Powrót

## Twój wynik: 4 punktów na 6 możliwych do uzyskania (66,67 %).

| Nr | Opcja | Punkty | Poprawna | Odpowiedź |
|---|---|---|---|---|
| 1 | Rozważmy algorytm AVLSequence postaci: (see below) | | | |

Rozważmy algorytm AVLSequence postaci:

```
AVLTree AVLSequence(AVLTree T, sequence S[]) {
  // T - drzewo początkowe typu AVL, tuż po zastosowniu operacji AVLConstruct
  // S - tablica sekwencji operacji słownikowych

  int i;

  for (i:=0; i<size(S); i++) do
    execute_sequence(T,S[i]);

  return T;
}
```

Niech drzewo $T$ będzie rezultatem działania algorytmu AVLSequence dla danych wejściowych:

- drzewo początkowe $T = AVLConstruct([9,14,11,12,0,15,13,5,19,8])$, gdzie algorytm AVLConstruct jest standardowym algorytmem budowy drzewa typu AVL przez kolejne wstawianie elementów,
- sekwencja operacji słownikowych $S$ :
  1. $INSERT(T,19)$
  2. $INSERT(T,5)$
  3. $DELETE(INSERT(T,8),15)$
  4. $MEMBER(T,7) \Rightarrow DELETE(T,19)$
  5. $INSERT(DELETE(T,13),1)$

Które z poniższych zdań jest prawdziwe? Uwaga! W trakcie wykonywania operacji DELETE w miejsce usuwanego wierzchołka wstawiamy wierzchołek bezpośrednio następny względem porządku etykiet.

| Nr | Opcja | Punkty | Poprawna | Odpowiedź |
|---|---|---|---|---|
| | Maksymalna wysokość drzewa AVL $T$ w trakcie wykonania przedstawionego ciągu operacji jest taka sama jak w przypadku wykonania następującego ciągu operacji: $MEMBER(T,16) \Rightarrow INSERT(T,8)$, $DELETE(INSERT(T,10),6)$, $DELETE(T,12)$, $MEMBER(T,10) \Rightarrow INSERT(T,14)$, $INSERT(T,7)$ | 1 | + | + |

| | | | | |
|---|---|---|---|---|
| | Ostateczna wysokość drzewa AVL $T$ tuż po wykonaniu przedstawionego ciągu operacji jest równa dokładnie $4$ | 0 | | |
| | Maksymalna wysokość drzewa AVL $T$ w trakcie wykonania przedstawionego ciągu operacji jest równa dokładnie $2$ | 0 | | |

2

Rozważmy algorytm AVLConstruct postaci:

```
AVLTree AVLConstruct(element E[]) {
  // E - tablica elementów

  AVLTree T; // drzewo typu AVL początkowo puste
  int i;

  for (i:=0; i<size(E); i++) do
    INSERT(T,E[i]);

  return T;
}
```

Które z poniższych zdań jest prawdziwe, jeżeli $size(E)=N$ oraz $N=n^{\frac{3}{2}}$?

| | | | | |
|---|---|---|---|---|
| | Niech $S(N)$ oznacza złożoność pamięciową algorytmu AVLConstruct (implementacja rekurencyjna operacji słownikowych) dla danych rozmiaru $N$, wtedy: $S(N)=O(\log^5 n)$ | 1 | + | + |
| | Niech $W(N)$ oznacza złożoność czasową algorytmu AVLConstruct dla danych rozmiaru $N$, w pesymstycznym przypadku, mierzoną liczbą porównań etykiet wierzchołków konstruowanego drzewa, wtedy: $W(N)=O(2^{\frac{n}{2}})$ | 0 | | |
| | Niech $A(N)$ oznacza złożoność czasową algorytmu AVLConstruct dla danych rozmiaru $N$, w średnim przypadku, mierzoną liczbą porównań etykiet wierzchołków konstruowanego drzewa, wtedy: $A(N)=\Theta(2^n)$ | 0 | | |

3

Rozważmy algorytm HashTableSequence postaci:

```
HashTable HashTableSequence(HashTable HT, sequence S[]) {
  // HT - początkowa tablica haszująca, tuż po zastosowaniu operacji HashTableConstruct
  // S - tablica sekwencji operacji słownikowych

  int i;

  for (i:=0; i<size(S); i++) do
    execute_sequence(HT,S[i]);

  return HT;
}
```

Które z poniższych zdań jest prawdziwe, jeżeli $N$ jest początkową liczbą elementów przechowywanych w tablicy haszującej $HT$ a $size(S)=N$ oraz $N=n^{\sqrt{3}}$? Uwaga! Problem kolizji w rozważanej strukturze rozwiązany jest za pomocą list.

| | | | |
|---|---|---|---|
| Niech $A(N)$ oznacza złożoność czasową algorytmu HashTableSequence dla danych rozmiaru $N$, w średnim przypadku, mierzoną liczbą operacji na listach, wtedy: $A(N)=O\left(n^{\sqrt{3}}\right)$ | 1 | + | + |
| Niech $T(N)$ oznacza złożoność czasową algorytmu HashTableSequence dla danych rozmiaru $N$, w każdym przypadku, mierzoną liczbą operacji słownikowych, wtedy: $T(N)=O\left(n^{10}\right)$ | 1 | + | + |
| Niech $T(N)$ oznacza złożoność czasową algorytmu HashTableSequence dla danych rozmiaru $N$, w każdym przypadku, mierzoną liczbą operacji na listach, wtedy: $T(N)=\Omega(1)$ | 1 | + | + |

4

Rozważmy algorytm BSTDestroy postaci:

```
BSTTree BSTDestroy(BSTTree T, element E[]) {
   // T - drzewo początkowe typu BST, tuż po zastosowaniu operacji BSTConstruct
   // E - tablica usuwanych elementów

 int i;


   for (i:=0; i<size(E); i++) do
     DELETE(T,E[i]);


   return T;
}
```

Niech drzewo $T$ będzie rezultatem działania algorytmu BSTDestroy dla danych wejściowych:

- drzewo początkowe $T = BSTConstruct([0,7,9,14,11,6,15,17,5,16])$, gdzie algorytm BSTConstruct jest standardowym algorytmem budowy drzewa typu BST przez kolejne wstawianie elementów,
- tablica usuwanych elementów $E = [14,0,16,5,11]$.

Które z poniższych zdań jest prawdziwe? Uwaga! W trakcie wykonywania operacji DELETE w miejsce usuwanego wierzchołka wstawiamy wierzchołek bezpośrednio następny względem porządku etykiet.

| | | | | |
|---|---|---|---|---|
| | Etykiety wierzchołków drzewa $T$ wypisane w kolejności PostOrder tworzą ciąg: $6,7,9,15,17$ | 0 | | + |
| | Wysokość drzewa $T$ jest równa dokładnie $2$ | 0 | | |
| | Etykiety wierzchołków drzewa $T$ wypisane w kolejności PostOrder tworzą ciąg: $6,17,15,9,7$ | 1 | + | |

5 Rozważmy algorytm HashTableSequence postaci:

```
HashTable HashTableSequence(HashTable HT, sequence S[]) {
  // HT - początkowa tablica haszująca, tuż po zastosowaniu operacji HashTableConstruct
  // S - tablica sekwencji operacji słownikowych

  int i;

  for (i:=0; i<size(S); i++) do
    execute_sequence(HT,S[i]);

  return HT;
}
```

Niech tablica haszująca $HT$ będzie rezultatem działania algorytmu HashTableSequence dla danych wejściowych:

- początkowa talica haszująca $HT = HashTableConstruct([2,8,1,16,14,0,18,12,13,10],k,f)$, gdzie algorytm HashTableConstruct jest standardowym algorytmem budowy tablicy haszującej przez kolejne wstawianie elementów,
- rozmiar tablicy haszującej $k=5$,
- funkcja haszująca $f:\mathbb{N}\rightarrow\{0,1,\ldots,4\}$ postaci $f(x)=x \mod 5$.
- sekwencja operacji słownikowych $S$ :
  1. $MEMBER(HT,5)\Rightarrow DELETE(HT,13)$
  2. $DELETE(INSERT(HT,10),1)$
  3. $MEMBER(HT,17)\Rightarrow INSERT(HT,14)$
  4. $MEMBER(HT,15)\Rightarrow INSERT(HT,12)$
  5. $INSERT(DELETE(HT,17),13)$

Które z poniższych zdań jest prawdziwe? Uwaga! Problem kolizji w strukturze $HT$ rozwiązany jest za pomocą list zorganizowanych w trybie LIFO.

| | | | | |
|---|---|---|---|---|
| | $MEMBER(HT,7)=TRUE$ | 0 | | |
| | Minimalna długość listy będącej elementem tablicy haszującej $HT$ na zakończenie wykonania przedstawionego ciągu operacji jest jest taka sama jak w przypadku wykonania następującego ciągu operacji: $DELETE(INSERT(HT,7),12)$, $MEMBER(HT,18)\Rightarrow DELETE(HT,19)$, $MEMBER(HT,6)\Rightarrow INSERT(HT,5)$, $INSERT(DELETE(HT,17),6)$, $MEMBER(HT,5)\Rightarrow DELETE(HT,5)$ | 1 | + | + |
| | Łączna liczba kolizji elementów, które wystąpiły w trakcie wykonaniu przedstawionego ciągu operacji na strukturze $HT$ wynosi dokładnie $0$ | 1 | + | |

6

Rozważmy algorytm BSTSequence postaci:

```
BSTTree BSTSequence(BSTTree T, sequence S[]) {
  // T - drzewo początkowe typu BST, tuż po zastosowaniu operacji BSTConstruct
  // S - tablica sekwencji operacji słownikowych

  int i;

  for (i:=0; i<size(S); i++) do
    execute_sequence(T,S[i]);

  return T;
}
```

Które z poniższych zdań jest prawdziwe jeżeli jeżeli $N$ jest początkową liczbą wierzchołków drzewa $T$ a $size(S)=N$ i $N=n^2$?

| | | | | |
|---|---|---|---|---|
| | Niech $T(N)$ oznacza złożoność czasową algorytmu BSTSequence dla danych rozmiaru $N$, w każdym przypadku, mierzoną liczbą operacji słownikowych, wtedy: $T(N)=\Theta(n \log n)$ | 0 | | |
| | Niech $T(N)$ oznacza złożoność czasową algorytmu BSTSequence dla danych rozmiaru $N$, w każdym przypadku, mierzoną liczbą porównań etykiet wierzchołków drzewa, wtedy: $T(N)=\Omega(3^n)$ | 0 | | |
| | Niech $S(N)$ oznacza złożoność pamięciową algorytmu BSTSequence (implementacja iteracyjna operacji słownikowych) dla danych rozmiaru $N$, wtedy: $S(N)=O(2^n)$ | 1 | + | + |