Algorytmy i struktury danych

Drzewa

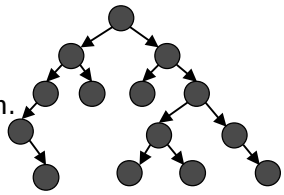
Witold Marańda
maranda@dmcs.p.lodz.pl

1

Drzewa - podstawy

Drzewo jest dynamiczną strukturą danych składającą się z elementu węzłowego, zawierającego wskazania na skończoną liczbę rozłącznych węzłów, stanowiących poddrzewa danego węzła.

- Węzeł początkowy (najwyższy) nazywany jest korzeniem.
- Element nie mający potomków (wskazań na inne węzły) nazywany jest liściem.
- Element nie będący liściem i korzeniem nazywamy węzłem wewnętrznym.

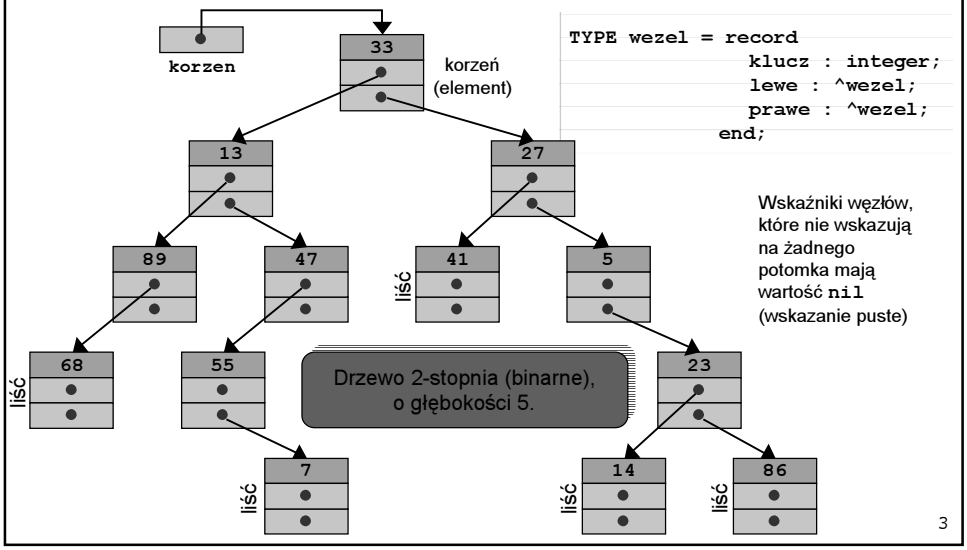


- Korzeń drzewa znajduje się na poziomie 1, a każdy węzeł potomny znajduje się na poziomie o jeden wyższym.
- Maksymalny poziom wszystkich elementów jest nazywany głębokością drzewa.
- Liczbę bezpośrednich potomków węzła nazywamy stopniem węzła.
- Maksymalny stopień wszystkich węzłów jest stopniem drzewa.
- Liczbę gałęzi od korzenia do węzła nazywa się długością drogi wewnętrznej.

2

Drzewa - podstawy

Szczególnie istotne okazują się drzewa binarne, tj. 2 stopnia (każdy węzeł ma co najwyżej 2 potomków).
(Najlepszym przykładem drzewa binarnego jest drzewo genealogiczne, gdzie rodzice każdej osoby traktowani są jako węzły potomne).




```
TYPE wezel = record
               klucz : integer;
               lewe : ^wezel;
               prawe : ^wezel;
             end;
```

Wskaźniki węzłów, które nie wskazują na żadnego potomka mają wartość `nil` (wskazanie puste)

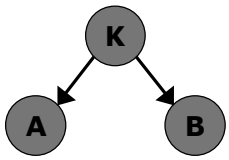
3

Przeglądanie drzew binarnych

Przeglądanie drzewa oznacza czynność odwiedzenia wszystkich elementów drzewa zwykle w celu wykonania na nich operacji **P**.

Rozróżnia się trzy porządki przeglądania drzewa:

1. wzdłużny: K, A, B
2. poprzeczny: A, K, B
3. wsteczny: A, B, K

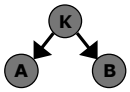


Ponieważ drzewa są strukturą danych o definicji rekurencyjnej (tj. drzewo składa się z poddrzew) procedury przeglądania drzew najłatwiej sformułować rekurencyjnie.

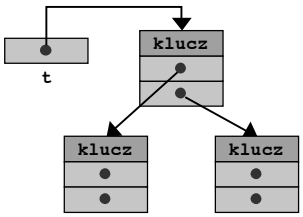
4

Przeglądanie drzew binarnych

1. wzdłużny: K, A, B
2. poprzeczny: A, K, B
3. wsteczny: A, B, K




```
PROCEDURE wzdluzny(t: ^wezel)
BEGIN
   IF t<>nil THEN
   BEGIN
       P(t^);
       wzdluzny(t^.lewe);
       wzdluzny(t^.prawe);
   END
END
```




```
PROCEDURE poprzeczny(t: ^wezel)
BEGIN
   IF t<>nil THEN
   BEGIN
       poprzeczny(t^.lewe);
       P(t^);
       poprzeczny(t^.prawe);
   END
END
```

```
PROCEDURE wsteczny(t: ^wezel)
BEGIN
   IF t<>nil THEN
   BEGIN
       wsteczny(t^.lewe);
       wsteczny(t^.prawe);
       P(t^)
   END
END
```

5

Wyszukiwanie elementu w drzewie binarnym

Wyszukiwanie elementu (o żądanej wartości klucza) w drzewie niezorganizowanym może korzystać z dowolnej procedury przeglądania drzewa, uzupełniając ją o operację P:

```
IF t^.klux=x THEN ...
```

np.:

```
PROCEDURE wzdluzny(t: ^wezel; x:integer)
BEGIN
   IF t<>nil THEN
   BEGIN
       IF t^.klucz=x THEN
       BEGIN
            {operacje na kluczu}
       END
       P(t^);
       wzdluzny(t^.lewe,x);
       wzdluzny(t^.prawe,x);
   END
END
```

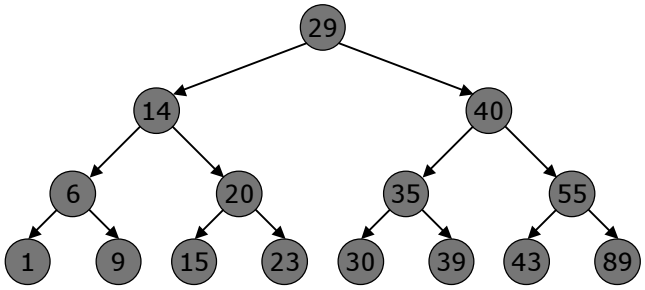
Złożoność obliczeniowa takiego wyszukiwania dla *n*-elementowego drzewa jest równa $O(n)$.

6

Wyszukiwanie elementu w zorganizowanym drzewie binarnym

Drzewa dobrze nadają się do przechowywania elementów w postaci zorganizowanej, co znacznie przyspiesza operację wyszukiwania (złożoność obliczeniowa $O(\log(n))$).

Rozważmy drzewo, w którym dla każdego węzła wszystkie klucze jego lewego poddrzewa są mniejsze o klucza danego węzła, a prawego – większe.



7

Wyszukiwanie elementu w zorganizowanym drzewie binarnym

Funkcja **`znajdz(t,x)`** wymaga podania wskaźnika korzenia oraz poszukiwanego klucza, a zwraca wskaźnik do elementu o szukanym kluczu (lub nil gdy elementu nie znaleziono).

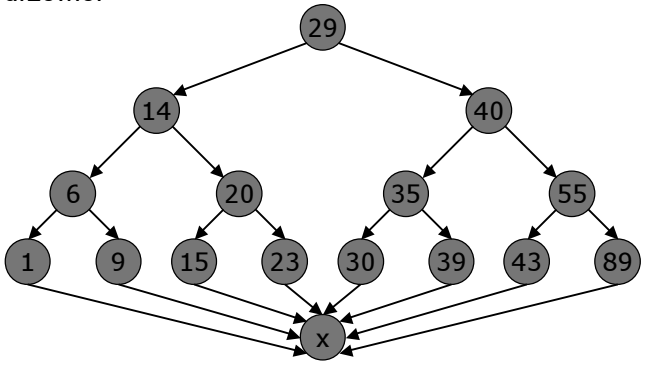
```
FUNCTION znajdz(x: integer; t: ^wezel): ^wezel;
VAR
   znaleziony: boolean;
BEGIN
   znaleziony:=FALSE;
   WHILE (t<>nil) AND NOT znaleziony DO
       IF t^.klucz=x THEN
            znaleziony:=TRUE
       ELSE
            IF t^.klucz>x THEN t:=t^.lewe ELSE t:=t^.prawe;
   znajdz:=t;
END
```

8

Wyszukiwanie elementu w drzewie z wartownikiem

Operację wyszukiwania elementu w drzewie zorganizowanym można znacznie uprościć poprzez wstawienie elementu kotwicznego dla wszystkich liści (wartownika).

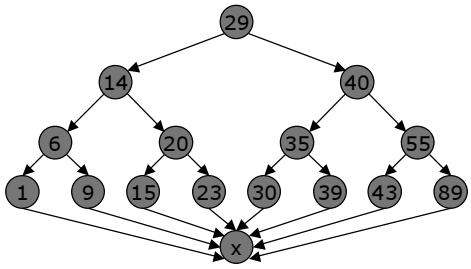
Podczas wyszukiwania ustawia się wartość klucza wartownika równą poszukiwanemu kluczowi, co zapewnia poprawne zakończenie wyszukiwania, również w przypadku nie znalezienia elementu o danym klucza w drzewie.



9

Wyszukiwanie elementu w drzewie z wartownikiem

Funkcja **`znajdz(t,x)`** wymaga, jak poprzednio, podania wskaźnika korzenia oraz poszukiwanego klucza, a zwraca wskaźnik do elementu o szukanym kluczu, lub wskazanie na wartownika gdy elementu nie znaleziono.




```
FUNCTION znajdz(x: integer; t: ^wezel): ^wezel;
BEGIN
   wartownik^.klucz:=x;
   WHILE t^.klucz<>x DO
       IF x<t^.klucz THEN t:=t^.lewe ELSE t:=t^.prawe;
   znajdz:=t;
END
```

10