Wstęp do Informatyki 2023/2024
Lista 11

Instytut Informatyki, Uniwersytet Wrocławski

5, 9 i 17 stycznia 2024

Zadania na tej liście dotyczą drzew binarnych tworzonych z użyciem następujących definicji typów lub konstruktorów:

```
typedef struct node* pnode;
typedef struct node {
  int val;
  pnode left;
  pnode right; } snode;
```

```
class TreeItem:
  def __init__ (self, value):
    self.val = value
    self.left = None
    self.right = None
```

W rozwiązaniach zadań w języku C można korzystać z funkcji `pnode utworz(int wart)` tworzącej drzewo z jednym węzłem z wartością `val` równą `wart` (p. notatka do wykładu).

1. [1] Do drzewa BST (na początku pustego) wstawiane są elementy $1, 2, 3, 4, 5, 6, 7$. Podaj kolejność wstawiania elementów, przy której drzewo będzie miało największą/najmniejszą możliwą wysokość. Odpowiedź uzasadnij i uogólnij na przypadek ciągu liczb $1, 2, \ldots, 2^k–1$ dla dowolnego naturalnego $k > 1$.

2. [1] Napisz funkcję, która dla parametru `t` wskazującego na korzeń drzewa binarnego zwraca jako wartość liczbę elementów w drzewie o korzeniu `t`.

3. [1] Napisz funkcję, która dla parametru `t` zwraca jako wartość wysokość drzewa o korzeniu `t`.

4. [1] Napisz funkcję, która dla parametru `t` opisującego drzewo BST wypisuje (w porządku niemalejącym) wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu `t`.

   Oszacuj złożoność czasową Twojego rozwiązania i odpowiedz z uzasadnieniem na pytanie czy w najgorszym przypadku możliwe jest uzyskanie mniejszej złożoności czasowej.

5. [2] Napisz funkcję, która dla danego drzewa binarnego sprawdza czy jest ono drzewem BST.

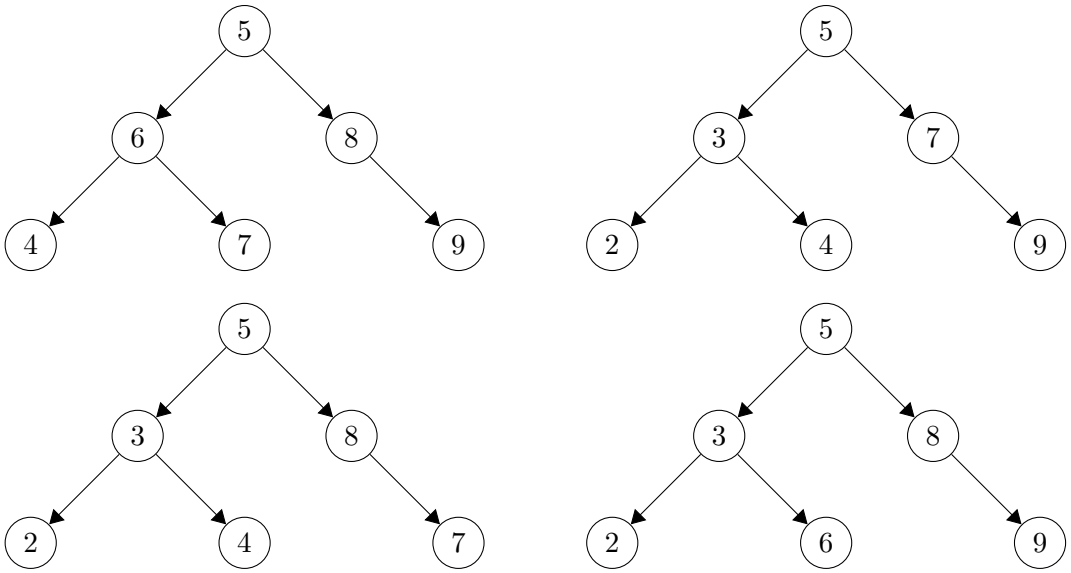
6. [1] Napisz funkcję, która łączy dwa drzewa BST w jedno drzewo przy założeniu, że największa wartość klucza (`val`) w pierwszym drzewie jest mniejsza od najmniejszej wartości klucza w drugim drzewie. Czas działania Twojej funkcji powinien być $O(h)$, gdzie h to wysokość pierwszego drzewa.

7. [1] Napisz funkcję wstawiającą element o podanym kluczu do drzewa BST bez użycia rekurencji.

8. [1] Rotacją nazywamy operację przebudowy drzewa BST tak, aby wskazany węzeł u i jego wskazane dziecko v „zmieniły miejsca" w taki sposób, że u stanie się dzieckiem v. Podaj sposób na wykonanie rotacji w sytuacji, gdy v jest lewym dzieckiem u, napisz funkcję/algorytm realizującą zaprezentowaną przez Ciebie metodę.

Zadania dodatkowe, nieobowiązkowe (nie wliczają się do puli punktów do zdobycia na ćwiczeniach, punktacja została podana tylko jako informacja o trudności zadań wg wykładowcy)

9. [0] Które z poniższych drzew są drzewami BST?



10. [0.5] Opisz efekt działania poniższych funkcji dla dowolnego drzewa binarnego `t` i dla drzewa BST.

```
void write1(pnode t) {
  if (t != NULL) {
    printf("%d\n", t->val);
    write1(t->left);
    write1(t->right); } }

void write2(pnode t) {
  if (t != NULL) {
    write2(t->left);
    printf("%d\n", t->val);
    write2(t->right); } }
```

```
def write1(t):
  if t != None:
    print(t.val)
    write1(t.left)
    write1(t.right)

def write2(t):
  if t != None:
    write2(t.left)
    print(t.val)
    write2(t.right)
```

11. [2] Napisz funkcję, która dla parametru `t` opisującego drzewo BST wypisuje (w porządku rosnącym) wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu `t`. Czas działania Twojej funkcji powinien być $O(h + m)$, gdzie h to wysokość drzewa t, a m to liczba elementów dodatnich w drzewie.

12. [1] Napisz funkcję wyszukującą element o podanym kluczu w drzewie BST bez użycia rekurencji.

13. [1] Napisz funkcję usuwającą element o podanym kluczu z drzewa BST bez użycia rekurencji.

14. [2] Napisz funkcję, która dla parametru `t` opisującego drzewo BST wypisuje w porządku niemalejącym wszystkie elementy dodatnie znajdujące się w drzewie o korzeniu `t`, bez użycia rekurencji.

15. [1] Dla funkcji wyszukującej element w drzewie BST utwórz wersję, w której operacja ta będzie przyspieszona przez użycie wartownika.

    Wskazówka: wszystkie wskaźniki `NULL`/`None` zastąp wskaźnikami do jednego węzła, w którym umieszczasz szukany klucz przed przystąpieniem do wyszukiwania.