

1 Drzewiaste struktury danych

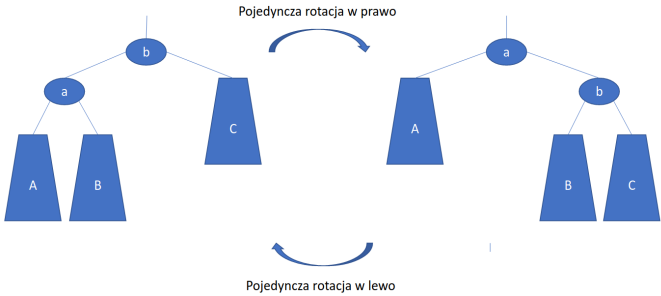
1.1 BST

- Drzewa wyszukiwań binarnych (drzewa BST – Binary Search Trees) - drzewa binarne, w których elementy rozmieszczone są w porządku symetrycznym.
- Właściwości BST:

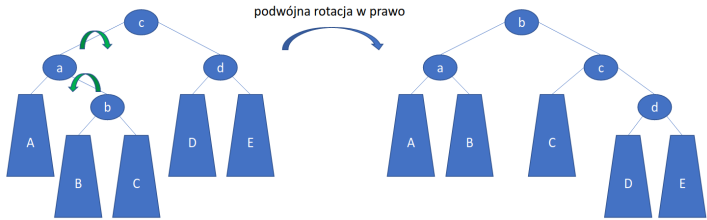
|                             | Drzewa binarne        |   | Drzewa wyszukiwań binarnych |   |
|-----------------------------|-----------------------|---|-----------------------------|---|
|                             | Wartość pesymistyczna | Wartość oczekiwana (każde drzewo jednako prawdopodobne) | Wartość pesymistyczna       | Wartość oczekiwana (drzewo powstaje poprzez wstawienie elementów losowej permutacji do początkowo pustego drzewa) |
| Długość ścieżki wewnętrznej | $n(n-1)/2$            | $n\sqrt{\pi n} - 3n + O(\sqrt{n})$                      | $n(n-1)/2$                  | $1,386n \log n - 2,846n$  |
| Wysokość drzewa             | $n-1$                 | $2\sqrt{\pi n} + O(\frac{1}{n^{3/4}})$                  | $n-1$                       | $4,311 \log n + o(\log n)$  |

1.2 AVL

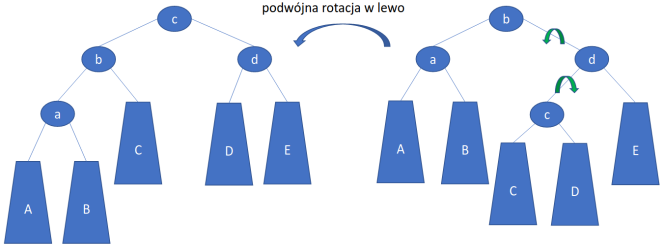
- AVL-drzewo – drzewo wyszukiwań binarnych, w którym dla każdego węzła wysokości jego poddrzew różnią się co najwyżej o 1
- Złożoność operacji to  $\log n$  gdzie  $n$  to wielkość drzewa.
- Atrybuty węzłów drzewa z wykładu:  
Key(v) – klucz  
Left(v), Right(v), Parent(v) – wskaźniki odpowiednio do lewego i prawego dziecka oraz rodzica  
Bf(v) – wskaźnik zrównoważenia (ang. balance factor) równy  $h(T(Left(v))) - h(T(right(v))) \in -1, 0, 1$
- Przyjmujemy, że węzeł zewnętrzny ma wysokość -1.
- Rotacje zachowują porządek symetryczny w BST, numerację infiksową (inorder) węzłów drzewa binarnego!
- Pojedyncza rotacja w prawo i lewo:



- Podwójna rotacja w prawo:



- Podwójna w lewo:

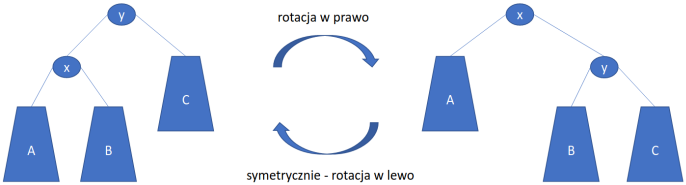


- Schemat wstawiania elementu gdy urosło lewe poddrzewo(symetrycznie dla prawego):  
Bf v = -1 ⇒ Bf v = 0; STOP  
Bf v = 0 ⇒ Bf v = 1; ↑ (poprawiaj w górę)  
Bf v = 1 ⇒ rotacja
- Schemat usuwania elementu gdy zmalało lewe poddrzewo(symetrycznie dla prawego):  
Bf v = -1 ⇒ rotacja  
Bf v = 0 ⇒ Bf v = -1; STOP  
Bf v = 1 ⇒ Bf v = 0; ↑ (poprawiaj w górę)
- Wzbogacenia AVL przedstawione na wykładzie:  
OnLeft(v) – liczba węzłów w lewym poddrzewie v  
Max(v) – maksimum z elementów w poddrzewie o korzeniu v
- Rotacje w drzewie zachowują kolejność infiksową(inorder)

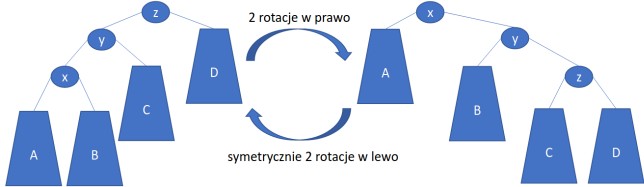
1.3 Splay

- Operacje na drzewach splay są w czasie zamortyzowanym logarytmicznym
- Podczas każdej operacji na drzewie splay wierzchołek, na którym wykonujemy operację, staje się korzeniem.
- Local Splay:

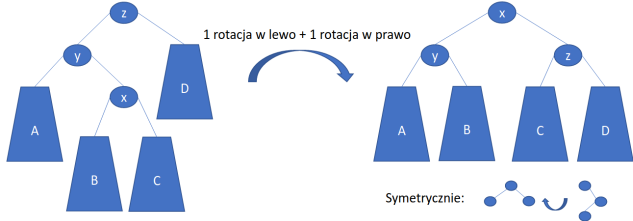
Przypadek 1: rodzic węzła x jest korzeniem drzewa  
B.o. utożsamiamy klucze z węzłami



Przypadek 2a: rodzicem x nie jest korzeń  
x jest lewym dzieckiem rodzica, który jest lewym dzieckiem swojego rodzica lub  
x jest prawym dzieckiem rodzica, który jest prawym dzieckiem swojego rodzica



Przypadek 2b: rodzicem x nie jest korzeń drzewa  
x jest prawym dzieckiem swojego rodzica, który jest lewym dzieckiem swojego rodzica  
x jest lewym dzieckiem swojego rodzica, który jest prawym dzieckiem swojego rodzica



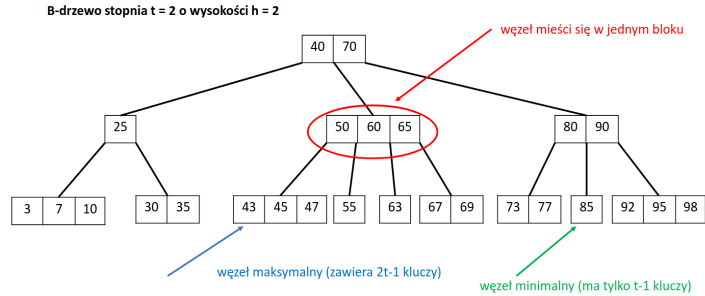
- Istnieje operacja split która dzieli drzewo na dwa poddrzewa według wierzchołka v. (lewe drzewo ma klucze mniejsze a prawe większe niż klucz v)

1.4 Drzewa B

- Niech t będzie liczbą całkowitą większą od 1.
- B-drzewem (minimalnego) stopnia t nazywamy drzewo z korzeniem spełniające poniższe warunki:
  - Każdy węzeł x ma następujące atrybuty:
    - a) m – liczba kluczy aktualnie pamiętanych w x
    - b) m kluczy Key1, Key2, ..., Keym uporządkowanych rosnąco (załóżmy ponadto, że mamy wirtualnych strażników Key0 = -∞ oraz Keym+1 = +∞)
    - c) Leaf – atrybut logiczny przyjmujący wartość TRUE w/w, gdy x jest liściem
  - Każdy węzeł wewnętrzny x zawiera m+1 wskaźników P1, P2, ..., Pm+1 do swoich dzieci w drzewie. Dla liści wartością tych wskaźników jest NULL.
  - Każdy klucz K w poddrzewie wskazywanym przez Pi spełnia warunek Keyi-1 < K < Keyi.
  - Wszystkie liście leżą na tej samej głębokości równej wysokości drzewa h.
  - Każdy węzeł różny od korzenia całego drzewa musi zawierać co najmniej t-1 i co najwyżej 2t-1 kluczy (odpowiednio, co najmniej t i co najwyżej 2t wskaźników do dzieci, o wartościach NULL, jeśli jest to jednocześnie liść).
  - W niepustym drzewie korzeń musi zawierać, co najmniej 1 i co najwyżej 2t-1 kluczy (odpowiednio co najmniej 2 i co najwyżej 2t wskaźników do dzieci - o wartościach NULL, jeśli jest to jednocześnie liść).
- Złożoności

| n-węzłowe B-drzewo stopnia t |                                  |                                  |
|------------------------------|----------------------------------|----------------------------------|
|                              | # odwołań do pamięci zewnętrznej | # operacji w pamięci wewnętrznej |
| Search                       | $O(\log_t n)$                    | $O(t \log_t n)$                  |
| Insert                       | $O(\log_t n)$                    | $O(t \log_t n)$                  |
| Delete                       | $O(\log_t n)$                    | $O(t \log_t n)$                  |

- B-drzewo o (minimalnym) stopniu  $t = 2$  nazywamy 2-3-4-drzewem (każdy węzeł, nie liść, ma 2 lub 3 lub 4 dzieci)
- Korzeń poddrzewa do którego wchodzimy, różny od korzenia, nigdy nie jest minimalny!
- Korzeń poddrzewa do którego wchodzimy nigdy nie jest maksymalny!
- Wysokość drzewa:  $h \leq \log_t(\frac{n+1}{2})$
- Przykład drzewa

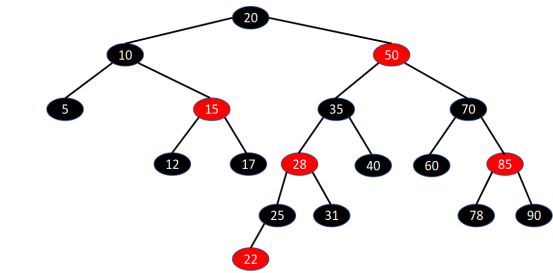


1.5 Drzewa Czerwono Czarne

- Drzewo czerwono-czarne – drzewo BST, w którym każdy węzeł jest pokolorowany na czerwono lub czarno zgodnie z następującymi regułami:
  1. korzeń drzewa jest czarny
  2. każdy czerwony węzeł ma czarnego rodzica
  3. każdy węzeł zewnętrzny (NULL) jest czarny
  4. każda ścieżka (elementarna) z ustalonego węzła do węzła zewnętrznego w jego poddrzewie zawiera tyle samo węzłów czarnych

- Wysokość drzewa wynosi  $h \leq 2 \log_2(\frac{n+1}{2})$
- Slajdy

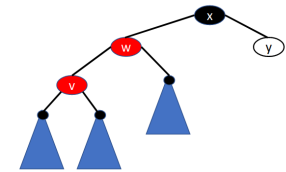
Drzewo czerwono-czarne



Wstawianie do drzewa czerwono-czarnego

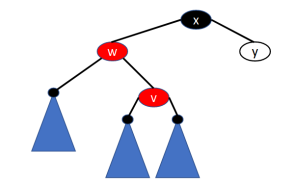
Nowy klucz zostaje umieszczony w nowym węźle v, który zastępuje węzeł zewnętrzny. Węzeł otrzymuje kolor czerwony.

- jeśli v jest korzeniem drzewa zmieniamy mu kolor na czarny
- jeśli rodzic P(v) węzeł v jest czarny, nic nie trzeba więcej robić
- jeśli rodzic P(v) jest czerwony, zaburzona została własność 2, drzew czerwono-czarnego



- y jest czerwony: w i y kolorujemy na czarno; jeśli x nie jest korzeniem, kolorujemy x na czerwono i poprawiamy kolorowanie rekurencyjnie w górę drzewa
- y jest czarny: pojedyncza rotacja w prawo (x zostaje prawym dzieckiem w); v otrzymuje kolor czarny; jeśli w jest korzeniem kolorujemy w na czarno, w przeciwnym razie poprawiamy kolorowanie rekurencyjnie w górę drzewa poczynając od w

Wstawianie do drzewa czerwono-czarnego, c.d.



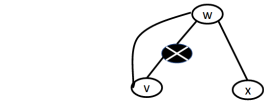
- y jest czerwony: w i y kolorujemy na czarno; jeśli x nie jest korzeniem, kolorujemy x na czerwono i poprawiamy kolorowanie rekurencyjnie w górę drzewa
- y jest czarny: podwójna rotacja w prawo (x zostaje prawym dzieckiem v, w – lewym dzieckiem v); w otrzymuje kolor czarny; jeśli v jest korzeniem kolorujemy v na czarno, w przeciwnym razie poprawiamy kolorowanie rekurencyjnie w górę drzewa poczynając od v

Symetrycznie postępujemy, gdy dwa czerwone węzły są w prawym poddrzewie x.

koszt Insert do drzewa czerwono-czarnego –  $O(\log n)$

Usuwanie z drzewa czerwono-czarnego

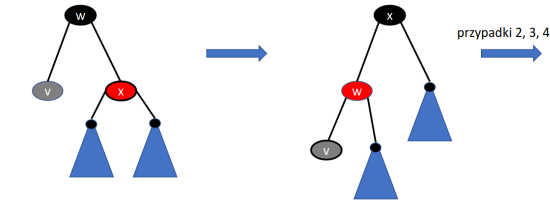
- usuwanie sprowadza się do usunięcia węzła z co najwyżej jednym dzieckiem
- węzeł czerwony zawsze można usunąć
- jeżeli usuwany węzeł jest korzeniem (w = NULL), kolorujemy v na czarno
- problem stanowi usuwany węzeł czarny, gdy w jest różny od NULL



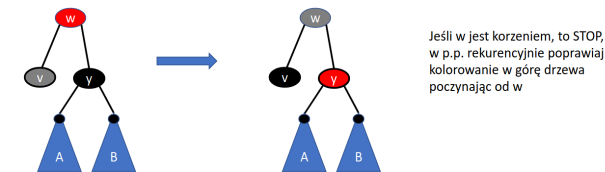
v jest czerwony – kolorujemy v na czarno, STOP  
v jest czarny – patrzymy na konfigurację w poddrzewie o korzeniu x; uwaga – v może być równy NULL (węzeł zewnętrzny)

kolor szary oznacza, że rzeczywisty kolor jest czarny, ale na ścieżce do węzła zewnętrznego brakuje 1 węzła czarnego

Przypadek 1 – x jest czerwony

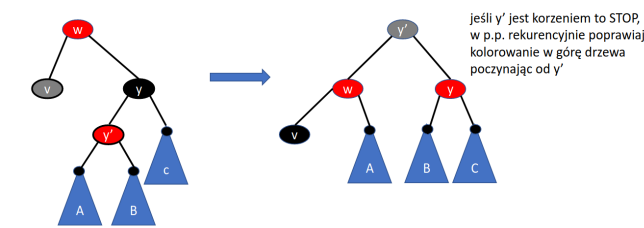


Przypadek 2 – oboje dzieci czarnego sąsiada y są czarne

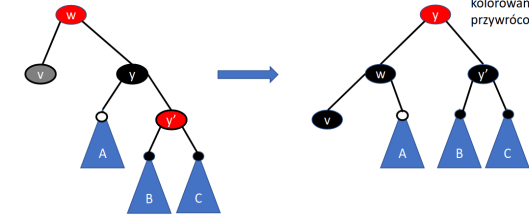


Jeśli v jest korzeniem, to STOP, w p.p. rekurencyjnie poprawiamy kolorowanie w górę drzewa poczynając od w

Przypadek 3 – lewe dziecko czarnego sąsiada y jest czerwone, a prawe czarne



Przypadek 4 – prawe dziecko czarnego sąsiada y jest czerwone, a kolor lewego może być dowolny



Symetrycznie postępujemy, gdy węzeł v jest w prawym poddrzewie w.

koszt Delete z drzewa czerwono-czarnego –  $O(\log n)$

## 2 Algorytmy grafowe

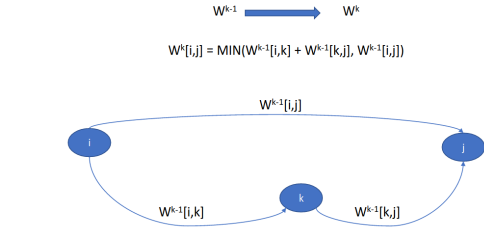
### 2.1 Algorytm Floyda-Warshalla

- Problem najbliższych ścieżek pomiędzy wszystkimi parami wierzchołków
- Idea:

Idea:

$W^k[i,j]$  = waga najbliższej ścieżki z i do j, na której każdy wewnętrzny wierzchołek (poza i oraz j) jest nie większy od k

Zauważmy, że  $W^0[i,j] = A[i,j]$ .



### 2.2 Spójne składowe

- Czas  $O(n+m)$  gdzie n to liczba wierzchołków a m liczba krawędzi.

- Dane

$G=(V,E)$  – graf n-wierzchołkowy

Wynik

funkcja  $C: V \rightarrow V$  taka, że  $C[u] = C[v]$  wtedy i tylko wtedy, gdy u i v są w tej samej spójnej składowej

### 2.3 Najkrótsze ścieżki

- Czas  $O(n+m)$  gdzie n to liczba wierzchołków a m liczba krawędzi.

- Dane

$G=(V,E)$  - graf spójny,  $n=|V|$

s – wyróżniony wierzchołek w G

Wynik

$D[1..n]$  – tablica, w której  $D[u]$  to długość najkrótszej ścieżki z wierzchołka s do wierzchołka u mierzona liczbą krawędzi

Metoda

przeszukiwanie grafu, w którym zbiór S implementujemy jako kolejkę FIFO (First In First Out) jest to tzw. przeszukiwanie wszerz (ang. BFS – Breadth First Search)

### 2.4 Przeszukiwanie wglęb

- Własności

- krawędź niedrzewowa łączy zawsze potomka z przodkiem w w drzewie przeszukiwania w głąb

- numer dfs wierzchołka jest zawsze większy od numeru dfs jego właściwego przodka

### 2.5 Dwuspójne

- Wierzchołek v w grafie G nazywamy rozdzielającym (punktem artykulacji), jeśli jego usunięcie z G (wraz z incydentnymi z nim krawędziami) zwiększa liczbę spójnych składowych w G.

- mostem w grafie G nazywamy krawędź, której usunięcie zwiększa liczbę spójnych składowych grafu.

- spójny graf G jest grafem dwuspójnym wierzchołkowo (krawędziowo), jeśli nie zawiera wierzchołków rozdzielających (mostów)

- Dwuspójną składową grafu G nazywamy każdy jego maksymalny dwuspójny podgraf (z maksymalną możliwą liczbą wierzchołków i krawędzi).

- Istnieje algorytm znajdowania liczby dwuspójnych w grafie w czasie  $O(n+m)$  za pomocą funkcji low

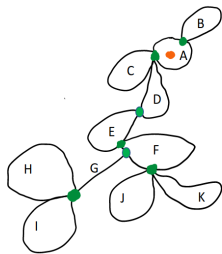
- Struktura dwuspójnych składowych

### Struktura dwuspójnych składowych

- przypomina drzewo
- **czerwona kropka, to korzeń drzewa przeszukiwania w głąb**
- **zielone kropki to wierzchołki rozdziałające**
- składowe B, C, H, I, J, K to „liście”

### Ważne spostrzeżenie:

- podczas przeszukiwania w głąb, jeśli wejdziemy do liścia-dwuspójnej składowej, to przed jego opuszczeniem „przejrzane” zostaną wszystkie krawędzie tej dwuspójnej składowej
- pierwszą krawędzią przeglądana jest zawsze krawędź drzewowa, którą wchodzimy z wierzchołka rozdziałającego w głąb tej składowej
- jeśli odkładamy krawędzie grafu na stos w kolejności ich przeglądania (krawędź nie drzewowa jest najpierw odkrywana w potomku jej drugiego końca), to po odkryciu wierzchołka rozdziałającego dla liścia w chwili wychodzenia z odpowiadającej jej dwuspójnej składowej, wszystkie krawędzie tej dwuspójnej składowej znajdują się na stosie, a najgłębiej jest położona krawędź drzewowa prowadząca w głąb składowej z wierzchołka rozdziałającego



- Istnieje algorytm znajdowania dwuspójnych składowych w czasie  $O(m)$

## 3 Kolorowanie

- Kolorowaniem (wierzchołkowym) grafu nieskierowanego nazywamy takie przypisanie kolorów wierzchołkom grafu, że żadne dwa sąsiednie wierzchołki nie mają przypisanych tych samych kolorów.
- Jeżeli istnieje kolorowanie grafu  $G$  z użyciem  $k$  kolorów, to mówimy że  $G$  jest  $k$ -kolorowalny.
- Minimalną liczbę kolorów, którymi można pokolorować graf  $G$  nazywamy jego liczbą chromaticzną i oznaczamy przez  $\chi(G)$
- Przez  $\Delta(G)$  oznaczamy maksymalny stopień wierzchołka w grafie  $G$  (stopień wierzchołka to liczba jego sąsiadów w tym grafie)
- Oczywiście:  $\chi(G) \leq \Delta(G) + 1$ .

### 3.1 Brooks

- Każdy graf  $G$  różny od cyklu nieparzystej długości i grafu pełnego można pokolorować  $\Delta(G)$  kolorami.
- Graf jest  $k$ -kolorowalny wtedy i tylko wtedy, gdy każda jego dwuspójna składowa jest  $k$ -kolorowalna.
- Algorytm kolorowania Brooksa+ działa w czasie  $O(n+m)$
- Slajd:

#### Kolorowanie Brooks'a+

##### Dane

$G=(V,E)$  – dwuspójny graf  $G$  o co najmniej 4 wierzchołkach

$a, b, c \in V$  – trzy wierzchołki takie, że  $a, b$  nie są połączone krawędzią, natomiast  $c$  jest sąsiadem zarówno  $a$ , jak i  $b$  (odległość pomiędzy  $a$  i  $b$  w grafie  $G$  wynosi 2) oraz  $G \setminus \{a,b\}$  jest spójny

##### Wynik

$Color[1..n]$  – tablica kolorów wierzchołków taka, że  $1 \leq Color[i] \leq \Delta(G)$  oraz  $Color[i] \neq Color[j]$ , jeśli tylko  $i-j \in E$

##### Idea

- kolorujemy wierzchołki  $a$  i  $b$  kolorem 1
- przeglądamy w głąb graf  $G \setminus \{a,b\}$  poczynając od  $c$  i numerujemy wierzchołki w kolejności odwiedzania
- kolorujemy wierzchołki w kolejności od największego do najmniejszego numeru, kolorując każdy wierzchołek najmniejszym kolorem, który nie został jeszcze użyty do pokolorowania jego sąsiadów w całym grafie  $G$