Programowanie i projektowanie obiektowe

Struktury danych

Paweł Daniluk

Wydział Fizyki

Jesień 2011

Listy

Podstawowe cechy

- Tablicowa
- Ciąg węzłów połączonych referencjami.
- Jednokierunkowe lub dwukierunkowe
- Cykliczne
- Z wartownikiem

Listy

Podstawowe cechy

- Tablicowa
- Ciąg węzłów połączonych referencjami.
- Jednokierunkowe lub dwukierunkowe
- Cykliczne
- Z wartownikiem

Zastosowania

- Stosy i kolejki
- Możliwość wstawiania i usuwania elementów ze środka

Drzewa

Charakterystyka

- korzeń, węzły, liście
- stopień węzła (drzewa binarne)
- głębokość liścia
- ścieżki w drzewie

Drzewa

Charakterystyka

- korzeń, węzły, liście
- stopień węzła (drzewa binarne)
- głębokość liścia
- ścieżki w drzewie

Zastosowania

- drzewa wyszukiwania
- B-drzewa
- kopce
- hierarchie

Struktury

Definicja

```
static class ListNode {
    int val;
    ListNode prev;
    ListNode next;
}
```

Tworzenie

```
lista = new ListNode();
```

Odwołanie do elementów

```
l.val=10;

if(l.next!=null) {
    doSomething();
}
```

Zadanie 1 – Wyszukiwanie binarne

Zadanie

Tablica A typu int[N] wypełniona posortowanymi rosnąco liczbami. Stwierdzić, czy wartość val występuje w tablicy. Jeżeli tak, zwrócić indeks. Jeżeli nie, zwrócić indeksy największej liczby mniejszej od val i najmniejszej liczby większej od val.

Zadanie 1 – Wyszukiwanie binarne

Zadanie

Tablica A typu int[N] wypełniona posortowanymi rosnąco liczbami. Stwierdzić, czy wartość val występuje w tablicy. Jeżeli tak, zwrócić indeks. Jeżeli nie, zwrócić indeksy największej liczby mniejszej od val i najmniejszej liczby większej od val.

Wskazówka

Da się to zrobić wykonując $O(\log(n))$ porównań.

Zadanie 2 – Sortowanie bąbelkowe przy użyciu listy

Algorytm

Polega na porównywaniu dwóch kolejnych elementów i zamianie ich kolejności, jeżeli zaburza ona porządek, w jakim się sortuje tablicę. Sortowanie kończy się, gdy podczas kolejnego przejścia nie dokonano żadnej zmiany.

Przykład

$$[\underbrace{4,2}_{4>2},5,1,7] \rightarrow [2,\underbrace{4,5}_{4<5},1,7] \rightarrow [2,4,\underbrace{5,1}_{5>1},7] \rightarrow [2,4,1,\underbrace{5,7}_{5<7}]$$

$$[\underbrace{2,4}_{2<4},1,5,7] \rightarrow [2,\underbrace{4,1}_{4>1},5,7] \rightarrow [2,1,\underbrace{4,5}_{4<5},7]$$

$$[\underbrace{2,1}_{2>1},4,5,7] \rightarrow [1,\underbrace{2,4}_{2<4},5,7]$$

$$[\underbrace{1,2}_{1<2},4,5,7]$$

Zadanie 3 – Sortowanie przez wstawianie przy użyciu listy

Algorytm

1. Utwórz zbiór elementów posortowanych i przenieś do niego dowolny element ze zbioru nieposortowanego.
2. Weź dowolny element ze zbioru nieposortowanego.
3. Wyciągnięty element porównuj z kolejnymi elementami zbioru posortowanego póki nie napotkasz elementu równego lub elementu większego (jeśli chcemy otrzymać ciąg niemalejący) lub nie znajdziemy się na początku/końcu zbioru uporządkowanego.
4. Wyciągnięty element wstaw w miejsce gdzie skończyłeś porównywać.
5. Jeśli zbiór elementów nieuporządkowanych jest niepusty wróć do punkt 2.

Zadanie 4 – binarne drzewo wyszukiwania

Zadanie

Tablica A typu int[N] wypełniona liczbami. Stwierdzić, czy wartość val występuje w tablicy. Jeżeli tak, zwrócić indeks. Jeżeli nie, zwrócić największą liczbę mniejszej od val i najmniejszą liczbę większej od val.

Zadanie 4 – binarne drzewo wyszukiwania

Zadanie

Tablica A typu int[N] wypełniona liczbami. Stwierdzić, czy wartość val występuje w tablicy. Jeżeli tak, zwrócić indeks. Jeżeli nie, zwrócić największą liczbę mniejszej od val i najmniejszą liczbę większej od val.

Wskazówka

Stworzyć drzewo binarnego wyszukiwania i wypełnić je liczbami z A.