Laboratorio di Algoritmi e Strutture Dati I Docenti: Murano / Cutugno

-- -- -- 20 GIUGNO 2006 -- -- --

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

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	3	Totale
/10	/10	/10	/30

Non utilizzate altri fogli. Utilizzate soltanto lo spazio sottostante. Fogli differenti non saranno presi in considerazione per la correzione. Non scrivere a matita

Gli studenti che hanno frequentato e consegnato i progetti devono svolgere solamente gli esercizi 1, 2 e 3. Tutti gli altri devono anche svolgere anche l'esercizio 4 e l'esercizio 5.

1. Si considerino due Stack **S1 e S2**, implementati con array **S1[MAX] e S2[MAX]**. Supponendo che S1 e S2 abbiano entrambi n elementi e che n sia al più la metà di MAX/2, si implementi la funzione ricorsiva **void moltiplica Stack(int S1[MAX], S2[MAX])** che sostituisce ogni elemento di posizione i nello stack S1 il suo prodotto con l'elemento n-i. Attenzione, lo stack S2 deve risultare invariato, possibilmente senza utilizzare strutture dati di appoggio. Si ricorda che lo stack è una struttura dati che permette l'accesso ai suoi dati solo dal top. Scrivere tutte le funzioni utilizzate. Esempio: stack S1 = 4|3|5 -- stack S2 = 2|1|4. Stack S1 finale = 16|3|10|

2. Si considerino due liste di numeri interi **Lista1 e Lista2** implementate come lista doppiamente puntata e non circolare, utilizzando la seguente struttura

```
struct elemento {
        int inf;
        struct elemento *prec;
        struct elemento *succ;}

struct elemento *Lista1, Lista2;
```

Si implementi la funzione **void togli_somma** che prende in input le due liste (**Lista1** e **Lista2**) ed elimina dalla lista **Lista2** il numero m se la somma degli elementi contenuti in **Lista1** è proprio m, e ripete l'operazione invertendo le due liste.
Si discuta la complessità della funzione implementata.
Esempio, sia **Lista1** uguale a 2→0→-4→8→0→2 e Lista2 uguale 4→8→-4→8. Dopo la prima iterazione **Lista1** è uguale a 2→0→-4→8→0→2 e Lista2 è uguale 4→-4 (tolgo 8 da **Lista2**). Dopo la seconda iterazione, **Lista1** è uguale a 2→-4→8→2 (tolgo 0 da **Lista1**) e **Lista2** sarà uguale a 4→-4. A questo punto si termina perché la somma degli elementi in **Lista1** è 8, il quale non è presente in **Lista2**.

3. 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;
```

scrivere in linguaggio C una funzione che presi in input i due grafi **G e H**,

a. per ogni arco (a, b) presente sia in G che in H, rimuove l'arco con il peso maggiore (in caso di equivalenza di pesi ne rimuove uno a caso).
b. per ogni arco (a, b) non presente sia in G che in H, ne aggiunge in modo alternante uno a G e uno a H, con il seguente peso. Se (a,b) è l'arco i-esimo aggiunto, il suo peso è fattoriale di i.
c. Studiare la complessità della funzione implementata.

Gli studenti che non hanno consegnato il progetto devono risolvere i seguenti esercizi aggiuntivi:

Spazio riservato alla correzione

1	2	3	4	5	Totale
/5	/6	/6	/8	5	/30

4. Dato un albero binari di ricerca T implementato con la seguente struttura a puntatori:

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

struct nodo *T;
```

   implementare una funzione in linguaggio C ricorsiva che restituisca un Grafo G implementato con la struttura dati di cui al punto 3.
   Descrivere la complessità della funzione implementata

5. Dopo una breve descrizione sugli algoritmi di ordinamento, implementare un algoritmo di ordinamento a scelta e descriverne la complessità.