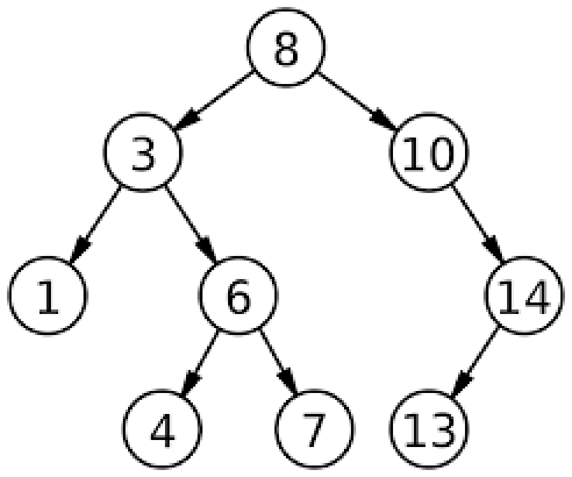
Drzewo binarne

Wojciech Macyna

Drzewo binarne

Przykład



Rysunek: *Przykład drzewa*

Komentarz

Informacje ogólne

- Drzewo binarne składa się z węzłów (elementów).
- Każdy węzeł może mieć maksymalnie dwóch synów oraz ojca. W przypadku korzenia (root) wskaźnik na ojca jest pusty.
- Klucz węzła jest zawsze większy od klucza węzła lewego syna oraz mniejszy lub równy od prawego.

Oznaczenia w algorytmach

- key[x] - klucz węzła x (węzeł może mieć inne pola oprócz kluczowego)
- left[x] - wskaźnik na lewego syna węzła x
- right[x] - wskaźnik na prawego syna węzła x
- p[x] - wskaźnik na ojca węzła x
- root[T] - wskaźnik na korzeń drzewa binarnego T

Drzewo binarne

Search

```
TREE-SEARCH(x, k)
    if x = NIL lub k = key[x]
        then return x
    if k < key[x]
        then return TREE-SEARCH(left[x], k)
        else return TREE-SEARCH(right[x], k)
```

Rysunek: *Tree Search*

Drzewo binarne

Insert

```
TREE-INSERT(T, z)
y ← NIL
x ← root[T]
while x ≠ NIL
    do y ← x
       if key[z] < key[x]
          then x ← left[x]
          else x ← right[x]
p[z] ← y
if y = NIL
   then root[T] ← z
   else if key[z] < key[y]
          then left[y] ← z
          else right[y] ← z
```

Rysunek: *Tree Insert*

Komentarz

Wyszukiwanie (search)

- Wyszukujemy węzeł o kluczu k w kontekście węzła x: jeśli k jest mniejszy od klucza węzła x, to szukamy w lewym poddrzewie; w przeciwnym przypadku, szukamy w prawym poddrzewie (linie 3 - 5).
- Jeśli k jest równe kluczowi węzła x, to zwracamy ten węzeł.
- Jeśli tam gdzie powinien być węzeł o kluczu k, nie ma nic, wówczas zwracamy wartość pustą (linie 1 - 2).

Wstawianie insert)

- W liniach 2-7 szukamy miejsca, gdzie powinien być wstawiony węzeł z;
- W linii 8 określamy ojca dla węzła z;
- W liniach 9 -12 podpinamy węzeł z pod lewego lub prawego syna węzła y.

Drzewo binarne

Delete

```
TREE-DELETE(T, z)
if left[z] = NIL lub right[z] = NIL
    then y ← z
    else y ← TREE-SUCCESSOR(z)
if left[y] ≠ NIL
    then x ← left[y]
    else x ← right[y]
if x ≠ NIL
    then p[x] ← p[y]
if p[y] = NIL
    then root[T] ← x
    else if y = left[p[y]]
        then left[p[y]] ← x
        else right[p[y]] ← x
if y ≠ z
    then key[z] ← key[y]
        ▷ Jeśli y ma inne pola, to je także należy skopiować.
return y
```

Drzewo binarne

Dodatkowe algorytmy

```
TREE-SUCCESSOR(x)
  if right[x] ≠ NIL
     then return TREE-MINIMUM(right[x])
  y ← p[x]
  while y ≠ NIL i x = right[y]
     do x ← y
        y ← p[y]
  return y
```

```
TREE-MINIMUM(x)
  while left[x] ≠ NIL
     do x ← left[x]
  return x
```

Rysunek: *Dodatkowe algorytmy*

Komentarz

Delete

Rozpatrywane są trzy przypadki:

- Jeżeli z nie ma synów, to w jego ojcu $p[z]$ zastępujemy wskaźnik do z wartością NIL.
- Jeżeli węzeł ma tylko jednego syna, to wycinamy z poprzez ustalenie wskaźnika pomiedzy jego ojcem a jedynym synem.
- Jeśli węzeł ma dwóch synów, to wycinamy następnik y węzła z, o którym wiadomo, że nie ma lewego syna oraz zastępujemy zawartość z zawartością y.

Nastepnik

Rozpatrywane są dwa przypadki:

- Jeżeli prawe poddrzewo węzła x jest niepuste, to następnikiem x jest najbardziej na lewo położony węzeł w prawym poddrzewie
- Jeżeli jednak węzeł x nie ma prawego poddrzewa, choć ma następnik y, to y jest najniższym przodkiem węzła x, którego lewy syn jest także przodkiem x.