Laboratorio di Algoritmi e Strutture Dati I Docente: A. Murano

-- -- -- 25 Gennaio 2010 -- -- --

Laurea in Informatica Università degli Studi di Napoli "Federico II"

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4	Totale
/6	/6	/6	/14	/32

Non utilizzate altri fogli. Utilizzate soltanto lo spazio sottostante. Fogli differenti non saranno presi in considerazione per la correzione. Non scrivere a matita. Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri una coda di priorità per la gestione della coda di stampa di una rete implementata con una struttura dati heap **H[MAX]**. Si supponga di avere memorizzata la dimensione dell'heap in **heapsize** e di disporre delle seguenti funzioni:
   a. **void Heapify(int H[MAX], int el); \\ el** è un indice del vettore **H**
   b. **void BuildHeap(int H[MAX]);**
   c. **void HeapSort(int H[MAX]);**

   Si implementino:
   a. una funzione **int ricerca (int H[MAX], int el)** capace di restituisce l'indice del vettore in cui si trova l'elemento **el** e **-1** se l'elemento non è presente nel vettore
   b. una funzioni **void annulla_lavoro(int H[MAX], int el)**, che prende in input l'heap **H[MAX]** e il lavoro **el** da eliminare ed elimina in lavoro **el** dall'heap **H[MAX]** nel caso esso sia presente. Scrivere un breve paragrafo sull'idea di implementazione della funzione.

2. Si considerino una lista di numeri interi **L** implementate come lista doppiamente puntata e non circolare, utilizzando la seguente struttura

```
struct elemento {
        struct elemento *prev;
        int inf;
        struct elemento *next;}

typedef struct elemento Lista;
Lista *L;
```

Si implementi una funzione **ricorsiva** che presi in input **L** e una variabile puntata di interi **el_rimossi** (e niente altro) rimuova gli elementi negativi da **L** e incrementi di uno la variabile **el_rimossi**, per ogni elemento rimosso. Attenzione, la funzione deve essere definita in modo da riportare alla funzione chiamante **L** e **el_rimossi** opportunamente modificati.

3. Sia T un albero binario, implementato con la seguente struttura a puntatori:

```
struct nodo {
        int info1;
        struct nodo *left;
        struct nodo *right;}

struct nodo *T;
```

a. Verificato che si tratti di un albero binario di ricerca, aggiungere ad ogni nodo un valore pari alla sua posizione nella visita in ordine. Domanda: dopo l'operazione, l'albero rimane un ABR?
b. Creare una struttura dati **ternaria** che inserisca come valore di mezzo la somma della chiave sx e dx.

c. Siano **G** e **H** due grafi orientati pesati entrambi con pesi positivi, di **n** vertici 0, 1,…, n-1 e rappresentati con liste di adiacenza utilizzando la seguente struttura:

```
typedef struct graph {
        int   nv;
        edge   **adj; } graph;

        graph *G, *H;
```

```
typedef struct edge  {
        int   key;
        int   peso;
        struct  edge   *next; } edge;
```

a. Scrivere in linguaggio C una funzione che, presi in input i due grafi **G e H** costruisca un terzo grafo T (e lo restituisca) in cui un arco (a,b) è in T con peso 1 se non è presente ne in G e ne in H.
b. Scrivere una funzione in C che presi in input sia la lista **L** (come descritta nell'esercizio 2) e il grafo T (e niente altro), verifichi che esiste un percorso in T uguale a L nei due versi.