Esercizi – Strutture Dati 2

Anno Accademico 2022/2023

Dott. Staccone Simone

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Esercizi Alberi

1. Codificare una funzione che riceve in input due alberi e restituisce 1 se i due alberi sono identici, 0 altrimenti.

2. Si scriva una funzione che prende in ingresso un albero binario e restituisce la somma di tutti i valori dei nodi dell'albero.

3. Si scriva una funzione che prende in ingresso un albero binario e restituisce 1 se è completo (cioè se tutti i nodi o sono foglie o hanno 2 figli e se tutte le foglie sono alla stessa profondità), 0 altrimenti.

4. Si scriva una funzione che prende in ingresso un albero binario bilanciato e lo trasforma in una lista ordinata. (Per semplicità supponi di avere una funzione addList(…) che appende in coda gli elementi ad una lista)

Esercizi Alberi

1. Codificare una funzione che riceve in input due alberi e restituisce 1 se i due alberi sono identici, 0 altrimenti.

```
fun f(tree t1,tree t2) : int
   if(t1 == NULL AND t2 == NULL)
        return 1;
   if(t1 == NULL || t2 == NULL)
        return 0;
   return (t1.value == t2.value AND f(t1.right,t2.right) AND f(t1.left,t2.left))
```

Esercizi Alberi

2. Si scriva una funzione che prende in ingresso un albero binario e restituisce la somma di tutti i valori dei nodi dell'albero.

```
fun count(tree t): int
   if(t == NULL)
       return 0
   else
       return t.value+count(t.left)+count(t.right)
```

Esercizi Alberi

3. Si scriva una funzione che prende in ingresso un albero binario e restituisce 1 se è completo (cioè se tutti i nodi o sono foglie o hanno 2 figli e se tutte le foglie sono alla stessa profondità), 0 altrimenti.

```
fun f(tree t) : int
   if (t == NULL)
      return 0;
   if (aux(t, 0) >= 0)
      return 1;
   else
      return 0;
```

Esercizi Alberi

3. Si scriva una funzione che prende in ingresso un albero binario e restituisce 1 se è completo (cioè se tutti i nodi o sono foglie o hanno 2 figli e se tutte le foglie sono alla stessa profondità), 0 altrimenti.

```
fun aux(tree t, counter) : int
   int fromLeft;
   int fromRight;
   if (t.left == NULL AND t.right == NULL)
      return counter;
   else if ((t.left == NULL AND t.right != NULL) ||(t.left != NULL AND t.right == NULL))
      return -1;
   else
      fromLeft = aux(t.left, counter + 1);
      fromRight = aux(t.right, counter + 1);
      if (fromLeft != fromRight) {
          return -1;
      else
          return fromLeft;
```

Quiz (Esame)

Data una collezione di elementi, identificati ciascuno da una chiave intera, voglio scegliere una struttura dati appropriata, che minimizzi il costo dell'esecuzione dei miei programmi, prendendo in considerazione l'operazione che svolgo più di frequente. Quale struttura dati sceglierete, nei seguenti casi?

- Inserimento ordinato di un elemento:
- Rappresentazione di un file system:
- Estrazione del valore minimo:
- Inserimento in testa:
- Estrazione dell'elemento k-esimo:
- Inserimento in coda:

Dopo l'inserimento in un albero Rosso-Nero:
a) possono essere necessarie no a 2 rotazioni.
b) possono essere necessarie anche più di 2 rotazioni.
c) basta al più una rotazione.
d) se lo zio è nero, basta ricolorare.

Quiz (Esame) - Soluzioni

Data una collezione di elementi, identificati ciascuno da una chiave intera, voglio scegliere una struttura dati appropriata, che minimizzi il costo dell'esecuzione dei miei programmi, prendendo in considerazione l'operazione che svolgo più di frequente. Quale struttura dati sceglierete, nei seguenti casi?

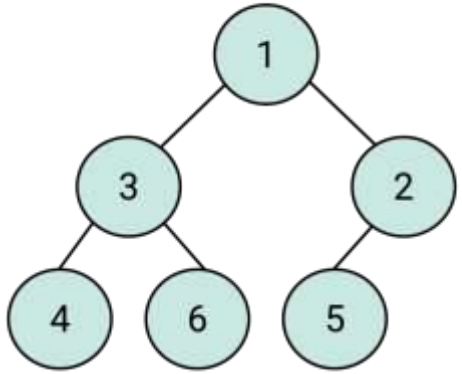
- Inserimento ordinato di un elemento: albero auto bilanciante
- Rappresentazione di un file system: albero n-ario
- Estrazione del valore minimo: heap
- Inserimento in testa: lista singolarmente collegata
- Estrazione dell'elemento k-esimo: array
- Inserimento in coda: lista testa-coda

Dopo l'inserimento in un albero Rosso-Nero:

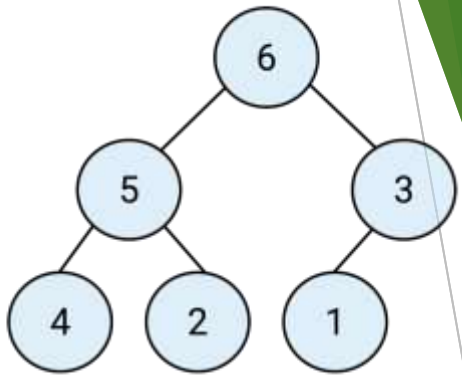
a) possono essere necessarie no a 2 rotazioni.

Heap

- Struttura ad albero rappresentata attraverso un array.
- La radice dell'albero deve essere l'element minimo (o massimo) di tutti gli altri nodi che compongono l'albero. Questa proprietà deve essere ricorsivamente vera per ogni sotto-albero analizzato.
- Dato il nodo padre nella posizione I dell'array, ogni nodo figlio si troverà nella posizione 2i (figlio sinistro) e 2i+1 (figlio destro). Inoltre dato un nodo per ottenere il padre abbiamo [i/2].
- Esistono min-Heap o max-Heap in base al criterio di ordinamento (crescente/decrescente)
- Approfondimento: https://www.geeksforgeeks.org/binary-heap/
- Heap Sort: https://www.geeksforgeeks.org/heap-sort/



Min heap



Max Heap