Politechnika Wrocławska, Wydział Informatyki i Zarządzania

Zbigniew Szpunar

Wykład 9.

Drzewa

Drzewa – podstawowe pojęcia

Drzewo (*ang. tree*) ukorzenione jest zbiorem węzłów (wierzchołków) takim, że:
- każdy węzeł posiada k węzłów-dzieci (k>=0) i co najwyżej jednego przodka-rodzica,
- dokładnie jeden węzeł nie posiada przodka-rodzica; jest on korzeniem drzewa (*ang. root*),
- dla każdego węzła istnieje jedna droga od korzenia do tego wierzchołka (co oznacza, że w drzewie, traktowanym jak graf, nie występują cykle).

Węzły (wierzchołki) nie posiadające węzłów-dzieci są nazywane liśćmi drzewa.

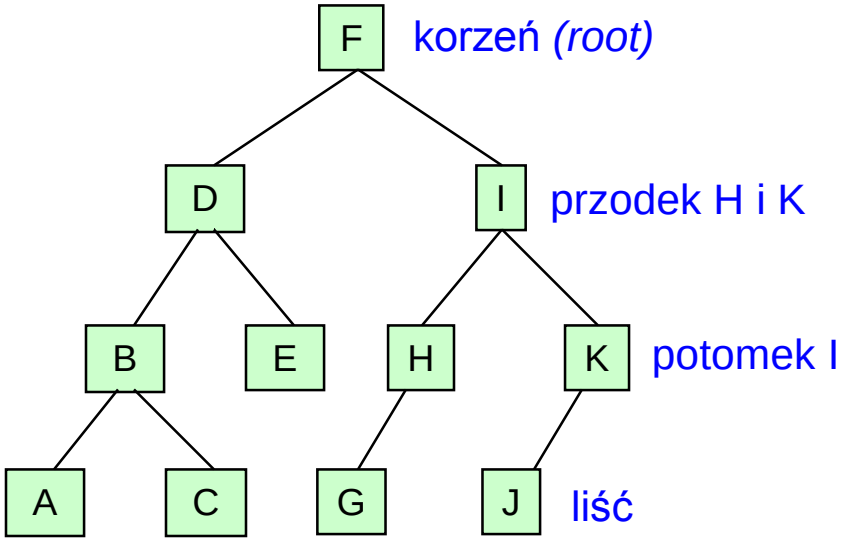
Uwaga: W niektórych definicjach drzew liście traktuje się jako zewnętrzne węzły w drzewie, które nie przechowują wartości kluczy (węzły „puste” *nil*).

Jeśli w drzewie, traktowanym jak graf skierowany, istnieje krawędź (gałąź) od węzła X do węzła Y, to X nazywamy przodkiem (rodzicem, ojcem) Y, a Y nazywamy potomkiem (dzieckiem, synem) X.

Liście drzewa są więc węzłami (wierzchołkami) bez potomków.

Drzewa – podstawowe pojęcia

Przykład drzewa



Węzeł X wraz z jego potomkami nazywamy poddrzewem o korzeniu X.

Głębokość (poziom) węzła X jest to długość drogi (wyrażona liczbą krawędzi) od korzenia do węzła X (korzeń ma głębokość 0).

Wysokość drzewa jest to maksymalna długość drogi od korzenia do liści (czyli największa z głębokości liści).

Wysokość drzewa pustego wynosi -1, a drzewa jednowęzłowego 0.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa uporządkowane, binarne drzewo poszukiwań

Drzewo uporządkowane - drzewo w którym węzły potomne każdego wierzchołka są uporządkowane (na rysunku - od lewej do prawej).

W programowaniu największe znaczenie mają uporządkowane drzewa binarne.

Binarnym drzewem poszukiwań (*ang. Binary Search Tree, BST*) będziemy nazywali drzewo uporządkowane spełniające warunki nieco bardziej zaostrzone w stosunku do literatury (np. *Cormen*):

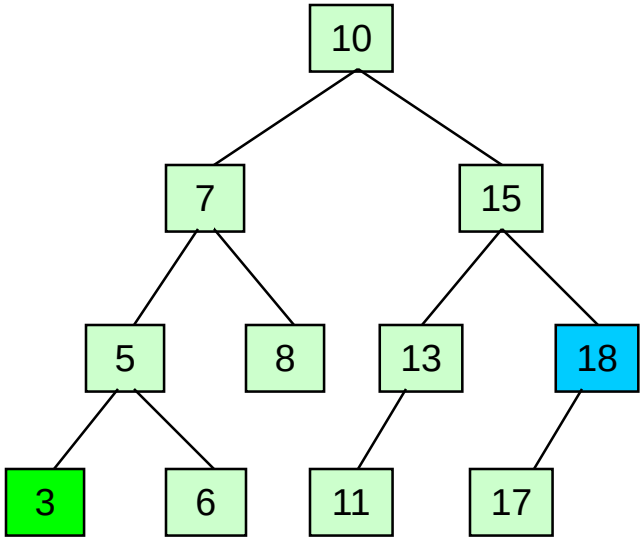
1. Potomek wierzchołka może być lewy lub prawy.
2. Każdy wierzchołek ma co najwyżej dwa węzły potomne.
3. Wartości kluczy nie powtarzają się w drzewie (w praktyce można rozważać uogólniony wariant BST z powtarzającymi się wartościami kluczy).
4. W lewym poddrzewie znajdują się wartości kluczy mniejsze od klucza korzenia, a w prawym - większe (dla przypadku uogólnionego: w lewym poddrzewie znajdują się wartości kluczy mniejsze od klucza korzenia lub równe, a w prawym - większe lub równe).

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa uporządkowane, binarne drzewo poszukiwań

Minimum – węzeł o najmniejszej wartości klucza jest osiągalny na końcu ścieżki od korzenia poprzez lewych potomków.

Maksimum – węzeł o największej wartości klucza jest osiągalny na końcu ścieżki od korzenia poprzez prawych potomków.

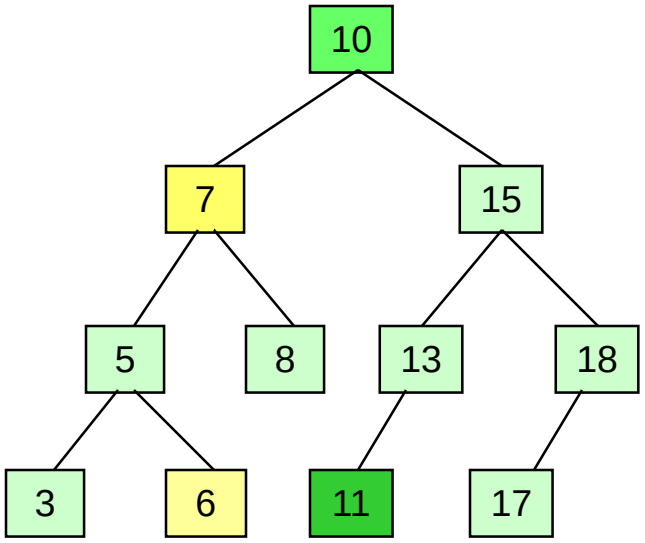


Drzewa uporządkowane, binarne drzewo poszukiwań

Następnik (*ang. successor*) danego węzła w drzewie poszukiwań to węzeł o wartości klucza bezpośrednio większej niż wartość klucza danego węzła.

Dwa przypadki:

1. Dla węzła posiadającego prawego potomka następnikiem jest minimum prawego poddrzewa (dla którego korzeniem jest ten prawy potomek).
2. Dla węzła nie posiadającego prawego potomka następnik jest wyszukiwany na ścieżce "w górę", aż do napotkania węzła X, który jest czyimś lewym potomkiem. Wówczas przodek tego węzła X jest szukanym następnikiem.



Przykład dla przypadku 1.:
dla węzła "10" następnikiem jest węzeł "11"

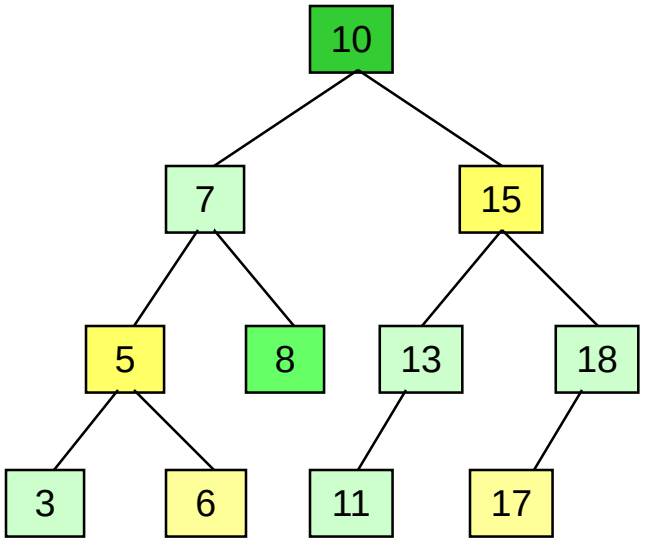
Przykład dla przypadku 2.:
dla węzła „6" następnikiem jest węzeł „7"

Drzewa uporządkowane, binarne drzewo poszukiwań

Poprzednik (*ang. predecessor*) danego węzła w drzewie poszukiwań to węzeł o wartości klucza bezpośrednio mniejszej niż wartość klucza danego węzła.

Dwa przypadki:

1. Dla węzła posiadającego lewego potomka poprzednikiem jest maksimum lewego poddrzewa (dla którego korzeniem jest ten lewy potomek).
2. Dla węzła nie posiadającego lewego potomka poprzednik jest wyszukiwany na ścieżce "w górę", aż do napotkania węzła X, który jest czyimś prawym potomkiem. Wówczas przodek tego węzła X jest szukanym poprzednikiem.



Przykład dla przypadku 1.:
dla węzła "10" poprzednikiem jest węzeł "8"

Przykłady dla przypadku 2.:
dla węzła „6" poprzednikiem jest węzeł „5"
dla węzła "17" poprzednikiem jest węzeł "15"

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Najprostsza postać węzła drzewa:

```
class Node {
   Object value;   // tu jest przechowana wartość klucza dla węzła
   Node left;      // lewe poddrzewo
   Node right;     // prawe poddrzewo
   Node(Object x) { // konstruktor węzła (wierzchołka) - liścia
      value=x; left = right = null; }
   Node(Object x, Node lewe, Node prawe) { // konstruktor węzła (wierzchołka)
                                           // spinającego (łączącego) poddrzewa
      value=x; left =lewe; right=prawe; // bez sprawdzania różnych warunków
                                        // dla uporządkowanego drzewa binarnego
   }
   // .. stosowne metody dla węzła
}
```

Podstawowe działania na drzewie:

1. Przechodzenie (w określonym porządku/kolejności) przez wszystkie wierzchołki drzewa ("przetwarzanie całego drzewa").
2. Wyszukiwanie elementu w drzewie (sprawdzanie czy w drzewie występuje wierzchołek o zadanej wartości klucza).
3. Wstawianie elementu (dołączanie nowego wierzchołka do drzewa).
4. Usuwanie wierzchołka z drzewa.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Przechodzenie przez wszystkie wierzchołki drzewa (o korzeniu x)

Istnieje kilka sposobów przechodzenia (trawersacji) drzewa:

1. Przejście "in-order"

Porządkujące, zwane czasem "porządkiem poprzecznym".
Daje wierzchołki od najmniejszego do największego; najważniejszy sposób przejścia przez drzewo binarne.

przejdź_InOrder(Wierzchołek x) <= korzeń drzewa (poddrzewa)

1. przejdź_InOrder(lewy potomek x);
2. przetwórz(x); // odwiedź (wykorzystaj) węzeł x
3. przejdź_InOrder(prawy potomek x)

Pozostałe sposoby przechodzenia drzewa (2..6):
każda z innych możliwych (sześciu) kombinacji operacji przejdź(...) i przetworz(...)

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Przechodzenie przez wszystkie wierzchołki drzewa (o korzeniu x)

// przykład metody klasy Node do wyprowadzenia ciągu wartości z drzewa
// w porządku poprzecznym (in order):

```
public String toStringInOrder(){
        String b="";
        if (left!=null) b+=left.toStringInOrder();
        b+=" "+value.toString();
        if (right!=null) b += " " + right.toStringInOrder();
        return b;
}
```

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Przejście "in-order"

// przykład użycia metody toStringInOrder() dla drzewa

```
import sorting.Comparator;
public class BST {
    private final Comparator _comparator;
    private Node _root;
    public BST(Comparator comparator) {
       _comparator = comparator;
       _root=null;
    }
    public String treeToStringInOrder(){
        String b="In order: ";
        if (_root!=null) b+=_root.toStringInOrder();
            else b+="drzewo puste";
        return b;
   }
   // .. miejsce na pozostałe metody dla drzewa BST ..
}
```

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Przechodzenie przez wszystkie wierzchołki drzewa (o korzeniu x)

Inne sposoby przechodzenia (trawersacji) drzewa:

2. Przejście "pre-order"

przejdź_PreOrder(Wierzchołek x) <= korzeń drzewa (poddrzewa)

1. przetwórz(x); // odwiedź (wykorzystaj) węzeł x
2. przejdź_PreOrder(lewy potomek x);
3. przejdź_PreOrder(prawy potomek x)

3. Przejście "post-order"

przejdź_PostOrder(Wierzchołek x) <= korzeń drzewa (poddrzewa)

1. przejdź_PostOrder(lewy potomek x);
2. przejdź_PostOrder(prawy potomek x);
3. przetwórz(x); // odwiedź (wykorzystaj) węzeł x

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Wyszukiwanie elementu w drzewie

Oznacza sprawdzanie czy w drzewie występuje element o zadanej wartości V klucza oraz (jeśli jest) pobranie wartości przechowywanej w tym węźle.

wyszukaj(Wartość V):

- Ustal: węzeł = korzeń drzewa.
- Jeśli węzeł == null -> koniec, element o wartości klucza V nie istnieje w drzewie.
- Porównanie wartości klucza dla węzła z wartością V (z użyciem komparatora).
  - jeśli równe -> element znaleziono; zwróć wartość elementu, koniec.
  - jeśli szukana wartość jest mniejsza niż V, ustal węzeł = lewy potomek, przejdź do 2.
  - jeśli szukana wartość jest większa niż V, ustal węzeł = prawy potomek, przejdź do 2.

Przy każdym przejściu do potomka (w głąb drzewa) zostaje wyeliminowana z przeglądania połowa pozostałych jeszcze węzłów drzewa. Jest to więc wyszukiwanie binarne – algorytm o logarytmicznej złożoności obliczeniowej.
O maksymalnej liczbie "kroków" decyduje głębokość drzewa. Jeśli drzewo jest wyważone ("regularne", "symetryczne"), jego głębokość jest określona zależnością logarytmiczną. Może się jednak zdarzyć, że drzewo będzie miało głębokość równą n-1 (gdzie: n – liczba węzłów).

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Wyszukiwanie elementu w drzewie

Przykład iteracyjnej realizacji metody wyszukiwania elementu w drzewie:

```
public Object find(Object x) {
   Node t = search(x);
   return t !=null ? t.value : null;
}
private Node search(Object value) { // metoda iteracyjna
   Node node = _root;
   int cmp=0;
   while (node != null &&(cmp = _comparator.compare(value, node.value))!=0)
      node = cmp < 0 ? node.left : node.right;
   return node;
}
```

Realizacja rekurencyjna nie powinna nastręczać trudności. Proszę się o tym przekonać...

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Wstawianie elementu (dołączanie nowego wierzchołka do drzewa)

Oznacza przeszukanie drzewa i – jeśli nie ma w nim węzła o wartości równej wartości klucza wstawianego elementu – dołączenie go jako liść).
Jeśli taki węzeł jest to zgłaszany jest wyjątek.

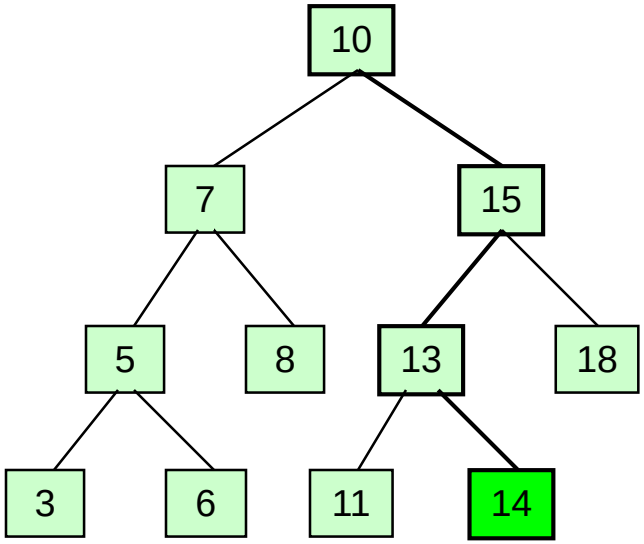
Ustal: węzeł = korzeń drzewa.
wstaw do drzewa (Wartość V):
1. Jeśli węzeł == null, to utwórz węzeł z wartością V. (koniec pomyślny).
2. Porównanie wartości klucza dla węzła z wartością V (z użyciem komparatora).
   - jeśli wartość V jest mniejsza od wartości węzła, wstaw wartość do poddrzewa wskazanego przez lewego potomka (węzeł=lewy potomek, przejdź do 2.)
   - jeśli wartość V jest większa od wartości węzła, wstaw wartość do poddrzewa wskazanego przez prawego potomka (węzeł=prawy potomek, przejdź do 2.)
   - jeśli równe -> element już istnieje; zgłoś wyjątek. (koniec niepomyślny).

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Wstawianie elementu (dołączanie nowego wierzchołka do drzewa)

Przykład wstawiania elementu o wartości V=14
1. węzeł=10
2. V>10, wstaw do prawego (15)
1. węzeł=15
2. V<15, wstaw do lewego (13)
1. węzeł=13
2. V>13, wstaw do prawego (13)
1. węzeł==null. Utworz węzeł 14 (jako liść - prawy potomek węzła 13)



Uporządkowane drzewa binarne – podstawowe działania

Wstawianie elementu (dołączanie nowego wierzchołka do drzewa)

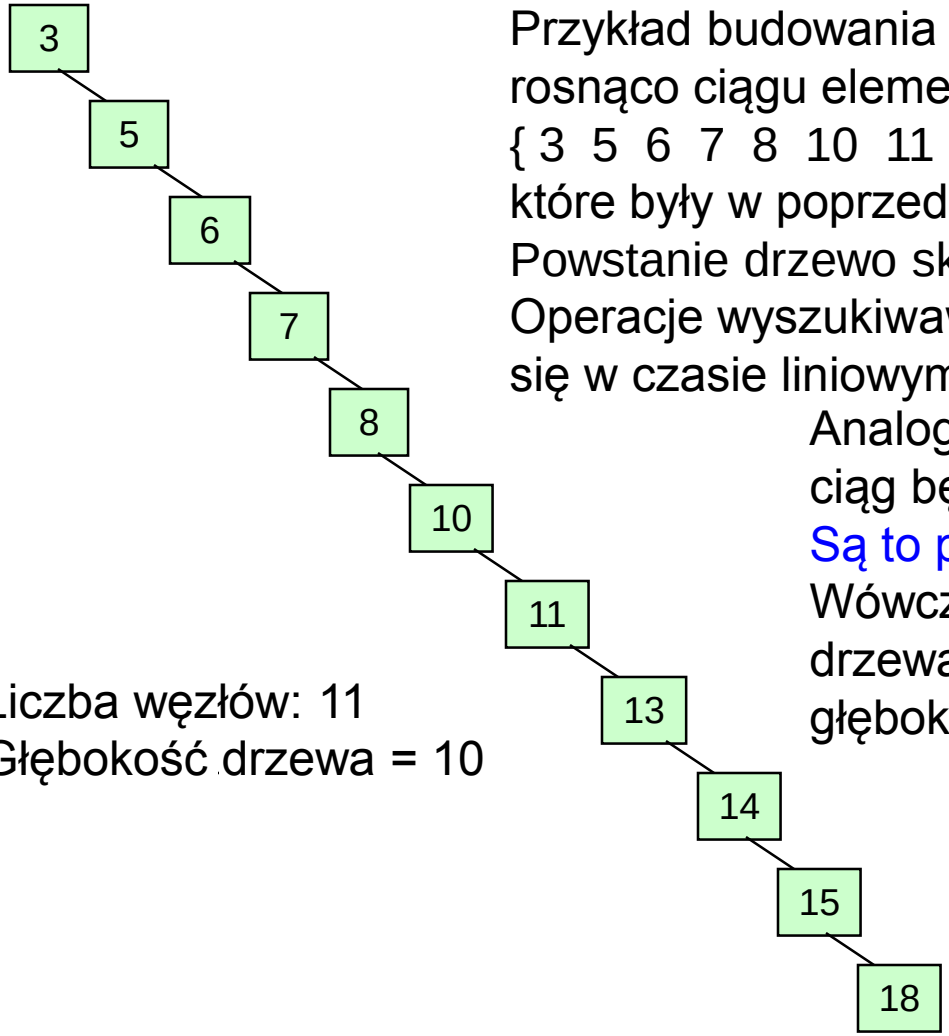
Przykład realizacji metody wstawiania elementu do drzewa:

```
public void insert(Object x) {
    _root= insert(x,_root);
}
protected Node insert(Object x, Node t) {
   if (t== null) t=new Node(x);
     else {
              int cmp=_comparator.compare(x,t.value);
              if(cmp<0) t.left=insert(x, t.left);
                else if(cmp>0) t._right=insert(x, t.right);
                        else throw new DuplicateItemException(x.toString());
            }
     return t;
}
public class DuplicateItemException extends RuntimeException {
     public DuplicateItemException(String message) { super(message); }
}
```

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Wstawianie elementu (dołączanie nowego wierzchołka do drzewa)



Przykład budowania drzewa dla uporządkowanego rosnąco ciągu elementów, np.:
{ 3 5 6 7 8 10 11 13 14 15 18 } – czyli tych, które były w poprzednim drzewie.
Powstanie drzewo skrajnie niewyważone.
Operacje wyszukiwawcze w tym drzewie wykonują się w czasie liniowym, zamiast w logarytmicznym!

Analogiczna sytuacja wystąpi, gdy ciąg będzie uporządkowany malejąco.
Są to przykłady degradacji BST.
Wówczas trzeba prowadzić wyważanie drzewa, by zapewnić możliwie najmniejszą głębokość drzewa.

Liczba węzłów: 11
Głębokość drzewa = 10

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Należy odszukać węzeł, który należy usunąć. Jeśli węzeł nie istnieje, należy zasygnalizować błąd (zgłosić wyjątek).

Węzeł X, który należy usunąć, może znajdować się w jednym z trzech stanów:

1. jest liściem (wówczas można go po prostu usunąć, bez dodatkowych działań),
2. posiada dokładnie jednego potomka (lewego lub prawego); wówczas potomek zajmuje miejsce tego usuwanego węzła (czyli miejsce węzła zajmuje lewe lub prawe poddrzewo usuwanego węzła),
3. posiada dwóch potomków; należy wówczas:
   - węzeł X zamienić miejscami z jego następnikiem N (co oznacza zamianę wartości elementów przechowywanych w węźle; po tej zamianie w węźle X będzie przechowywana wartość następnika N); w tej chwili może być naruszony warunek drzewa binarnego.
   - usunąć węzeł N, który po tej zamianie znalazł się o w jednym ze stanów: 1 (jest liściem) lub 2 (posiada tylko prawego potomka; nie mógł mieć lewego potomka, bo wówczas nie byłby następnikiem - elementem minimalnym w prawym poddrzewie względem usuwanego elementu X).

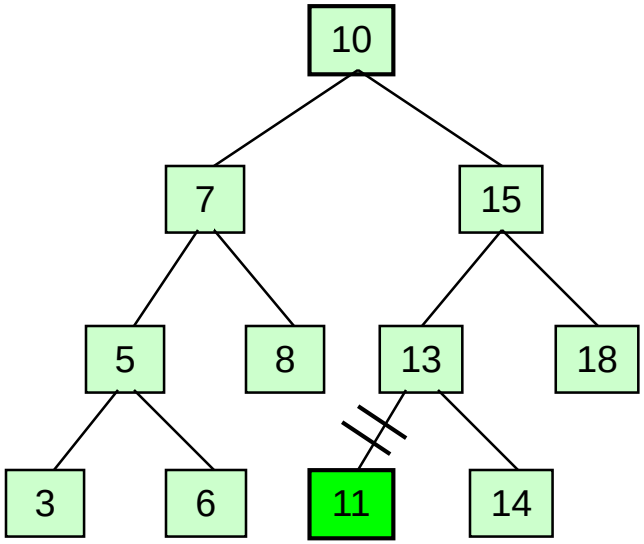
   Węzeł X można zastąpić jego poprzednikiem (elementem maksymalnym, czyli skrajnie prawym w lewym poddrzewie węzła X.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

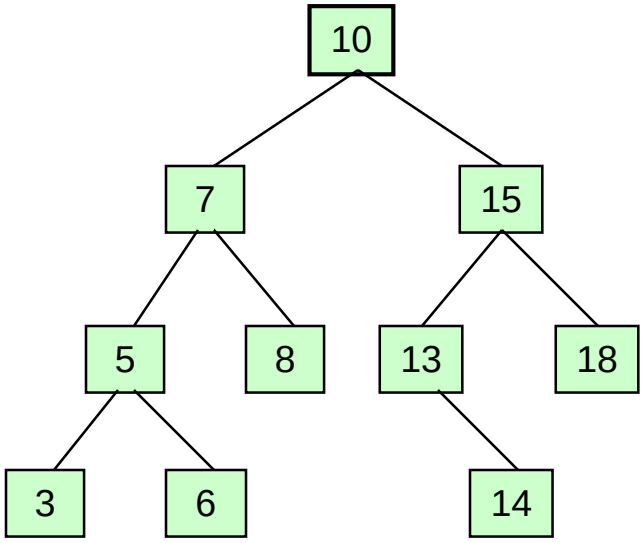
Przykład usuwania elementu o wartości V=11 jest bezproblemowe:
przodek (w tym przypadku "13") traci potomka (w tym przypadku lewego).



Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Stan po usunięciu węzła 11



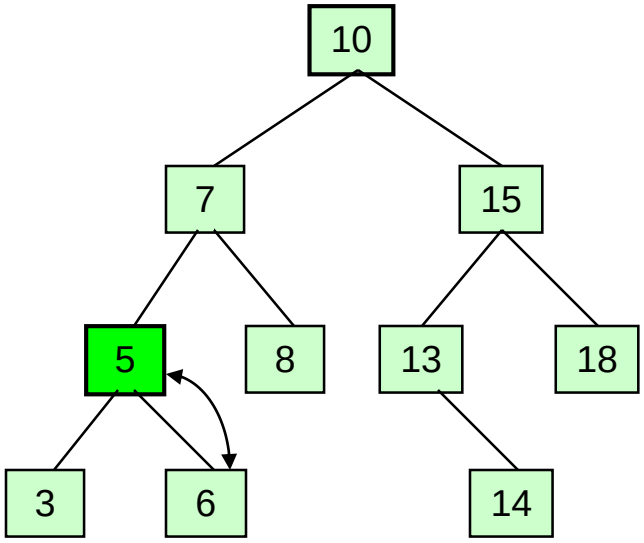
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Przykład usuwania elementu o wartości V=5:

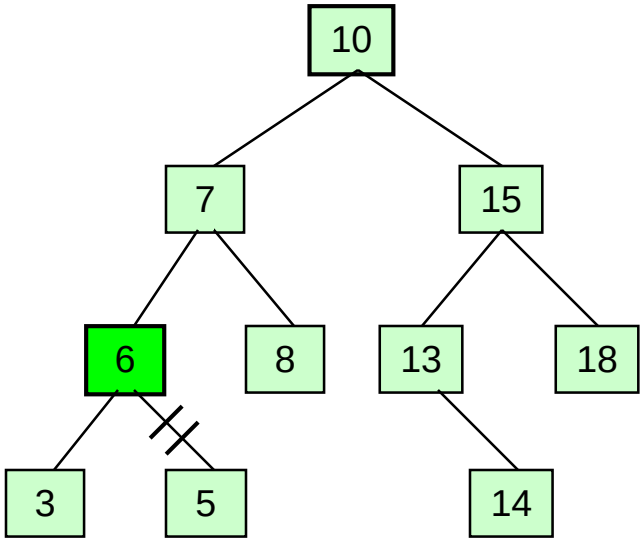
Następnikiem 5 jest 6 (skrajny lewy element jego prawego poddrzewa), więc następuje zamiana wartości 5 i 6 oraz bezproblemowe usunięcie węzła "6", w którym (w wyniku przestawienia) znalazła się usuwana wartość 5.



Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Stan po przestawieniu elementów 5 i 6...

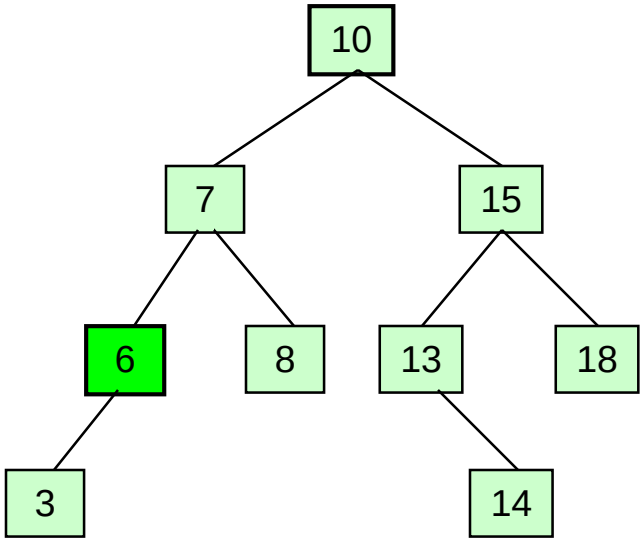


Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Stan po usunięciu elementu 5 (z poprzedniej pozycji elementu 6).



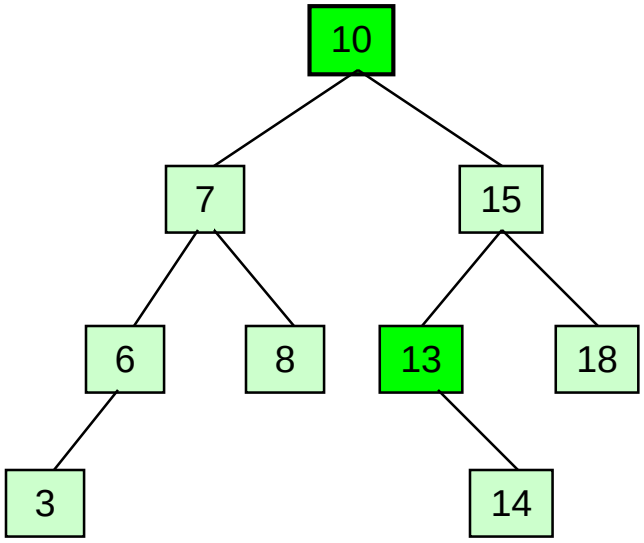
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Jeszcze jedno usunięcie – tym razem korzenia 10

Jego następnikiem jest 13 (lewy skrajny element prawego poddrzewa).



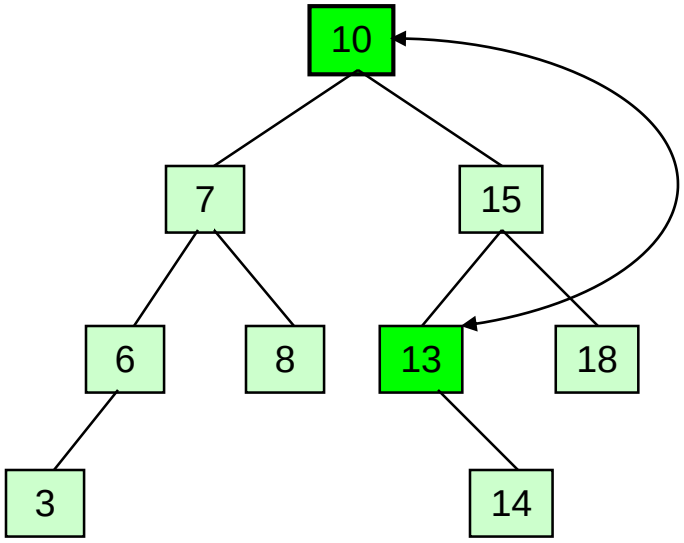
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Jeszcze jedno usunięcie – tym razem korzenia 10

Jego następnikiem jest 13 (lewy skrajny element prawego poddrzewa).
Elementy te należy zamienić miejscami.



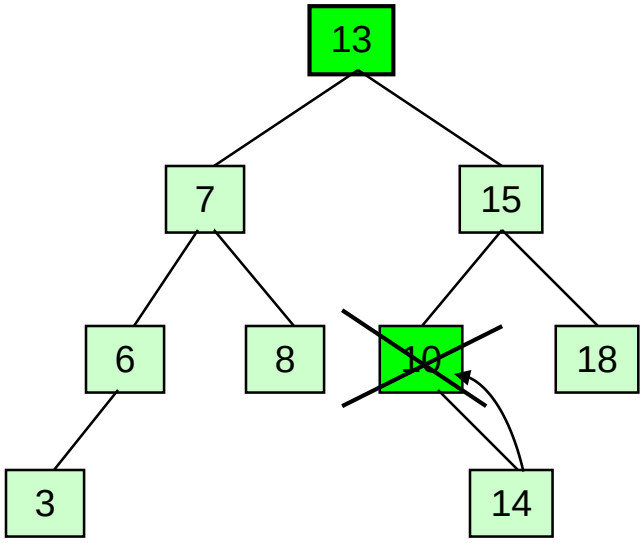
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Jeszcze jedno usunięcie – tym razem korzenia 10

Jego następnikiem jest 13 (lewy skrajny element prawego poddrzewa).
Elementy te należy zamienić miejscami.
Element 10 należy usunąć, zastępując go jego następnikiem (jest nim 14).



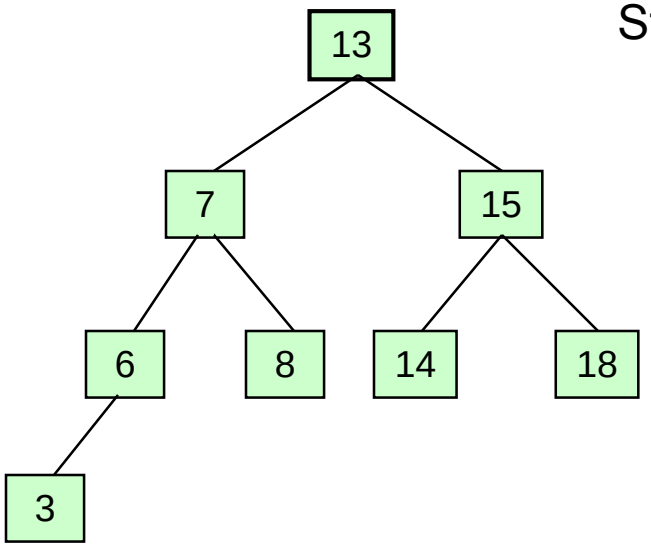
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Jeszcze jedno usunięcie – tym razem korzenia 10

Jego następnikiem jest 13 (lewy skrajny element prawego poddrzewa).
Elementy te należy zamienić miejscami.
Element 10 należy usunąć, zastępując go jego następnikiem (jest nim 14).
Stan po usunięciu elementu 10 z korzenia:



Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Uporządkowane drzewa binarne – podstawowe działania

Usuwanie węzła z drzewa

Przykład realizacji metody usuwania elementu z drzewa:

```
public void delete(Object x) { _root=delete(x,_root); }

protected Node delete(Object x, Node t) {
    if (t==null) throw new ItemNotFoundException(x.toString());
       else {
               int cmp=_comparator.compare(x,t.value);
               if (cmp<0) t._left=delete(x, t.left);
                else if(cmp>0) t.right=delete(x,t.right);
                       else if(t.left!=null &&t.right!=null) t.right=detachMin(t.right,t);
                              else  t = (t.left != null) ? t.left : t.right;
          }
      return t;
    }
//zastąpienie usuwanego elementu jego następnikiem i usunięcie następnika
protected Node detachMin(Node t, Node del) {
  if (t.left!=null) t.left=detachMin(t.left, del);
    else {del.value=t.value; t=t.right;}
 return t;
}
```

Wyważanie drzewa

Wstawianie i usuwanie węzłów drzewa binarnego może naruszyć jego wyważenie i znacznie zwiększyć wysokość drzewa przy stosunkowo niewielkiej liczbie węzłów (aż do postaci listy uporządkowanej -> zdegenerowane drzewo binarne), co sprowadza złożoność obliczeniową operacji z logarytmicznej do liniowej.

Dlatego należy dbać na bieżąco o „kształt” drzewa i dokonywać wyważenia (balansowania, *ang. balancing*) praktycznie po każdej operacji naruszającej to wyważenie.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Wyważanie drzewa

Wyważanie (balansowanie) drzewa jest ciągiem działań polegających na przekształceniu drzewa binarnego przeciążonego (w lewo lub w prawo) do postaci równoważnej o mniejszej wysokości.

Operacje wyważania na bieżąco (po każdej zmianie) dużych drzew są kosztowne.

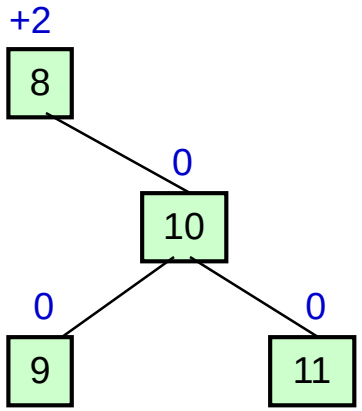
Jedną z najprostszych metod zaproponowali Adelson-Velskij i Landis (stąd pochodzi akronim AVL dla oznaczania drzew wyważonych tą metodą).

Idea metody:

Śledzenie wysokości obydwu poddrzew każdego z węzłów;
jeśli dla każdego z węzłów wysokości poddrzew różnią się nie więcej, niż o 1, to takie drzewo uznawane jest za wyważone (-> drzewo AVL).

Wyważanie drzewa

Prosty przykład drzewa niewyważonego:



Wyważenie przywraca się poprzez stosowanie ciągu elementarnych operacji (przekształcania drzewa binarnego w drzewo równoważne) zwanych rotacjami.

Rotacje wykonuje się rozpoczynając od miejsca wstawienia/usunięcia węzła w kierunku do korzenia drzewa.

Typy rotacji:
- w lewo,
- w prawo,
- w wariancie pojedynczym,
- w wariancie podwójnym.

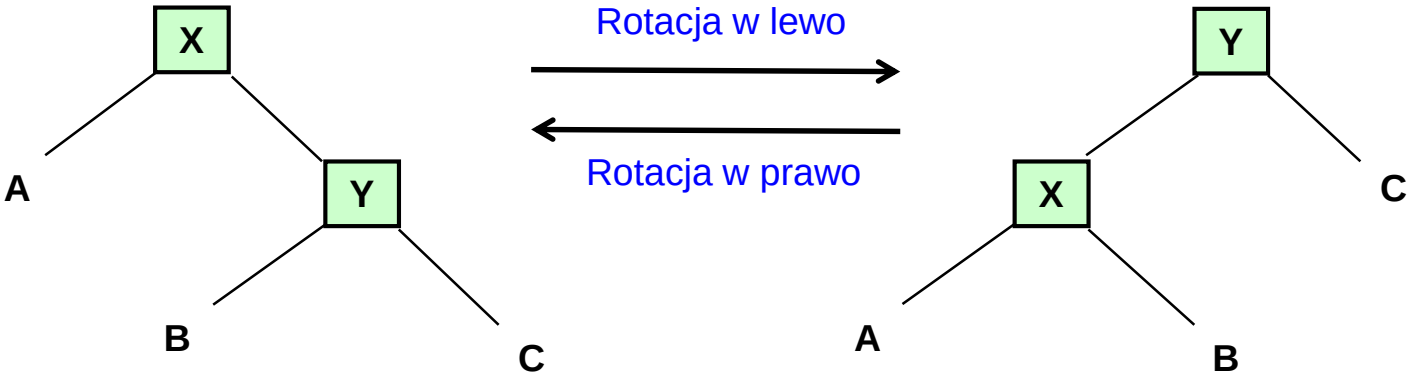
Wyważanie drzewa

Rotacja jest kluczową procedurą wykorzystywaną w procesie wstawiania i usuwania.

Schemat przedstawia działanie 2 odmian: lewej i prawej rotacji.

Rotacja zachowuje porządek *inOrder* kluczy.

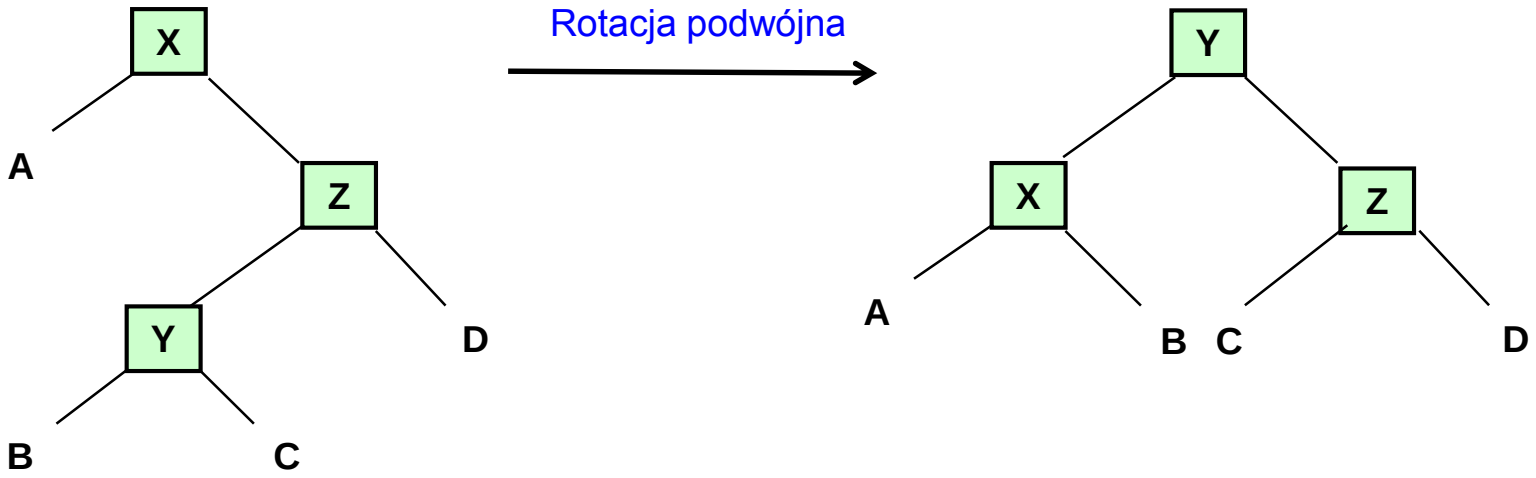
Lewa rotacja może być wykonana tylko, gdy prawy syn *X* jest niepusty. Lewa rotacja polega na „obrocie” wokół krawędzi między węzłami *X* i *Y*. W wyniku rotacji *Y* staje się nowym korzeniem poddrzewa, *X* staje się jego lewym synem, a lewy syn *Y* zostaje prawym synem węzła *X*.



Przedstawiona rotacja jest tzw. rotacją pojedynczą.

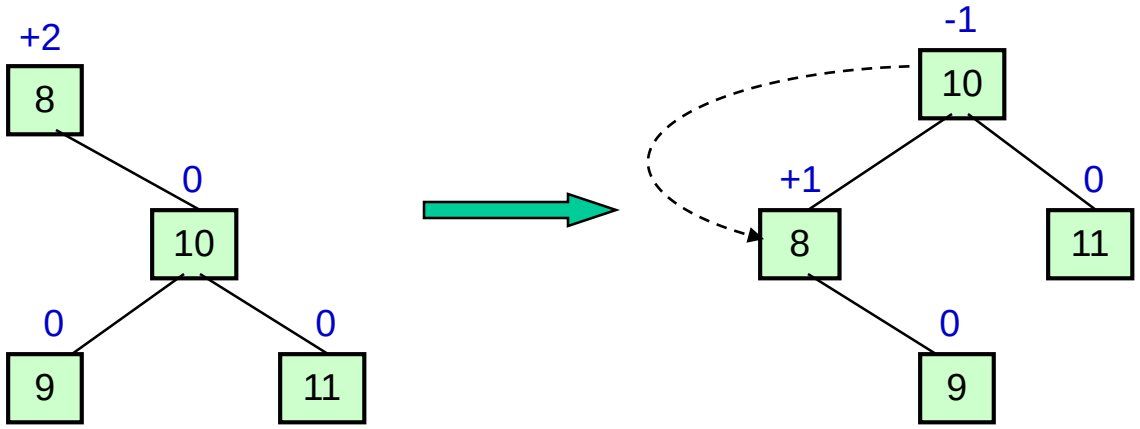
Wyważanie drzewa

Rotacja podwójna obejmuje 3 węzły i jest złożeniem dwóch rotacji pojedynczych.



Wyważanie drzewa

Prosty przykład rotacji wyważającej drzewo:

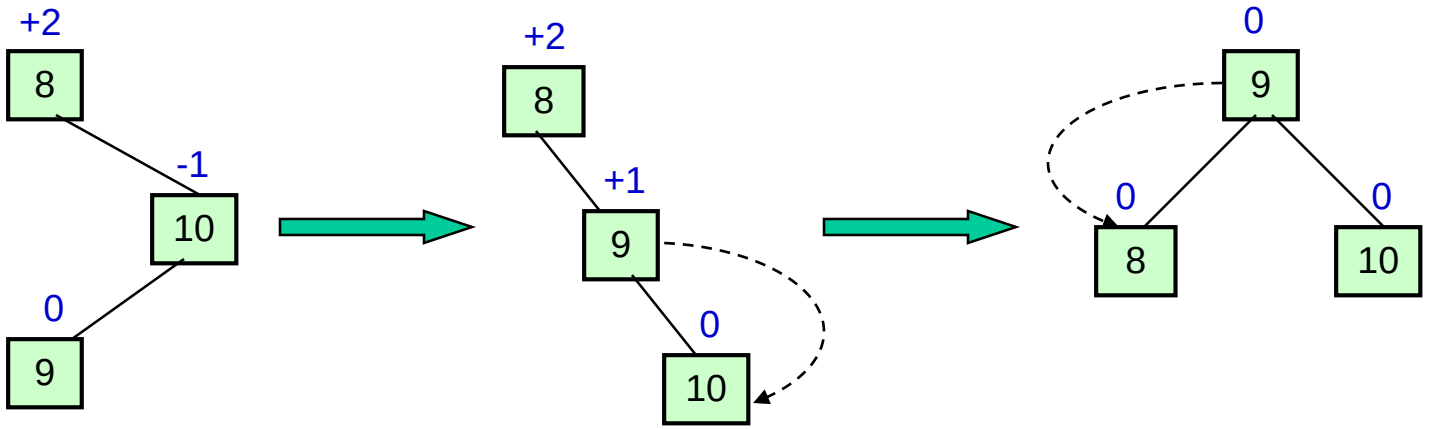


Zastosowana rotacja (pojedyncza, w lewo – ponieważ węzeł 10 był przeciążony w prawo) promuje węzeł 10 i degraduje węzeł 8.

Taka sytuacja "przeciążenia” mogła nastąpić np. po usunięciu (nie pokazanego na rysunku z lewej strony) węzła 7 (lewego potomka węzła 8).

Wyważanie drzewa

Inna sytuacja:



Zastosowane zostały dwie rotacje (pierwsza w prawo dla węzła 10, druga w lewo – dla węzła 8), czyli rotacja podwójna.

Wyważanie drzewa

W zależności od rodzaju przeciążenia wybiera się rodzaj rotacji:

Jeśli występuje przeciążenie w lewo, to:

jeśli potomek zrównoważony lub przeciążony w lewo, to rotacja pojedyncza;
jeśli potomek przeciążony w prawo, to rotacja podwójna.

Jeśli występuje przeciążenie w prawo, to:

jeśli potomek zrównoważony lub przeciążony w prawo, to rotacja pojedyncza;
jeśli potomek przeciążony w lewo, to rotacja podwójna.

Ciekawe wyliczenie dla poparcia tezy o potrzebie wyważania drzewa binarnego:
W drzewie doskonale wyważonym złożonym z 1 000 000 węzłów znalezienie węzła wymaga średnio około log2(1000000) porównań (czyli ok. 20 porównań).
W drzewie AVL o takich rozmiarach liczba porównań jest tylko o 45% większa (ok. 28). W drzewie zdegenerowanym do listy: 500 000.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa binarne – podsumowanie dotychczasowych rozważań

Drzewa BST są wygodne do efektywnego implementowania operacji:

search, successor, predecessor, minimum, maximum, insert, delete

w czasie $O(h)$, (gdzie h jest wysokością drzewa).

Jeśli drzewo jest zbalansowane (ma wtedy wysokość $h = O(\lg n)$), operacje te są najbardziej efektywne.

Operacje insert i delete mogą powodować, że:

- drzewo przestaje być zbalansowane,
- w najgorszym przypadku drzewo staje się listą liniową, wysokość $h = O(n)$.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Staramy się znaleźć takie metody operowania na drzewie, by pozostawało ono zbalansowane.

Jeśli operacja insert lub delete spowoduje utratę zbalansowania drzewa, będziemy przywracać tę własność w czasie co najwyżej $O(\lg n)$, tak, aby nie zwiększać złożoności.

Potrzebujemy zapamiętywać dodatkowe informacje, aby to osiągnąć.

Najbardziej popularne struktury danych dla binarnych drzew zbalansowanych to:

- drzewa AVL: różnica wysokości dla poddrzew wynosi co najwyżej 1,
- drzewa czerwono-czarne: o wysokości co najwyżej $2(\lg n + 1)$.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

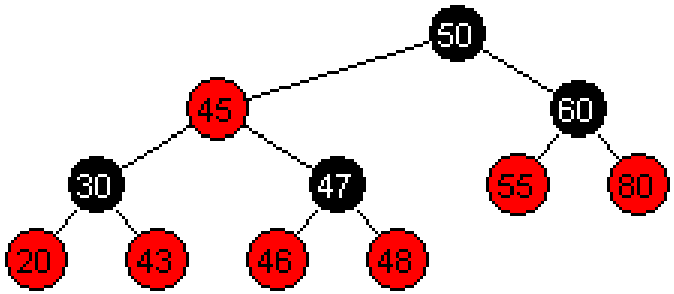
Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)

Drzewem czerwono-czarnym (RBT) nazywany binarne drzewo poszukiwawcze, dla którego każdy węzeł posiada dodatkową informację: bit koloru (czerwony albo czarny, pełniący taką samą rolę, jak wskaźnik wyważenia w drzewie AVL), o następujących własnościach:

1. Każdy węzeł posiada kolor – czerwony lub czarny.
2. Korzeń jest koloru czarnego.
3. Każdy liść (*null*) jest czarny.
4. Węzły potomne czerwonego węzła są czarne.
5. Każda ścieżka od ustalonego węzła do liścia musi zawierać tę samą liczbę czarnych węzłów.

Przykład RBT:



Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)

Definicja:
Czarną wysokością węzła x, oznaczaną jako *bh*(*x*), nazywamy liczbę czarnych węzłów na ścieżce prowadzącej od tego węzła do dowolnego liścia.

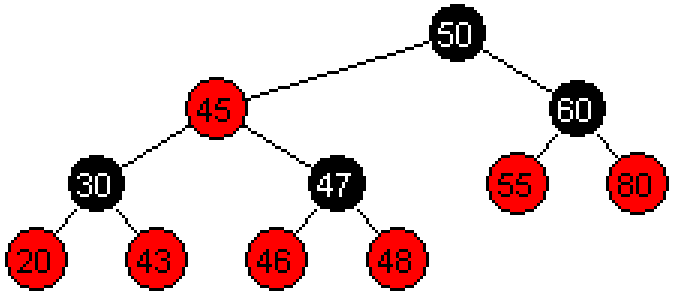
Wybrane cechy użytkowe RBT:

1. Wysokość RBT o n węzłach wynosi co najwyżej 2 lg(n+1).
2. Operacje wyszukiwania w RBT mają więc złożoność O(lg n). Algorytmy dodawania i usuwania węzłów do drzew RBT również są w stanie wykonywać się w czasie O(lg n).
3. Przewaga drzew RBT nad drzewami AVL polega na ograniczeniu liczby rotacji do dwóch, co zwiększa ich przydatność w przypadkach konieczności dokonywania częstych aktualizacji drzewa (czyli wykonywania operacji insert i delete).
4. Drzewa AVL lepiej nadają się do operacji wyszukiwania, niż drzewa RBT (są zazwyczaj niższe; ich wysokość nie przekracza 1.44 lg(n).
5. Operacje insert i delete są dość skomplikowane (ze względu na dużą liczbę różnych sytuacji, spowodowanych tymi operacjami, czyli mnogością wariantów).

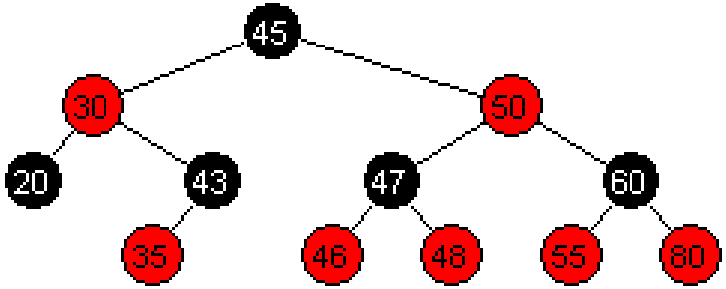
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)



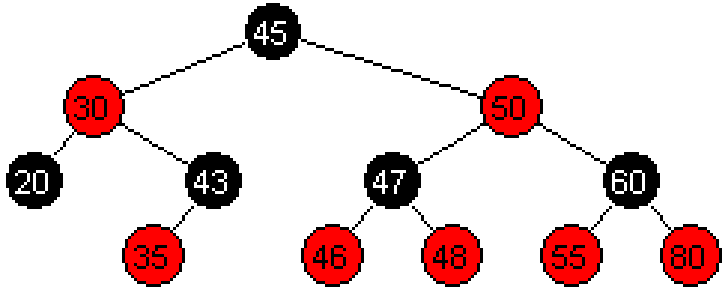
Po dodaniu (*insert*) węzła 35:



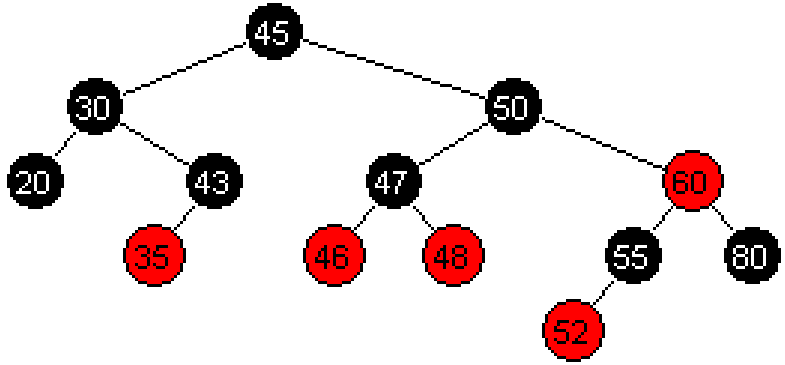
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)



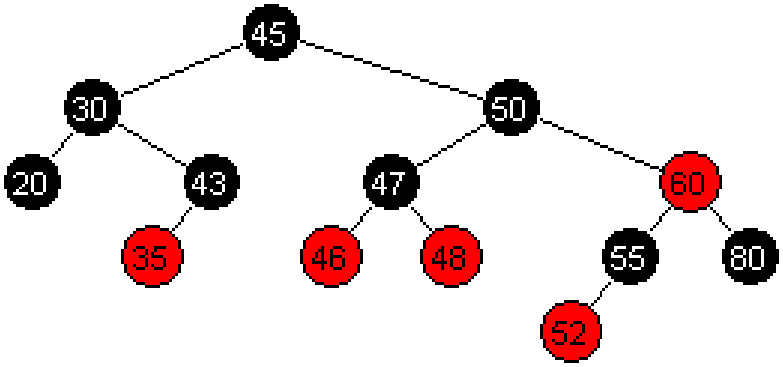
Po dodaniu (*insert*) węzła 52:



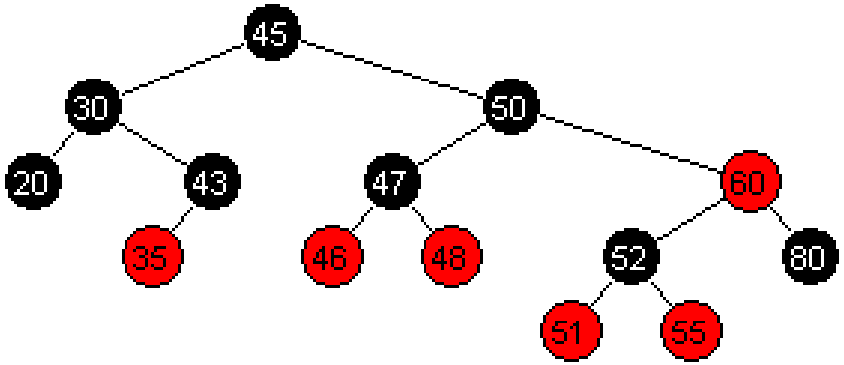
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)



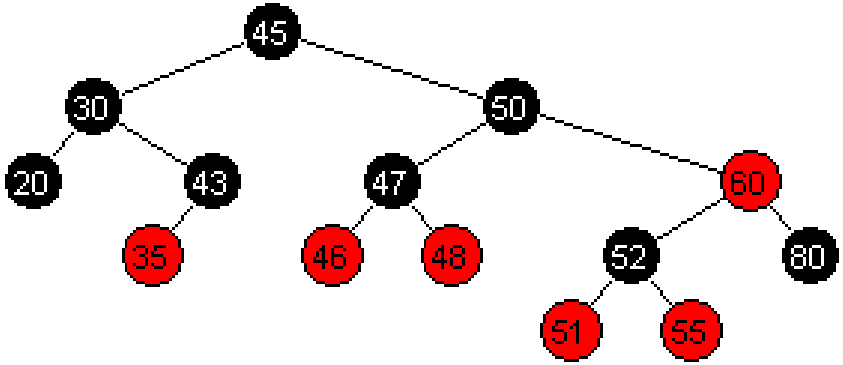
Po dodaniu (*insert*) węzła 51:



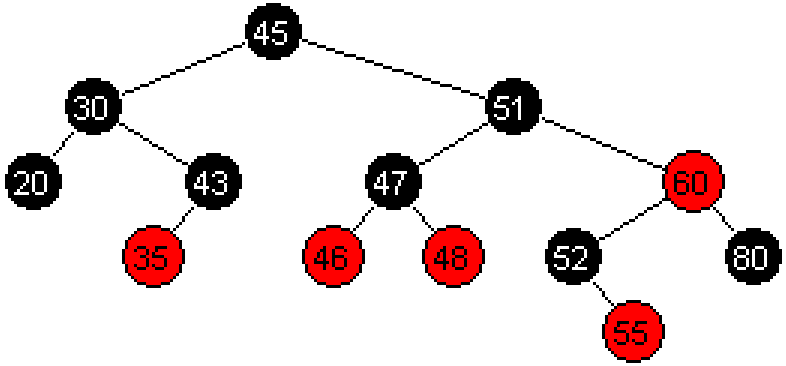
Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)



Po usunięciu (*delete*) węzła 50:



Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)

Operacje insert i delete powodują naruszenie właściwości RBT, co wymusza szereg działań przywracających te właściwości.

Przykładowo, przy wstawianiu elementu:
- Początkowo wstawiamy element tak, jak do standardowego drzewa BST.
- Kolor każdego nowo dodanego elementu jest czerwony.
- Jeżeli rodzic wstawionego węzła jest czarny to własność drzewa została zachowana.
- Jeżeli rodzic wstawionego węzła jest czerwony to własność 4 została zaburzona (rodzic i syn mają kolor czerwony).

Aby przywrócić własność należy przekolorować niektóre węzły i zmienić relację (wartości referencji) między niektórymi węzłami.

Są 3 możliwe przypadki:
1. Czerwony stryjek.
2. Czarny stryjek, prawy syn.
3. Czarny stryjek, lewy syn.

Każdy przypadek wymaga oddzielnego algorytmu.

Drzewa zbalansowane

Drzewa czerwono-czarne (*Red-Black Tree, RBT, RB-drzewa*)

Przy usuwaniu elementu należy zadbać dodatkowo o zachowanie równoważenia drzewa.

- Jeśli usuwany wierzchołek jest czerwony, czarna wysokość drzewa nie jest zakłócona.
- Jeśli usuwany wierzchołek jest czarny, czarna wysokość drzewa jest zakłócona i należy naprawić czarną wysokość dla każdej ścieżki w drzewie.

Są 4 możliwe przypadki:

1. Czerwony brat.
2. Czarny brat, czarny bratanek.
3. Czarny brat, lewy czerwony bratanek.
4. Czarny brat, prawy czerwony bratanek.

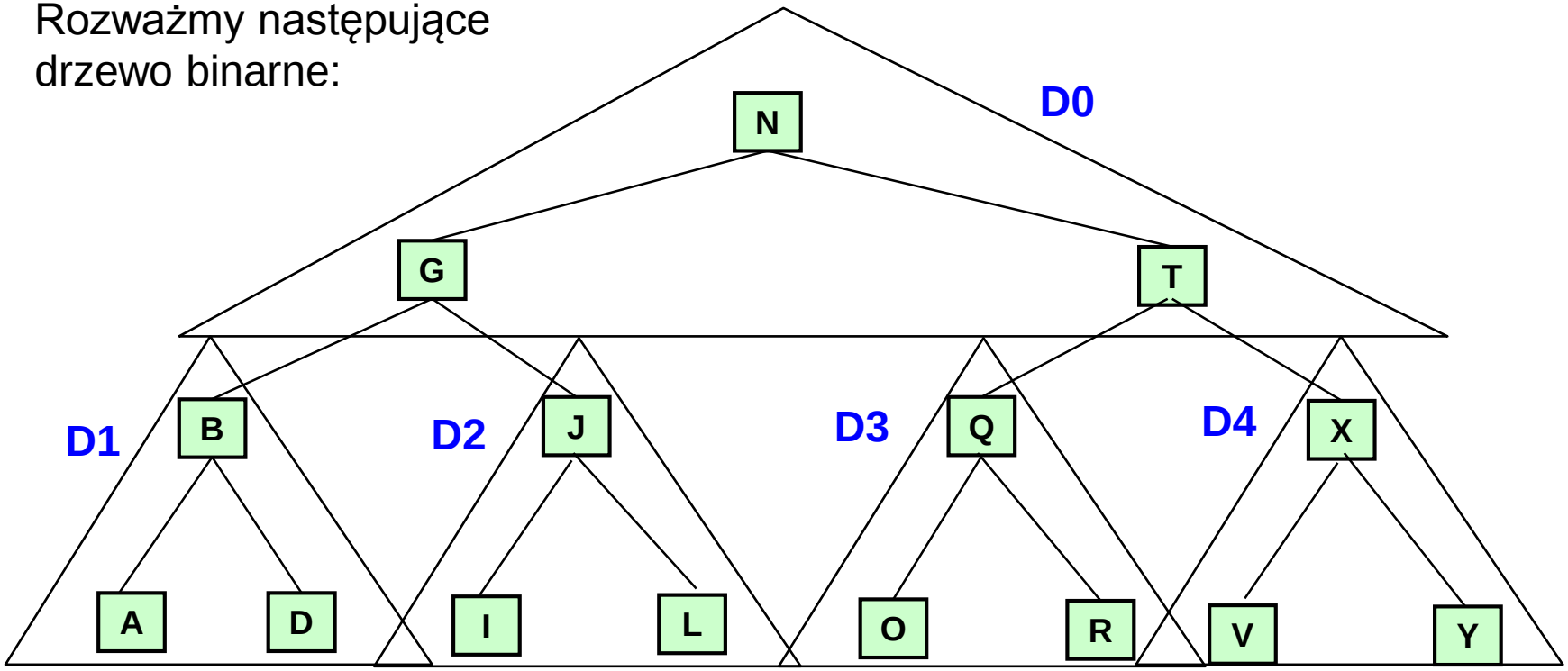
Szczegółów należy szukać w literaturze podstawowej (*Cormen*).

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

2-3-4-drzewa, B-drzewa (B-tree – *balanced tree*)

Rozważmy następujące drzewo binarne:



Wierzchołki zostały pogrupowane w „trójki”. Klucze wierzchołków drzewa:

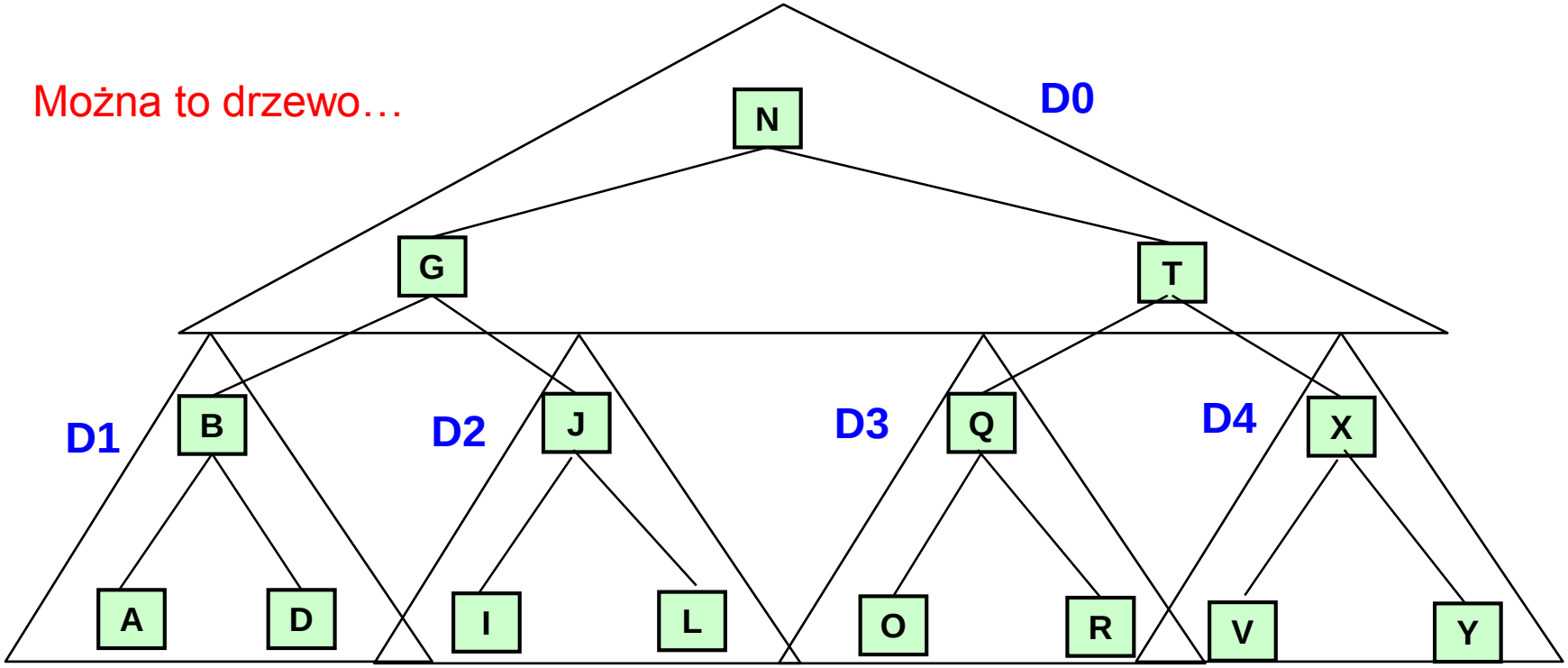
- D1 są mniejsze niż G,
- D2 są pomiędzy G a N,
- D3 są pomiędzy N a T,
- D4 są większe niż T

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

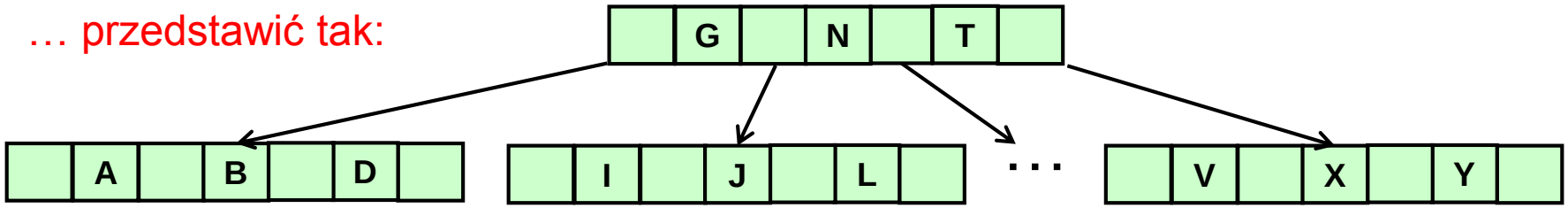
Drzewa zbalansowane

2-3-4-drzewa, B-drzewa (B-tree – *balanced tree*)

Można to drzewo…



… przedstawić tak:



Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

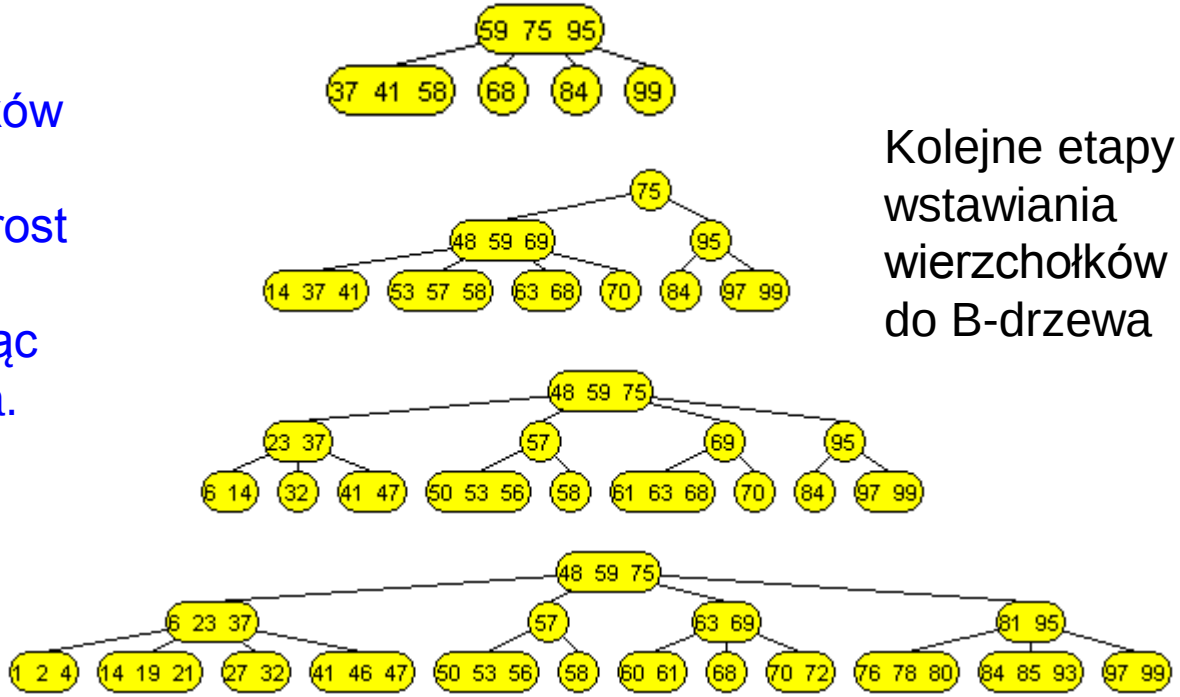
Drzewa doskonale zbalansowane

B-drzewa

B-drzewo klasy t(h, m) to drzewo skierowane, które spełnia następujące warunki:

1. Wszystkie drogi prowadzące z korzenia drzewa do liści są jednakowej długości (h) równej wysokości drzewa.
2. Każdy wierzchołek, z wyjątkiem korzenia i liści, ma co najmniej m+1 synów. Korzeń jest albo liściem, albo ma co najmniej dwóch synów.
3. Każdy wierzchołek ma co najwyżej 2m+1 synów.

Wstawianie wierzchołków (wartości kluczy) powoduje łagodny rozrost drzewa w kierunku do korzenia, nie naruszając zbalansowania drzewa.



Kolejne etapy wstawiania wierzchołków do B-drzewa

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

B-drzewa

1. B-drzewo (*perfectly balanced multiway tree*) jest drzewem wyszukiwania o takich ograniczeniach, które zapewniają, że jest ono zawsze w pełni zrównoważone, a stopień jego wypełnienia nie jest nigdy zbyt mały.
   Komplikuje to algorytmy wstawiania i usuwania, ale przyspiesza wyszukiwanie, ponieważ:
   - wszystkie drogi od korzenia do liści są jednakowej długości h,
   - każdy wierzchołek z wyjątkiem korzenia ma, co najmniej m+1 następników (poddrzew),
   - każdy wierzchołek ma co najwyżej 2m+1 następników (poddrzew),
   - korzeń ma, co najmniej 2 następniki - poddrzewa.

2. B-drzewa umożliwiają wyszukiwanie, dołączanie i usuwanie w czasie logarytmicznym (należy poznać te metody studiując literaturę!).

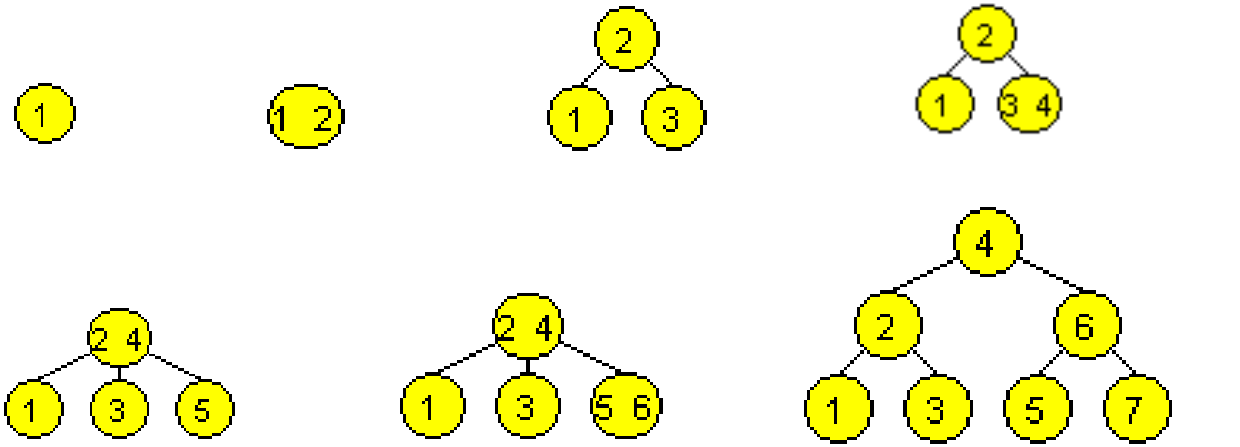
3. B-drzewa i ich warianty (B+-drzewa, przechowujące wartości kluczy tylko w liściach) stanowią podstawę implementacji indeksów w komercyjnych systemach baz danych.

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

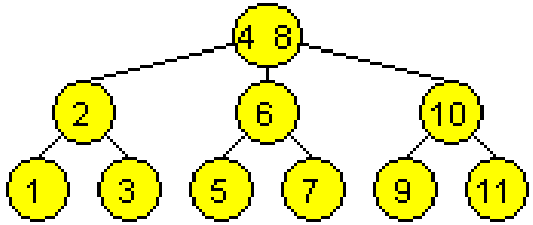
Drzewa zbalansowane

B-drzewa

Przykład wstawiania uporządkowanego ciągu liczb (wartości kluczy) do B-drzewa (dla zmniejszenia rozmiaru węzła m=3).
Kolejne postaci drzewa:



….. Po wstawieniu 11:



Drzewo niezmiennie wykazuje doskonałe zbalansowanie

Politechnika Wrocławska, Wydział Informatyki i Zarządzania
Zbigniew Szpunar

Drzewa zbalansowane

B-drzewa – pytania kontrolne

Jaką maksymalną liczbę wartości kluczy można przechować w B-drzewie t(3,10)?

Jaką wysokość musi mieć B-drzewo klasy t(h, 5), aby przy maksymalnym wypełnieniu zaindeksować plik liczący 14 500 rekordów?