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

-- -- -- 13 FEBBRAIO 2006 -- -- --

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

Nome e Cognome

Spazio riservato alla correzione

1	2	3	Totale
/8	/12	/10	/30

Numero di Matricola

Non utilizzate altri fogli. Utilizzate soltanto lo spazio sottostante e il retro dei fogli. 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 gli esercizi 1, 2 e 3.
I non frequentanti devono svolgere anche gli esercizi 4 e 5.
Gli studenti che devono integrare mod. B devono svolgere gli esercizi 3 e 5

1. Si consideri una Coda **Q**, implementato con array **Q[MAX]**. Si implementi la funzione ricorsiva **void elimina dispari(int Q[MAX])** che elimina dalla coda tutti i numeri dispari lasciando invariato l'ordine degli elementi. Si ricorda che la Coda è una struttura dati che permette l'inserimento dei dai in coda e l'estrazione dei dati dalla testa. Implementare tutte le funzioni di libreria necessarie (EmptyQueue, EnQueue, DeQueue, ecc.) e la funzione **elimina dispari** indipendentemente dalla implementazione delle librerie.
   Esempio: Coda iniziale 1|3|3|6|1|2 -- Coda finale 6|2

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

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

struct elemento *Lista_1,*Lista_2;
```

   a. Si implementi la funzione ricorsiva **struct elemento *togli_neg(struct elemento *L)** che elimina dalla lista **L** gli elementi negativi senza cambiare l'ordine.
   b. Si implementi la funzione **ordina** che prende in input ***Lista1** e ***Lista2,** prima chiama **toglineg** e poi senza modificare ulteriormente Lista1 e Lista2 restituisce una nuova lista Lista3, ottenuta con gli elementi di Lista1 e Lista2, tale che Lista3 è ordinata in modo crescente e non ha elementi ripetuti.
      Esempio, sia **Lista1** uguale a 1→2→3→2, **Lista2** uguale a 1→2→1→2→4→2→5, dopo **togli_ripetizioni, Lista1** è uguale a 1→2→3, **Lista2** è uguale a 1→2→4→5, ordina restituisce 1→2→3→4→5

3. Siano **G** un grafo orientati pesato di **n** vertici 0,1,…, n-1 e rappresentato con liste di adiacenza utilizzando la seguente struttura:

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

        graph *G;
```

scrivere in linguaggio C una funzione che preso in input il grafo **G**, trasformi **G** in modo che ogni nodo abbia al più un solo arco uscente rappresentato da quello con peso maggiore, e restituisca in output un grafo H rappresentato con matrice di adiacenza. Descrivere la complessità della funzione implementata.

Gli studenti che non hanno frequentato il corso e non hanno consegnato i progetti devono risolvere anche i seguenti esercizi aggiuntivi:

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 che verifichi che T goda delle proprietà di albero binario di ricerca .

5. Dopo una breve descrizione sulla rappresentazione dei grafi pesati, descrivere la procedura per il calcolo del percorso minimo su di un grafo