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

– – – 30 GENNAIO 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	/12	/12	/34

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 solamente gli esercizi 1, 2 e 3. Gli altri devono svolgere anche gli esercizi 4 e 5.

1. Si consideri uno Stack **S**, implementato con array **S[MAX]**. Si implementi la funzione ricorsiva **void raddoppia Stack(int S[MAX])** che raddoppia le occorrenze di ogni elemento nello stack lasciando invariato l'ordine degli elementi. Si ricorda che lo stack è una struttura dati che permette l'accesso ai suoi dati solo dal top.
   Esempio: stack iniziale 1|3|3|6|2 -- stack finale 1|1|3|3|3|6|6|2|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 **void togli_pari_dispari(struct elemento *L, int a)** che elimina dalla lista **Lista1** tutte i numeri dispari (passando 1 al posto di a) e tutti i numeri pari dalla lista **Lista2 (**passando 0 al posto di a**)** senza usare strutture dati aggiuntive e senza cambiare l'ordine degli elementi.
   b. si implementi la funzione ricorsiva **interleaving** che prende in input ***Lista1** e ***Lista2** modificate e restituisce l'interleaving della 2 liste.
   Esempio, sia **Lista1** uguale a 1→2→3→2, **Lista2** uguale a 1→2→1→2→4→2→5, dopo **togli_pari_dispari, Lista1** è uguale a 2→2, **Lista2** è uguale a 1→1→5, interleaving restituisce 2→1→2→1→5

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 preso in input i due grafi **G e H**, restituisce **G** con i pesi dei suoi archi diminuiti dei pesi degli archi corrispondenti in H. Se un arco ottiene peso negativo, l'arco deve essere rimosso. 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:

Spazio riservato alla correzione

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

4. Dati due alberi binari di ricerca T1 e T2 implementati con la seguente struttura a puntatori:

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

struct nodo *T1,*T2;
```

   implementare una funzione in linguaggio C che rimuova dall'albero tutti i nodi con valore **inforadice** dispari. Poi si costruisca un nuovo albero **T** in cui **inforadice** è data dalla somma dei rispettivi valori **inforadice** nei nodi corrispondenti.
   Rispondere inoltre alla seguente domanda motivando la risposta: L'albero **T** ottenuto è ancora un albero binario di ricerca?

5. Dopo una breve descrizione sulla rappresentazione di un grafo tramite matrice di adiacenza e liste di adiacenza, descrivere due scenari in cui si evince il vantaggio di usare l'una o l'altra rappresentazione, in termini di complessità asintotica delle operazioni.

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

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.

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

-- -- -- 5 LUGLIO 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
/8	/10	/12	/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 code **C1 e C2**, implementate con array. Supponendo che C1 e C2 abbiano inizialmente entrambi n elementi, si implementi un algoritmo **ricorsivo** che prende in input **C1 e C2** come array di dimensione **MAX** (cioè, C1[MAX] e C2[MAX]) e restituisce una lista puntata definita come

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

struct elemento *Lista;
```

scegliendo solo le chiavi dispari da C1 e le chiavi pari da C2 e riordini le chiavi al volo durante l'inserimento.

Possibilmente, si operi in modo che alla fine dell'operazione le due code risultino invariate.

Si implementi l'algoritmo tenendo conto che le code sono strutture dati il cui **accesso a nodi interni è proibito**. In particolare, si richiede che l'accesso alle code nell'algoritmo avvenga **solo tramite funzioni indipendenti dalla implementazione** delle code stesse. Tutte **le funzioni devono essere opportunamente implementate**.

Valutare la complessità di tempo e di spazio dell'intero algoritmo.

2. Si consideri una lista di numeri interi **Lista1** implementate come lista doppiamente puntata e non circolare, utilizzando la seguente struttura

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

struct elemento *Lista;
```

a) Si implementi la funzione ricorsiva **void gira(struct elemento *L1)** che elimina dalla lista **Lista1** in successione tutti gli elementi eliminando alternativamente la chiave minore e quella maggiore ad ogni giro. La funzione deve terminare se la lista si svuota completamente o se in un qualsiasi momento risultano presenti 3 chiavi consecutive uguali. Dal momento che la lista è circolare il concetto di consecutività si estende agli estremi.

b) Si consideri la seguente variante alla funzione **gira():** si considerino due liste circolari L1 ed L2 definite come sopra, un processo attraversa L1, i passaggi dei puntatori sono monitorati con messaggi a schermo, ogni volta che in L1 viene identificata una chiave **int Start_Forward** data, il processo viene sospeso, la chiave Start_Forward viene incrementata di 1 e viene dato il via ad un processo identico su L2, allo stesso modo, ogni volta che in L2 viene identificata una chiave **int Start_Back** il processo viene sospeso, la chiave Start_back viene decrementata di 1 e si continua al rimbalzo. Discutere quali sono le possibili condizioni di terminazione dell'algoritmo, individuare il caso peggiore e discuterne la complessità.

3. Sia **G** un grafo non orientato pesato con pesi positivi, di **n** vertici 0, 1,…, n-1 e rappresentato con lista di adiacenza utilizzando la seguente struttura:

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

        graph *G;
```

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

scrivere in linguaggio C una funzione che preso in input il grafo **G**,

a. data una coppia di nodi (x,y), identifichi ogni possibile percorso (senza ripetizioni) di nodi che porta da x a y, ne indichi il costo totale di ogni percorso, e ne rimuova l'ultimo arco incontrato nel percorso più costoso;
b. dato il percorso meno costoso fra quelli precedentemente riportati, aggiunga un nuovo nodo w nel grafo e un nuovo arco fra il nodo y il nuovo nodo w.
c. Studiare la complessità della funzione implementata al punto a e b.

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/	5/	8/	7/	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, dato un intero **n** di valore massimo pari a quello della profondità dell'albero, restituisca l'elenco orizzontale di tutti i nodi di livello n indicando raggruppandoli per nonni.
   Descrivere la complessità della funzione implementata.

5. Si descriva un algoritmo di visita di grafo a piacere e se ne discuta la complessità.

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

-- -- -- 11 SETTEMBRE 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
/12	/4	/14	/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.

1. Si considerino due Stack **S1** e **S2**, implementati con array **S1[MAX] e S2[MAX],** e riempiti con interi **da 1 a 9**. Si implementi la funzione ricorsiva **void gioco, che indipendentemente dalla implementazione della struttura dati Stack,** prendendo in input i due Stack, effettui un gioco nel seguente modo:
   Ad ogni turno del gioco si considera **la somma modulo 10** dei valori al top dei due Stack. Se tale somma è minore di 5, vince il primo Stack, altrimenti vince il secondo. Ad ogni iterazione, si rimuove il top dallo Stack perdente. Perde lo Stack che finisce per primo i suoi valori. La funzione termina indicando lo Stack vincente, che ritorna come all'inizio del gioco, mentre quello perdente risulta vuoto. Si ricordi che lo Stack è una struttura dati che permette l'accesso solo al top. Scrivere tutte le funzioni utilizzate. Non utilizzare altre strutture dati.
   Esempio: Sia S1=|4|7|9| e S2 = |2|9|. Risposta: S2 vince.

2. Si considerino due liste di numeri interi **Lista1 e Lista2 "ordinate in modo crescente"** implementate come liste singolarmente puntate e non circolari, utilizzando la seguente struttura

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

struct elemento *Lista1,*Lista;
```

Si implementi la funzione ricorsiva **merge** che ricorsivamente prendendo in input le due liste, inserisce nella prima lista, mantenendo l'ordine crescente della lista, i valori pari della seconda, rimuovendoli.

Esempio, sia **Lista1** uguale a 1→10 e **Lista2** uguale a 2→4→7. Dopo l'esecuzione di merge, **Lista1** sarà uguale a **1→2→4→10,** e **Lista2** sarà uguale a 7.

3. Siano **G** e **H** due grafi non 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 tre funzioni che in successione, presi in input i due grafi **G e H**,

a. Sostituiscano tutti gli archi in G e H in modo che ogni arco (a, b) in G abbia peso a+b in G, e a*b in H
b. Per ogni arco (a, b) in G e H, rimuovano l'arco in G se la somma dei due pesi è dispari e da H in caso contrario.
c. Preso **m** come input, controllino se esiste un sottografo di G, di m nodi, fortemente connesso
d. Studiare la complessità delle funzioni implementate.

Gli studenti che non hanno consegnato il progetto devono risolvere il seguente esercizio aggiuntivo:

Spazio riservato alla correzione

1	2	3	4	Totale
/10	/4	/10	/6	/30

4. Dato un albero binario 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 rimuova tutti ogni nodo con valore negativo dall'albero T e il suo sottoalbero destro. Dunque, per ogni nodo rimosso, il suo sottoalbero sinistro diventa sottoalbero sinistro di suo padre.
Descrivere la complessità della funzione implementata.

5. Scrivere in linguaggio C la funzione **reverse** che preso in input un grafo orientato pesato cambi il verso di tutti i suoi archi e se ne discuta la complessità. Si utilizzi come struttura dati quella dell'esercizio precedente, opportunamente adattata.

Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

-- -- -- 18 DICembre 2008 -- -- --

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	/10	/12	/34

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 uno Stack **S**, implementato con array **S[MAX+1]**. Si implementi la funzione ricorsiva **moltiplica** che sostituisce ogni elemento di posizione **i** con il prodotto di tutti gli elementi che lo precedono nello stack (dal bottom all'elemento i). Si ricordi che lo stack è una struttura dati che permette l'accesso ai suoi dati solo dal top. Le funzioni di gestione stack possono essere omesse.

   Esempio: stack iniziale 4|3|1|2|5 (4 bottom dello stack) – stack finale 4|12|12|24|120|

2. Si considerino due liste di numeri interi **Lista1 e Lista2 "ordinate in modo strettamente crescente"** implementate come liste doppiamente puntate e non circolari, utilizzando la seguente struttura

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

struct elemento *Lista1,*Lista2;
```

Si implementi una funzione (**possibilmente ricorsiva**) che rimuova da Lista1 tutti i numeri presenti in Lista2, senza usare strutture dati aggiuntive e senza cambiare l'ordine degli elementi. Restituire solo Lista1.

Esempio, sia **Lista1** uguale a 1→2→4→7→10 e **Lista2** uguale a 2→3→7→11. Dopo l'esecuzione di remove, **Lista1** sarà uguale a **1→4→10**.

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

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

struct nodo *T;
```

   a. Dato un albero P implementato con la stessa struttura di T, implementare una funzione in linguaggio C che verifichi, senza usare strutture dati di appoggio, che T e P sono identici (struttura e valori);
   b. implementare una funzione in linguaggio C che verifichi (possibilmente senza strutture di appoggio), che per ogni elemento di T ci sia il suo predecessore o il suo successore.

4. 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,** restituisca un nuovo grafo P (della stessa struttura di G e H), che abbia

   i. gli stessi vertici di G e H
   ii. un arco (a, b) per ogni arco arco (a,b) presente sia in H che nella trasposta di G, e tale che la differenza tra i due pesi sia positiva. Tale differenza rappresenta poi il peso di (a,b) in P.

b. Implementare una funzione che calcoli, dato G e un suo nodo, il suo grado incidente.

Laboratorio di Algoritmi e Strutture Dati I Docente: Aniello Murano

-- -- -- 09 Febbraio 2009 -- -- --

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
/8	/6	/10	/10	/34

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 **Q**, implementata con array **Q[MAX+2], riempita con interi**. Si implementi la funzione ricorsiva **void moltiplica (int Q[])** che, utilizzando una libreria di funzioni di accesso alla coda (**da implementare**) prendendo in input Q, restituisce la coda con gli elementi nello stesso ordine, sostituendo però ogni tripla di numeri consecutivi (il “resto” è lasciato inalterato) con il loro prodotto, dando priorità agli elementi più vicini alla testa. Si ricordi che la coda è una struttura dati che permette l’accesso ai suoi dati solo dalla testa.
   Esempio: Coda iniziale 3|2|2|4|1|5|7|2 (3 testa della coda) – coda finale 12|20|7|2|

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

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

struct elemento *Lista1,*Lista2;
```

Si implementi una funzione che preso in input un intero **el** restituisca Lista1 e Lista2 rimuova da entrambe le Liste tutte le occorrenze di **el**. Si implementi la funzione in modo **ricorsivo.**

3. 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 dello heap in **heapsize.** Si implementi una funzione che controlli che H sia effettivamente un heap. Si implementi poi una funzione per la soluzione del seguente problema: sia k un numero intero. Si modifichi l'i-esimo elemento dello heap aggiungendogli k, ovvero A[i] viene sostituito con A[i] + k, e di ripristinare lo heap.

4. Si considerino due grafi **G** e **H** grafi orientati pesati di **n** vertici 0,1,…, n-1 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 grafi **G** e **H**, restituisca un nuovo grafo T con la stessa struttura in modo tale T ha l'arco (a,b) se esso è presente sia in G che in H e il suo peso è dato dalla somma del relativo peso in G e H più il grado adiacente di a più il grado incidente di b in H.

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

-- -- -- 22 Dicembre 2009 -- -- --

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	/8	/6	/14	/34

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 considerino due stack **S1 e S2**, implementati con array **S1[MAX+1] e S2[MAX+1], entrambi riempiti con interi**. Si implementi la funzione ricorsiva **void gioca (int S1[], int S2[])** che, utilizzando una libreria di funzioni di accesso agli stack (**da implementare**) prendendo in input i due stack, simuli il seguente gioco.
   a. Se i due stack non hanno lo stesso numero di elementi, vince lo stack con più elementi.
   b. Se invece i due stack hanno uguale numero di elementi, partendo dalla base, si confrontano ad ogni turno i numeri allo stesso livello. Se la loro somma è dispari, si elimina l'elemento di S1, se, invece è pari si elimina l'elemento di S2.
   Esempio: S1 iniziale |3|2|4|10| (3 bottom dello stack) – S2 iniziale |6|2|18| (6 bottom dello stack). Vince S1. Se invece S2 è |6|2|18|4|, allora S1 finale |2|4|10|– S2 finale |6|

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

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

struct elemento *Lista1,*Lista2;
```

   a. Si implementi una sola funzione **ricorsiva** (che eventualmente può richiamare sottofunzioni) che rimuova sia da Lista1 che da Lista2 tutte le occorrenze di numeri negativi.
   b. Dato un valore x, scrivere una funzione ricorsiva che, verificata l'esistenza di x in Lista1, inserisca Lista2 prima di x (in Lista1) e restituisca Lista1 così modificata. Se x non esiste in Lista1, accodare Lista2 a Lista1.

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

```
struct nodo {
        int info1;
        int info2;
        struct nodo *left;
        struct nodo *right;}

struct nodo *T;
```

   a. scrivere una funzione che verifichi che T è un albero binario di ricerca secondo la chiave (info1+ info2) per ogni nodo.
   b. Assumendo che T sia un ABR secondo la proprietà a), data una coppia di elementi (x1,x2) verificare se esiste un elemento in T che ha tale coppia usando una ricerca binaria.

4. 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 l'arco (a,b) è presente se è presente sia in G che in H, oppure in nessuno dei due. Nel primo caso prenderà come peso la somma dei pesi dei relativi archi in G e H. Nel secondo caso, prenderà peso -1
   b. Scrivere una funzione in C che verifichi che per ogni nodo di G e H, il grado incidente in G è uguale al grado adiacente in H.

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

-- -- -- 21 Giugno 2012 -- -- --

Laurea in Informatica Università degli Studi di Napoli “Federico II”

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	3	3	Totale
/6	/8	8	/10	32

Non utilizzate altri fogli. Utilizzate soltanto lo spazio sottostante. Fogli differenti non saranno presi in considerazione per la correzione. Non scrivere a matita. **Discutere della complessità per tutti gli esercizi**

1. Si considerino una coda Q, implementati con array Q**[MAX+2]**, come vista a lezione.
   Si implementi la funzione ricorsiva **void rimuovi(int Q[])** che presa la coda rimuova gli elementi dispari in posizione dispari e gli elementi pari se sono in posizioni pari e maggiori dell'elemento che segue nella coda (prima della rimozione). Attenzione, l'ordine degli elementi nella coda deve risultare invariato, non si devono usare strutture dati di appoggio e la funzione principale non ha altri parametri oltre la coda. Si ricorda che la coda è una struttura dati che permette l’accesso ai suoi dati solo dalla testa. Scrivere tutte le funzioni utilizzate.
   Esempio: Coda Q = 7|1|4|3|2|5| con 7 in testa. Coda finale Q = |7|2|. Le posizioni si contano dalla fine. Dunque, 5 è in posizione uno (dispari), 2 in posizione due (pari), ecc.
   Nota che 2 non si cancella anche se è in posizione pari perchè è preceduto da 5 che è più grande.

2. Si considerino due liste di numeri interi positivi **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;}
```

Si implementi la funzione **void togli_numero** che prende in input soltanto le due liste (**Lista1** e **Lista2**), calcola il numero degli elementi in **Lista1** (diciamo m) e se tale numero (m) è presente in **Lista2** lo rimuove da questa lista (in tutte le sue occorrenze). Poi, ripete l'operazione invertendo le due liste e così via. L'algoritmo termina non appena il numero m calcolato non è presente nella lista in cui deve essere rimosso o una delle due liste diventa vuota.
Si discuta la complessità della funzione implementata.
Esempio, sia **Lista1** uguale a 3→1→4→8→8 e **Lista2** uguale 8→5→4→8. Dopo la prima iterazione **Lista2** è uguale a 8→4→8 (rimosso 5) e alla seconda iterazione **Lista1** è uguale 1→4→8→8 (rimosso 3). Alla prossima iterazione, **Lista2** è 8→8 (rimosso 4). A questo punto l'algoritmo termina perchè gli elementi in **Lista2** sono 2 e tale numero non è presente in **Lista1.**

3. Sia T un Albero bianario definito secondo la seguente struttura

```
struct nodo {
        int inforadice;
        struct nodo *sx;
        struct nodo *dx;}

struct nodo *T;
```

a. Si implementi la funzione **check_ABR** che preso solo T verifichi che si tratta di un ABR.
b. Si implementi una funzione che preso T crei un albero ternario TER (struttura da definire) in cui sono presenti tutti i nodi di T e in aggiunta si inserisce in TER un figlio centrale foglia ad ogni nodo che ha entrambi i figli in T e il suo valore sia l'intero media dei valori dei due figli **.**

4. 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. crei un nuovo grafo T in modo che un arco (a, b) sarà presente in T se l'arco è presente in G e in H e i loro rispettivi pesi sono entrambi pari o entrambi dispari. Il peso associato all'arco inserito sarà la somma dei pesi dei due archi nei grafi corrispondenti.
b. Calcolare il grado incidente e adiacente di G e H.

Laboratorio di Algoritmi e Strutture Dati I — Docente: Murano

-- -- -- 09 Luglio 2014 -- -- --

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

Nome e Cognome — Numero di Matricola:

Spazio riservato alla correzione

1	2	2 ridotto	Totale
/12	/20	/12	/32

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate.

1. Si considerino uno stack S e una coda Q, rispettivamente implementati con array **S[MAX+1]** e **Q[MAX+2], entrambi riempiti con interi.** Si implementi una funzione che senza fare uso di costrutti iterativi, né di strutture di appoggio, ed utilizzando solo librerie di funzioni di accesso alle strutture, restituisca S e Q così modificati.
   - I valori dispari di S sono **copiati** in testa a Q (in ordine di lettura da S).
   - I valori pari di Q sono **rimossi** e inseriti sotto quelli di S. Se la somma dei valori di Q è inferiore a quella dei valori di S, l'ordine di iserimento in S è uguale all'ordine di precedenza in Q, altrimenti è invertito.

   Si creino stack e coda da tastiera e si stampino le strutture modificate.
   Esempio: S= 3|6|5 (5 top) e Q= 4|9|2|11 (4 testa) –> S = 4|2|3|6|5 e Q= 5|3|9|11

2. Sia **G** un grafo orientato e pesato con nodi 0,1,2,3,4,5,6,7 dato in input (anche random). Sia L una lista doppiamente puntata e non circolare **data in input da tastiera (anche vuota)** i cui nodi sono formati da tre valori interi (n,p,m).
   Eseguire le seguenti operazioni:
   a) Se la lista in input non è vuota, si rimuovano dalla lista le triple (n,p,m), tali per cui non esiste in G l'arco (n,p,m)
   b) Trasformare G in un albero minimo di copertura T1 a partire da una sorgente data.
   c) Trasformare T1 in un albero T2 i cui nodi sono etichettati con i pesi degli archi e la sorgente con la media di tutti i pesi inseriti in T2.
   d) Trasformare T2 in T3 secondo la seguente regola. Per ogni nodo si usi come sinistro il figlio con peso minimo e come destro quello con peso massimo.
   e) Si controlli se T3 è un ABR.

   In alternativa ai punti b), c), d), ed e) si può fare il seguente punto. Il voto del secondo esercizio di a+f vale 12 punti.
   f) **Rimuovere dal grafo [illegible] (n,p,m) presenti nella lista con m=n+1.**

archi

Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

-- -- -- 18 SETTEMBRE 2014 -- -- --

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

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	2 ridotto	Totale
/12	/12	/12	/36

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate.

1. Si implementino le funzioni necessarie a rappresentare un ABR con struttura puntata con link al padre (oltre che ai figli). Le funzioni devono permettere da menu:
   - la stampa in ordine,
   - la ricerca di un elemento x (con input da tastiera),
   - la rimozione della traccia da x alla radice con costruzione di una lista doppiamente puntata (nell'ordine da x alla radice)

   E' importante che gli elementi si possano inserire da tastiera e che sia possibile fare la stampa **in ordine** dell'albero prima e dopo le modifiche e la stampa della lista creata.

2. Si consideri una **lista di liste doppiamente puntate** non circolari. Si definisca la struttura dati e si implementi una funzione tale per cui nella lista i-esima (la prima ha indice 1) ci siano solo elementi dispari se i è dispari e pari altrimenti. La funzione dovrà spostare in avanti (inserimento in posizione arbitrario) gli elementi pari non posizionati correttamente e all'indietro gli elementi dispari. Usare la prima o la seconda lista (a seconda della parità) come successore dell'ultima lista.

Esempio se L in input è 1->8->5 | 2->3-> 6| 4->7->25 , in output avremo 3->1->5 |4->8->2->6
Importante, inserimento da tastiera con numero di liste arbitrario.

3. Sia **G** un grafo orientato e pesato con nodi 0, 1, … n-1. Si calcoli il grado entrante e uscente. Inoltre dato in input da tastiera un nodo x,

a) rimuovere fisicamente il nodo se la somma dei suoi archi entranti è maggiore di quelli uscenti.
b) per ogni arco **(a,x)** con peso **p1** e **(x,b)** con peso **p2**, si consideri l'arco **(a,b)** con peso **p=p1+p2**. Se tale arco **(a,b)** non è presente in **G** oppure è presente ma con un peso inferiore al valore **p**, si aggiunga in **G** l'arco **(a,b)** con peso **p**.

Si stampi il grafo prima e dopo le modifiche

Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

-- -- -- 15 Giugno 2015 -- -- --

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	/14	/34

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si considerino due stack **Pari e Dispari**, implementatati con array**, riempiti con interi**. Si implementi la funzione ricorsiva **void gioca** che, utilizzando una libreria di funzioni proceda a turni nel modo seguente. Ad ogni turno calcola la somma di Pari+Dispari se la somma è pari si toglie da Pari il numero minimo, altrimenti si toglie da Dispari. Il gioco finisce quando uno degli stack diventa vuoto. Si ricordi che la coda è una struttura dati che permette l’accesso ai suoi dati solo dalla testa. Si implementi una funzione che permetta di stampare ogni round.

2. Presi due ABR, si costruisca un albero binario che sommi i valori posti nelle stesse posizioni (se esistenti). Inoltre trasformare l'albero ottenuto in ternario in modo che il terzo nodo, aggiunto senza figli, abbia il valore medio dei fratelli esistenti nell'albero binario.

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 **G e H** e una lista doppiamente puntata e non circolare, **con input da tastiera sia per i grafi che per la lista (!!!)**, permetta di fare l'unione in un terzo grafo T, tale per cui l'arco (a,b) con peso p è inserito in T se (a,b) con peso positivo è presente sia in G e H, p è la somma dei relativi pesi ed inoltre il valore p è presente nella lista da dove deve essere rimosso. I grafi G ed H non devono essere modificati, la lista invece si può modificare. Grafo T e lista modificata vanno stampati alla fine delle operazioni. Si noti che esiste un solo valore p nella lista, allora due archi con peso p non possono essere inseriti in T.

Laboratorio di Algoritmi e Strutture Dati I — Docente: Murano

-- -- -- 12 Giugno 2019 -- -- --

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

Nome e Cognome: Numero di Matricola:

Spazio riservato alla correzione

1	2	3	Totale
/8	/8	/14	/30

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri una coda Q, implementata con array **Q[MAX+2]**, riempita con interi. Si implementi la funzione ricorsiva **void toglinegativi-positivi(int Q[])** che, utilizzando una libreria di funzioni di accesso alla coda, prendendo in input Q, restituisce la coda con gli elementi nello stesso ordine, rimuovendo, secondo la scelta, tutti i numeri negativi o positivi. Stampare a video la coda prima e dopo la modifica.

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

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

struct elemento *Lista1,*Lista2;
```

Si implementi una funzione ricorsiva che presi in input Lista1 e Lista2
   a. rimuova da Lista1 i negativi e si inseriscano in testa alla Lista2
   b. rimuova da Lista2 i positivi e li inserisca in testa alla lista1.
   c. Restituisca le due liste modificate.

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;
```

   a. scrivere in linguaggio C una funzione che, permetta di calcolare il grado adiacente e incidente di G e H.
   b. scrivere in linguaggio C una funzione che, presi in input G e H permetta di creare un terzo grafo T con la seguente regola: In T ci saranno archi presenti in G e H con peso dato dalla somma dei pesi corrispondenti in G e H, che sia pari e superiore a 10.

Laboratorio di Algoritmi e Strutture Dati I — Docente: Murano

-- -- -- 03 Settembre 2019 -- -- --

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	/6	/14	/30

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri uno stack implementato con array S[MAX+1], riempito con interi. Si implementi la funzione ricorsiva void minmax(int S[]) che, utilizzando una libreria di funzioni di accesso allo stack, restituisce S con gli elementi riordinati nel seguente modo. Partendo dalla base ogni terna avrà il mediano eliminato e il minimo sotto il massimo o viceversa a seconda che sia una terna dispari o pari. Se l'ultima terna ha meno di tre elementi, non sarà eliminato nessuno valore, ma solo eseguito l'ordinamento. Si crei lo stack da tastiera e si stampi lo stack modificato.
   Esempio: stack iniziale 3|7|5|4|9|8|2|6 (3 in base) – risultato 3|7|9|4|2|6

2. Si considerino due alberi T1 e T2. Dopo aver verificato che T1 e T2 sono alberi binari di ricerca, verificare T1 è un sottoalbero di T2.

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;
```

   a. scrivere in linguaggio C una funzione che, permetta di fare la differenza tra G e H. Archi negativi vanno rimossi. Nodi con peso totale degli archi uscenti meno di 10 vanno rimossi (10).
   b. Scrivere la trasposta del grafo ottenuto e restituirla sia con rappresentazione a lista che a matrice (4).

Tutti gli esercizi devono prevedere una interfaccia per la gestione dei dati: riempimento manuale/random, modifica del contenuto della struttura dati, esecuzione dell'esercizio, stampa prima e dopo l'esecuzione.

Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

---9 GENNAIO 2020---

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

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	2 ridotto	Totale
/12	/12	/12	/36

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate.

1. Siano Q0 e Q1 due code statiche riempite con interi positivi e S1 e S2 due stack riempiti il primo con valori Booleani e il secondo con interi positivi. Si implementi una funzione che dopo aver controllato che Q0, Q1, S1 e S2 hanno lo stesso numero di elementi e riempiti correttamente, rimuova ricorsivamente da Qi l'elemento in posizone j, dove i è dato come top di S1 e j come top di S2 (valori i e j vanno rimossi ad ogni chiamata ricorsiva). Ad ogni chiamata ricorsiva controllare anche che non si accede fuori dalle code né che un elemento sia stato già eliminato

Esempio: Q0= 1,2,3,4,5 – Q1=11,12,13,14,15 – S1=0,1,0,1,0 S2=1,2,1,6,4 Allora Q1=2,3,5 – Q2=11,13,14,15. Importante non cambiare l'ordine delle code.

E' importante che le strutture si possano riempire da tastiera, che si utilizzino solo funzioni di libreria per l'accesso alle code ed agli stack e che sia possibile visualizzare i contenuti delle strutture di dati prima e dopo le operazioni.

2. Si consideri quattro liste doppiamente puntate non circolari L1, L2, L3, L4. A turni rimuovere il minimo dalle 4 liste, fino a che una delle liste non diventi vuota. Ad ogni turno c'è una priorità che avanza di 1. All'inizio è 1>2>3>4, al secondo turno è 2>3>4>1. A parità di minimi pede (si toglie l'elemento dalla lista con priorità minore.

E' importante che le liste si possano riempire da tastiera e che sia possibile visualizzare i contenuti delle liste date prima e dopo le operazioni. Facoltativo: Si può usare lo stack S del primo esercizio invece di L2, se si preferisce.

3. Siano G1, G2 e G3 tre grafi orientati e pesati, entrambi di n nodi. Si controlli che G3 è l'unione (con somma di pesi) di G1 e G.

Si crei una interfaccia che sia in grado di creare i grafi G1, G2 e G3, aggiungere e togliere archi, modificare i pesi dei singoli archi, stampare i grafi in ogni momento, avviare la funzione dell'esercizio.

Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

--- FEBBRAIO 2020 ---

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

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri una coda Q con implementazione statica e due Liste, **L0** e **L1**, dinamiche doppiamente puntate, non circolari. La coda è riempita con valori booleani mentre le liste con valori interi. Scrivere una procedure che, facendo uso solo di costrutti ricorsivi, dapprima rimuova i numeri modulo 2 da L0 e modulo 3 da L1, e poi crei una lista L2 ottenuta scorrendo Q utilizzando la seguente regola: con 0 si prende l'elemento da L0, con 1 si prende l'elemento da L1. La coda deve rimanere inalterata. Scrivere un procedura che permetta si costruire Q, L0 e L1 da tastiera. Permetta di stampare Q, L0 e L1 dati in input, di stamparli dopo la rimozione degli elementi ed infine di stampare la lista L2 a fine esercizio.

2. Si consideri un heap riempito con interi (valori da tastiera). Scrivere una procedura per la ricerca ed eliminazione di un valore.

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. Scrivere una procedura per il calcolo del grafo differenza T. Il grafo T avrà l'arco (a,b) con peso p se l'arco è presente in G con peso p, ma non in H, oppure è presente in entrambi i grafi ma la differenza dei pesi p è positiva.

Laboratorio di Algoritmi e Strutture Dati I — Docente: Murano

--- MARZO 2020 ---

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	/8	/12	/34

Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri una coda Q con implementazione statica e due Liste, **L0** e **L1**, dinamiche doppiamente puntate, non circolari. La coda è riempita con valori booleani mentre le liste con valori interi. Scrivere una procedure che, facendo uso solo di costrutti ricorsivi, dapprima rimuova i numeri modulo 2 da L0 e modulo 3 da L1, e poi operi su L0 e L1 nel seguente modo: con 0 si prende l'elemento in testa a L0 e si sposta in coda a L1, con 1 oò viceversa, si prende l'elemento in testa a L1 e si sposta in coda a L0. La coda deve rimanere inalterata. Scrivere un procedura che permetta si costruire Q, L0 e L1 da tastiera. Permetta di stampare Q, L0 e L1 dati in input e di stamparli dopo l'esecuzione della regola di cui sopra.

2. Si consideri un ABR. Scrivere una procedura per la creazione e la stampa di un albero "specchio".

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. Scrivere una procedura per il calcolo del grafo differenza T. Il grafo T avrà l'arco (a,b) con peso p se l'arco è presente in G con peso p, ma non in H, oppure è presente in entrambi i grafi ma la differenza dei pesi p è positiva pari ed è unica (la seconda volta che appare un arco con peso p già presente, il nuovo arco non si aggiunge.
Scrivere una funzione che permetta di creare il grafo parametrizzato sul numero n, che permetta di aggiungere e togliere archi da G e H, permetta di stampare i grafi in ogni momento, permetta di eseguire la creazione di T e stampare tutti i grafi dopo tale operazione.

Traccia LASD-Giugno 2022

Aniello Murano, Silvia Stranieri

Giugno 2022

1 Esercizio 1: 10 punti

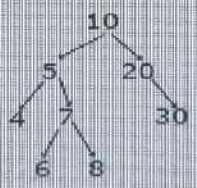
Scrivere una funzione che, data una stringa s, restituisca una lista L doppiamente puntata contenente tutte le vocali di s in ordine inverso a come compaiono in s. Inoltre, si scriva una funzione ricorsiva che elimini da L le ripetizioni consecutive.

Esempio:

$s =$ "*esempio*"
output $=$ **L** $=$ **e** $\leftrightarrow$ **i** $\leftrightarrow$ **e** $\leftrightarrow$ **NULL**

2 Esercizio 2: 10 punti

Scrivere una funzione che, dato un ABR T e un intero x, verifichi che T sia un ABR e restituisca la chiave minima del sottoalbero radicato in x.



Esempio:

$x = 7$
output $=$ "**si**", **6**

3 Esercizion 3: 12 punti

Scrivere una funzione che, dati due grafi G e H, verifichi se H è un sottografo di G. Si ricorda che $H = \langle V', E' \rangle$ si dice sottografo di $G = \langle V, E \rangle$ se $V' \subseteq V$ e $\forall (u, v) \in E', (u, v) \in E$.

LASD-settembre

Aniello Murano, Silvia Stranieri

Settembre 2022

1 Esercizio 1 (10 punti)

Data una lista doppiamente puntata e circolare, scrivere una funzione che rimpiazzi ogni elemento in posizione dispari con la media tra gli elementi alla sua destra e alla sua sinistra. Per la numerazione delle posizioni, si assuma che il primo elemento si trovi in posizione 1.

2 Esercizio 2 (10 punti)

Dato un albero e una chiave k, scrivere una funzione che verifichi che l'albero sia binario di ricerca e restituisca il valore minimo del sottoalbero radicato in k.

3 Esercizio 3 (10 punti)

Dato un grafo orientato pesato rappresentato con liste di adiacenza, scrivere una funzione che rimuova tutti gli archi con peso pari e tutti i nodi con grado uscente dispari.