Algorytmy 21.06.2013r.

1. Dla algorytmu A o rozmiarze danych n, $n \in \mathbb{N}$ uzupełnij definicję:
   $T_A(n) = O(nlogn)$ wtedy i tylko wtedy gdy:
   $\underset{n \geq n_0}{\forall} T_A(n) \leq c \cdot (n \log n)$, gdzie c to pewna stała
2. Wyznaczyć rząd pesymistycznej złożoności algorytmu:

```
P[1..m] , T[1..n],  n>=m>0

begin
  h := 10^(m-1) ;   p := 0 ;   t := 0 ;   k := 0;
  for i := 1 to m do
    begin
      p := (10 * p + P[i]);
      t := (10 * t + T[i]);
    end;
  for s := 0  to n-m  do
    begin
      if p = t  then  k := k+1;
      if s < n-m
        then t := 10 * (t – T[s+1] * h ) + T[s+m+1];
    end;
  return k ;
end;
```

Wyszło? $O(m+(n-m))$

3. Podaj rząd pesymistycznej złożoności algorytmów:
    - budowa kopca binarnego o kluczach z danego ciągu n liczb całkowitych: O(n)
    - wstawianie elementu o zadanym kluczu do BST n węzłach: O(logn)
    - algorytm Prima dla grafu o n wierzchołkach i m krawędziach: O(m logn)
    - przeglądanie drzewa binarnego o n węzłach metodą inorder: O(n)
    - algorytm Floyda-Warshalla dla grafu o n wierzchołkach: $O(n^3)$
4.
    - algorytm Grahama: zachłanny
    - algorytm Floyda-Warshalla: dynamiczny
    - algorytm sortowania przez wstawianie: przyrostowy
    - algorytm Kruskala: zachłanny
5. Dany jest zbiór odcinków na płaszczyźnie :
   {a = [(3,1), (7,6)], b = [(0,2), (2,2)], c =[(1,5), (6,1)], d =[(1,5), (6,1)], e=[(0,1), (7,2)],}
   Zastosuj algorytm na sprawdzanie czy w zbiorze istnieje para przecinających się odcinków. Podaj wszystkie stany „miotły" od początku działania algorytmu aż do znalezienia pierwszej pary przecinających się odcinków.

   ??????????????????????????????????????
6. Podaj definicję binarnego drzewa przeszukiwań BST:

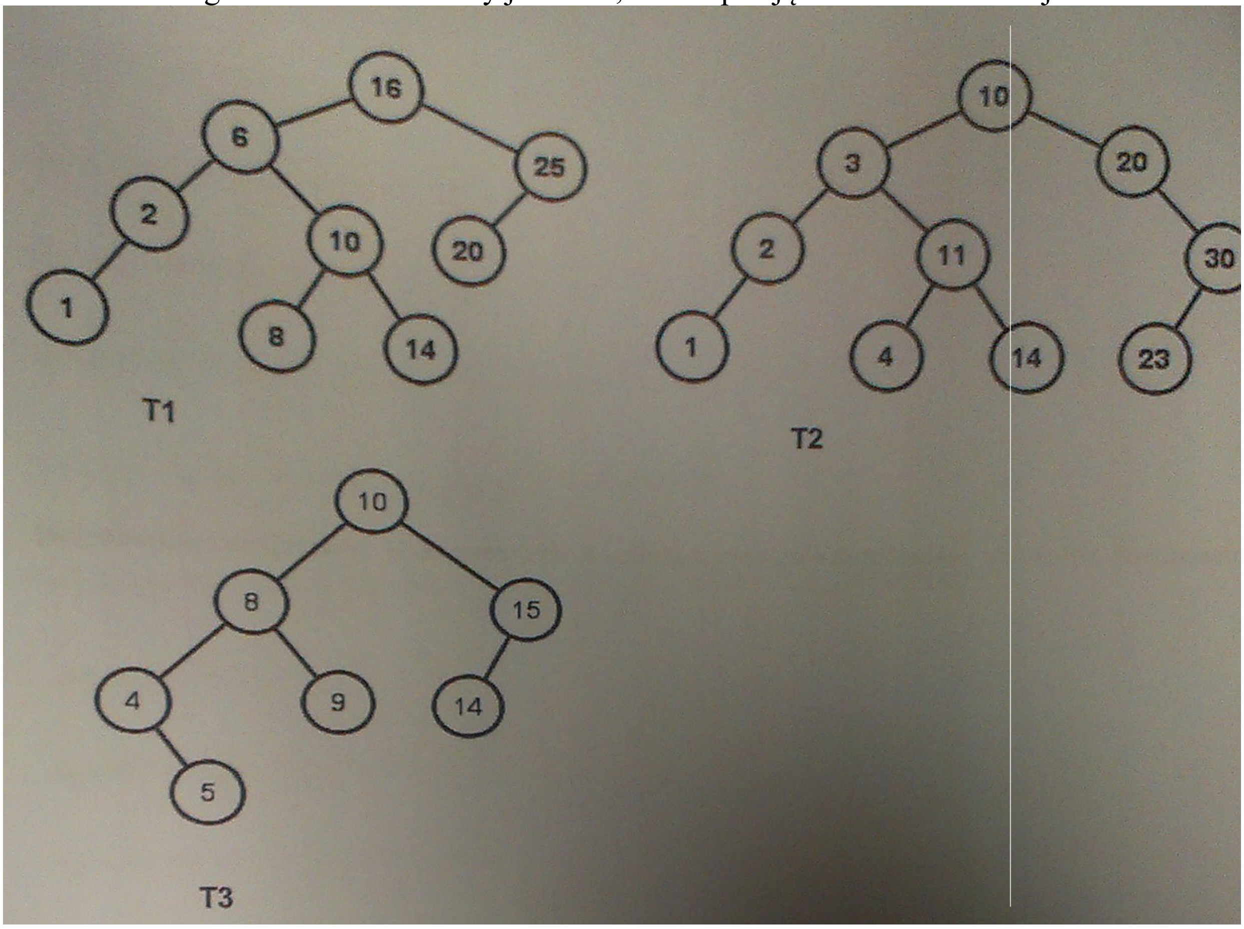
Klucze są przechowywane w drzewie **BST** w taki sposób, aby spełniona była **własność drzewa BST**:

Niech x będzie węzłem drzewa **BST**. Jeśli y jest węzłem znajdującym się w lewym poddrzewie węzła x, to $key[y] \leqslant key[x]$. Jeśli y jest węzłem znajdującym się w prawym poddrzewie węzła x, to $key[x] \leqslant key[y]$.

7. Podaj pseudokod procedury wstawianie elementu o kluczu k do drzewa BST w wersji nierekurencyjnej.

```
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
```

8. Dla każdego drzewa zaznacz czy jest BST, AVL wpisując tak/nie i uzasadnij:



T1: BST: tak?, AVL: tak?
T2: BST: ??? AVL: nie jest, bo nie jest zrównoważone
T3: BST: tak? AVL: tak?

9. Podaj definicję funkcji prefiksowej **π** używanej w algorytmie KMP i wyznacz jej wartość

Funkcja prefiksowa dla wzorca

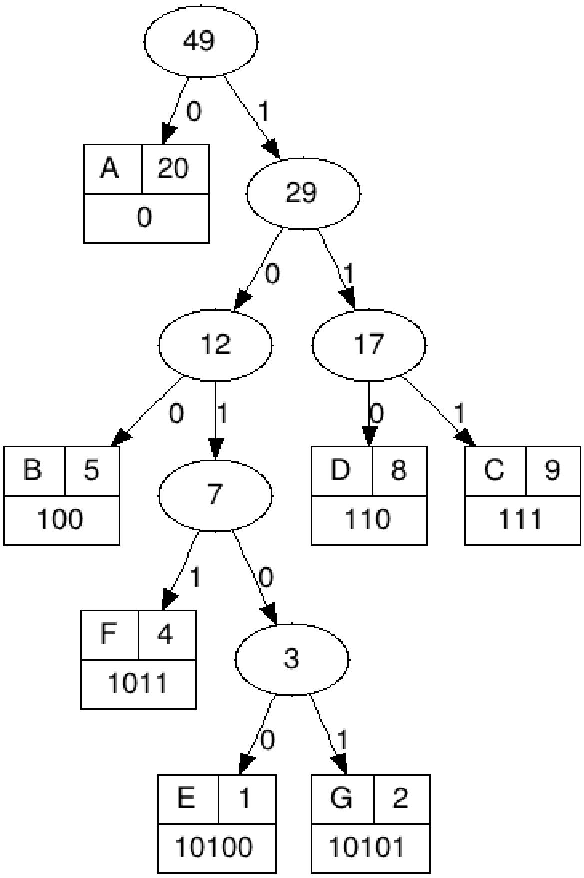
Funkcja prefiksowa dla wzorca zawiera informacje związane z porównywaniem wzorca z samym sobą. Informacja ta może zostać użyta do uniknięcia zbytecznych testów w algorytmie „naiwnym" lub w czasie wykorzystywania funkcji δ dla automatu wyszukiwania wzorca.

972

dla wzorca P=babbabab.

Π[]: b a b b a b a b
0,0,1,1,2,3,2,3

10. Dany jest tekst PL złożony z liter: {A,B,C,D,E,F,G}.
    Narysuj drzewo wyznaczone przez algorytm Huffmana i podaj kody liter.
    Częstość wystąpień: A:20, B:5, C:9, D:8, E:1, F:4, G:2



11. Podaj specyfikację algorytmu Kruskala (oraz definicje pojęć użytych w wyniku):
    Dane:Kruskal(G, w):
    G=(V,E) jest grafem, w = (tutaj zależy)
    Wynik: ET: zbiór krawędzi tworzących MST

    Definicja MST: minimalne drzewo rozpinające jest acykliczne i łączy wszystkie wierzchołki krawędziami o najmniejszych wagach.

Algorytm Kruskala

Algorytm Kruskala jest bezpośrednio oparty na schemacie obliczania minimalnego drzewa rozpinającego z podrozdz. 24.1. W tym algorytmie krawędzią

568

24.2. ALGORYTMY KRUSKALA I PRIMA

dodawaną do rozrastającego się lasu jest krawędź (u, v) o najmniejszej wadze spośród krawędzi łączących różne drzewa w lesie. Niech C_1 i C_2 oznaczają dwa drzewa, które są połączone krawędzią (u, v). Ponieważ (u, v) jest krawędzią lekką łączącą C_1 z innym drzewem, z wniosku 24.2 wynika, że (u, v) jest krawędzią bezpieczną dla C_1. Algorytm Kruskala jest algorytmem zachłannym, ponieważ w każdym kroku do lasu jest dodawana krawędź o najmniejszej możliwej wadze.