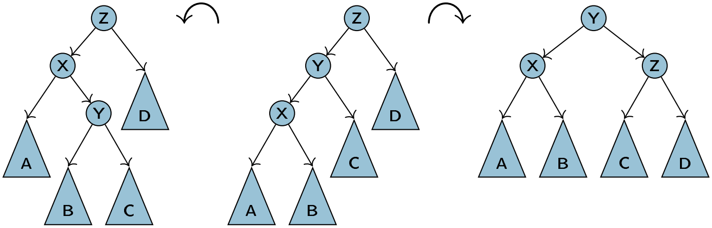
- Ако $b(X) = 1$, първо завъртаме наляво около X.

Двойно балансиране надясно (zag-zig)



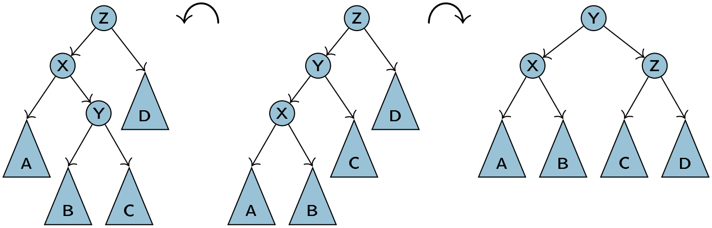
- Ако $b(X) = 1$, първо завъртаме наляво около X.
- Така $b'(X) \leq 0$ и $b'(Y) \leq 0$

Двойно балансиране надясно (zag-zig)



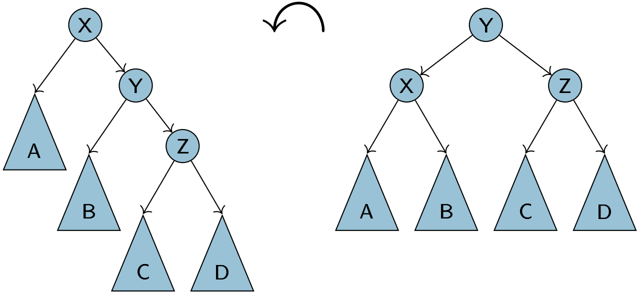
- Ако $b(X) = 1$, първо завъртаме наляво около X.
- Така $b'(X) \leq 0$ и $b'(Y) \leq 0$
- Вече можем да завъртим надясно около Y.

Двойно балансиране надясно (zag-zig)



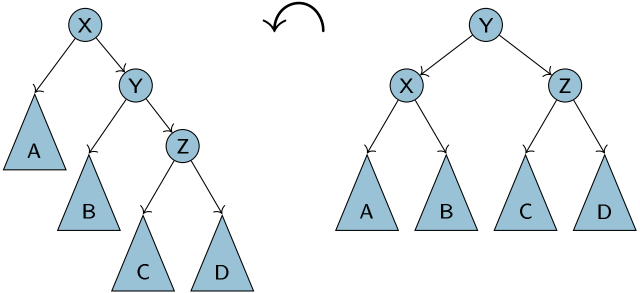
- Ако $b(X) = 1$, първо завъртаме наляво около X.
- Така $b'(X) \leq 0$ и $b'(Y) \leq 0$
- Вече можем да завъртим надясно около Y.
- След балансиране сме сигурни, че $h'(Y) < h(Z)$, т.е. намаляваме височината.

Балансиране наляво



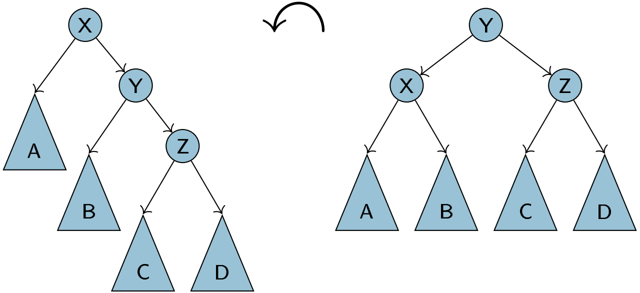
- Въртим наляво, ако $b(Z) = 2$

Балансиране наляво



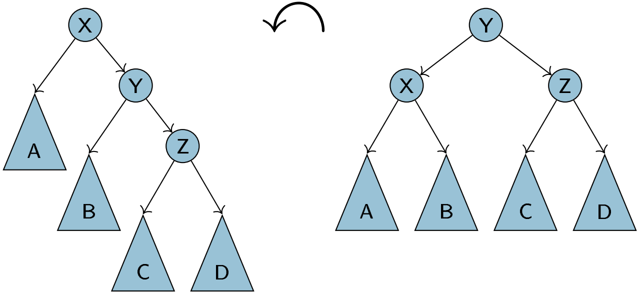
- Въртим наляво, ако $b(Z) = 2$
- Внимание: Ако $b(Y) = -1$, то

Балансиране наляво



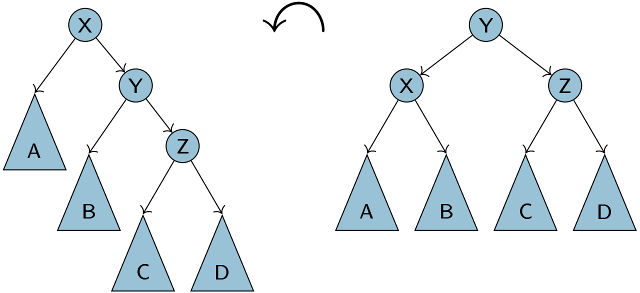
- Въртим наляво, ако $b(Z) = 2$
- Внимание: Ако $b(Y) = -1$, то
  - $b'(Z) = b(Z) - 1 = 1$

Балансиране наляво



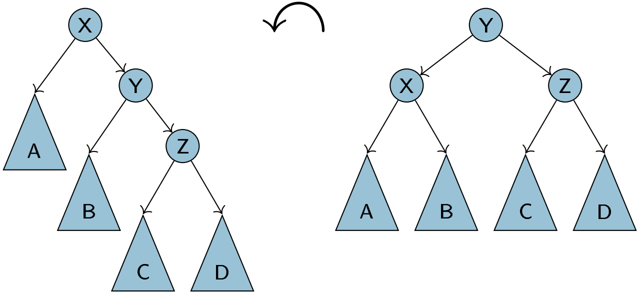
- Въртим наляво, ако $b(Z) = 2$
- Внимание: Ако $b(Y) = -1$, то
    - $b'(Z) = b(Z) - 1 = 1$
    - $b'(Y) = b(Y) - 1 = -2$

Балансиране наляво



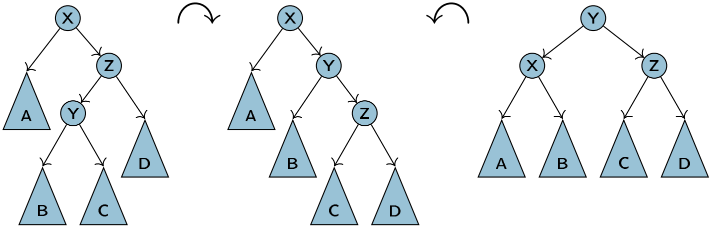
- Въртим наляво, ако $b(Z) = 2$
- Внимание: Ако $b(Y) = -1$, то
  - $b'(Z) = b(Z) - 1 = 1$
  - $b'(Y) = b(Y) - 1 = -2$
- Трябва да подсигурим, че $b(Y) \geq 0$...

Балансиране наляво



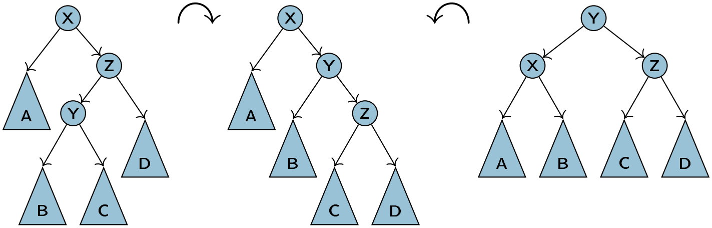
- Въртим наляво, ако $b(Z) = 2$
- Внимание: Ако $b(Y) = -1$, то
  - $b'(Z) = b(Z) - 1 = 1$
  - $b'(Y) = b(Y) - 1 = -2$
- Трябва да подсигурим, че $b(Y) \geq 0$...
- ...с предварително завъртане надясно!

Двойно балансиране наляво (zig-zag)



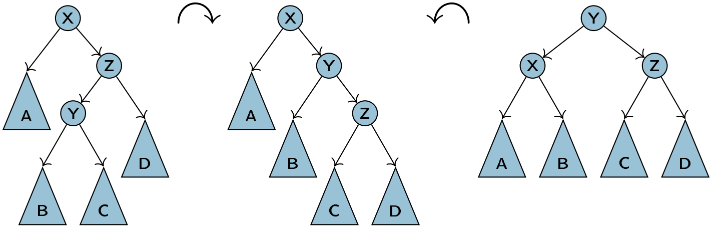
- Ако $b(Z) = -1$, първо завъртаме надясно около Z.

Двойно балансиране наляво (zig-zag)



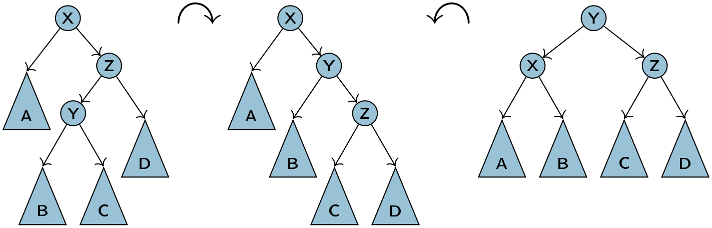
- Ако $b(Z) = -1$, първо завъртаме надясно около Z.
- Така $b'(Z) \geq 0$ и $b'(Y) \geq 0$

Двойно балансиране наляво (zig-zag)



- Ако $b(Z) = -1$, първо завъртаме надясно около Z.
- Така $b'(Z) \geq 0$ и $b'(Y) \geq 0$
- Вече можем да завъртим наляво около Y.

Двойно балансиране наляво (zig-zag)



- Ако $b(Z) = -1$, първо завъртаме надясно около Z.
- Така $b'(Z) \geq 0$ и $b'(Y) \geq 0$
- Вече можем да завъртим наляво около Y.
- След балансиране сме сигурни, че $h'(Y) < h(X)$, т.е. намаляваме височината.

Кога да балансираме?

- Когато включваме или изключваме елемент, трябва да следим кога балансът се променя

Кога да балансираме?

- Когато включваме или изключваме елемент, трябва да следим кога балансът се променя
- При промяна на баланс ще трябва да пребалансираме

Кога да балансираме?

- Когато включваме или изключваме елемент, трябва да следим кога балансът се променя
- При промяна на баланс ще трябва да пребалансираме
- Затова ще реализираме включването и изключването **рекурсивно**

Кога да балансираме?

- Когато включваме или изключваме елемент, трябва да следим кога балансът се променя
- При промяна на баланс ще трябва да пребалансираме
- Затова ще реализираме включването и изключването **рекурсивно**
- На обратния ход на рекурсията ще пребалансираме при нужда

Кога да балансираме?

- Когато включваме или изключваме елемент, трябва да следим кога балансът се променя
- При промяна на баланс ще трябва да пребалансираме
- Затова ще реализираме включването и изключването **рекурсивно**
- На обратния ход на рекурсията ще пребалансираме при нужда
- Балансът се променя когато височината на детето се е увеличила или намалила

Балансиране при включване и изключване

Балансиране при включване

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при изключване

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при изключване

- При дъното на **изключването** дъното височината винаги се намалява с 1

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при изключване

- При дъното на **изключването** дъното височината винаги се намалява с 1
- Ако височината на **по-високото дете** се намали, то височината на родителя не се променя

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при изключване

- При дъното на **изключването** дъното височината винаги се намалява с 1
- Ако височината на **по-високото дете** се намали, то височината на родителя не се променя
- Ако след балансиране $b(T) \neq 0$, значи сме компенсирали за намалената височина на детето

Балансиране при включване и изключване

Балансиране при включване

- При дъното на **включването** височината винаги се увеличава с 1
- Ако височината на **по-ниското дете** се увеличи, то височината на родителя не се променя
- При балансиране винаги компенсираме за увеличената височина на детето

Балансиране при изключване

- При дъното на **изключването** дъното височината винаги се намалява с 1
- Ако височината на **по-високото дете** се намали, то височината на родителя не се променя
- Ако след балансиране $b(T) \neq 0$, значи сме компенсирали за намалената височина на детето
- Ако след балансиране $b(T) = 0$, значи височината се е намалила

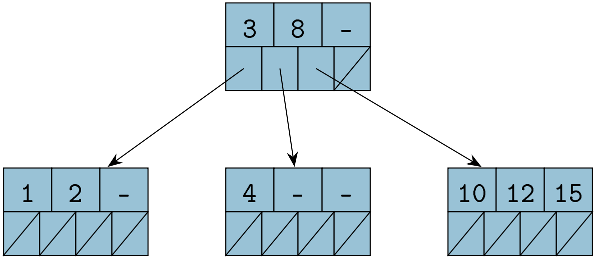
B-дървета

Дефиниция (B-дърво)

B-дърво от ред n наричаме n-арно дърво ($n \geq 3$), за което:

- всички листа са на еднаква височина
- всеки възел съдържа най-много $n-1$ ключа подредени в нарастващ ред
- всеки възел (освен корена) съдържа най-малко $\left\lfloor \frac{n-1}{2} \right\rfloor$ ключа
- всеки възел с m ключа $k_1, \ldots, k_m$ или няма поддървета, или има точно $m+1$ непразни поддървета $T_0, \ldots, T_m$, които са разположени максимално вляво (т.е. $T_{m+1}, \ldots, T_n$ са празни).
- k_i е по-голям от всички ключове в поддърветата T_j за $j \leq i$
- k_i е по-малък от всички ключове в поддърветата T_j за $j > i$

Пример за B-дърво от ред 4



Включване на елемент

- Прави се опит за включване на елемент в някое листо

Включване на елемент

- Прави се опит за включване на елемент в някое листо
- Ако се окаже, че се опитваме да включим елемент в листо, което вече е пълно с $n-1$ ключа:

Включване на елемент

- Прави се опит за включване на елемент в някое листо
- Ако се окаже, че се опитваме да включим елемент в листо, което вече е пълно с $n-1$ ключа:
  - Разцепваме възела на два други с приблизително еднакъв брой елементи

Включване на елемент

- Прави се опит за включване на елемент в някое листо
- Ако се окаже, че се опитваме да включим елемент в листо, което вече е пълно с $n-1$ ключа:
  - Разцепваме възела на два други с приблизително еднакъв брой елементи
  - Средния по големина ключ се опитваме да вмъкнем в родителя

Включване на елемент

- Прави се опит за включване на елемент в някое листо
- Ако се окаже, че се опитваме да включим елемент в листо, което вече е пълно с $n-1$ ключа:
    - Разцепваме възела на два други с приблизително еднакъв брой елементи
    - Средния по големина ключ се опитваме да вмъкнем в родителя
    - Ако в родителя вече има $n-1$ ключа, повтаряме същата схема

Включване на елемент

- Прави се опит за включване на елемент в някое листо
- Ако се окаже, че се опитваме да включим елемент в листо, което вече е пълно с $n - 1$ ключа:
  - Разцепваме възела на два други с приблизително еднакъв брой елементи
  - Средния по големина ключ се опитваме да вмъкнем в родителя
  - Ако в родителя вече има $n - 1$ ключа, повтаряме същата схема
  - Ако стигнем до корена, правим нов корен само с един ключ и две поддървета

Изключване на елемент

- Първо намираме ключа K на елемента в дървото

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
    - Ако е в листо, изтриваме го
    - Ако е във вътрешен възел, заменяме го с:
        - най-малкия ключ $> K$, или
        - най-големия ключ $< K$
        - такъв ключ задължително ще се намира в листо

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа
  - Ако и двата съседа също съдържат минимален брой ключове

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа
  - Ако и двата съседа също съдържат минимален брой ключове
    - Листото се слива с някой от двата си съседа

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа
  - Ако и двата съседа също съдържат минимален брой ключове
    - Листото се слива с някой от двата си съседа
    - Понеже броят на поддърветата в родителя намалява с 1, прехвърляме в листото ключа, който е стоял между двете слети листа в родителя

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа
  - Ако и двата съседа също съдържат минимален брой ключове
    - Листото се слива с някой от двата си съседа
    - Понеже броят на поддърветата в родителя намалява с 1, прехвърляме в листото ключа, който е стоял между двете слети листа в родителя
    - Сигурни сме, че в новото листо броят ключове не надвишава максимума, понеже $2\left\lfloor\frac{n-1}{2}\right\rfloor \leq n-1$

Изключване на елемент

- Първо намираме ключа K на елемента в дървото
  - Ако е в листо, изтриваме го
  - Ако е във вътрешен възел, заменяме го с:
    - най-малкия ключ $> K$, или
    - най-големия ключ $< K$
    - такъв ключ задължително ще се намира в листо
- Ако броят на ключовете в листото падне под минимума:
  - Опитваме се да заемем ключ и съответно поддърво от някой от двата съседа
  - Ако и двата съседа също съдържат минимален брой ключове
    - Листото се слива с някой от двата си съседа
    - Понеже броят на поддърветата в родителя намалява с 1, прехвърляме в листото ключа, който е стоял между двете слети листа в родителя
    - Сигурни сме, че в новото листо броят ключове не надвишава максимума, понеже $2\left\lfloor\frac{n-1}{2}\right\rfloor \leq n-1$
- Ако сега броят на ключовете в родителя падне под минимума, повтаряме същата процедура за него