Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Prova di laboratorio - Appello del 3 luglio 2019

Indice

Note importanti

- Si legga attentamente il testo degli esercizi e le indicazioni su come svolgerli. Se ci sono dubbi sul significato delle richieste, è opportuno chiedere chiarimenti!
- I file utili allo svolgimento degli esercizi, e indicati nel testo, sono contenuti nell'archivio zip, nella cartella `allegati`.
- Gli esempi di input/output proposti nel testo sono anch'essi contenuti nell'archivio zip, in file di testo separati, nella cartella `esempi`.
- Si leggano attentamente anche le indicazioni su come preparare le risposte. Per ogni esercizio è richiesto di preparare un file: in alcuni casi si tratta di un file di testo, in altri casi di un programma in C. Per ogni esercizio viene indicato il nome con cui salvare il file; è importante rispettare questa indicazione.
- Nella prima riga di tutti i file consegnati è necessario **scrivere nome, cognome e matricola**.
- Dopo essersi autenticati, si carichino sul sito `upload.di.unimi.it` i file contenenti le risposte. I nomi dei file devono essere i seguenti:
  `es1-valoreMancante.c`, `es2-radiceFoglie.c`, `es3-tubo.c`, `es4-reparti.txt`, `es5-reparti.c`.

1 Trova il valore mancante

Si consideri un array `A`, non vuoto, di n elementi, contenente tutti gli interi appartenenti all'insieme $\{0,1,2,\ldots,n\}$, tranne uno. L'array `A` è ordinato in senso crescente.

Scrivere una funzione chiamata `valoreMancante` che, dato l'array `A` e la sua lunghezza `n`, determini il valore mancante in `A`. Ad esempio, se `A` è il vettore $[0,1,2,3,4,6,7]$, la funzione deve restituire il valore 5.

Oltre alla correttezza della funzione verrà valutata anche l'efficienza dell'algoritmo usato. Una soluzione che impiega tempo lineare è poco interessante.

Note per la consegna. Si inserisca la funzione `valoreMancante` all'interno del file allegato di nome `es1-valoreMancante.c` e si completi opportunamente la sua invocazione all'interno del main. Si noti che il programma legge da standard input un intero n seguito da una sequenza di n interi ordinati (tutti gli interi appartenenti all'insieme $\{0,1,2,\ldots,n\}$, tranne uno).

Esempio di esecuzione 1

Ricevendo da standard input

```
7
0 1 2 3 4 6 7
```

il programma deve stampare

```
5
```

Esempio di esecuzione 2

Ricevendo da standard input

```
7
0 1 2 3 4 5 6
```

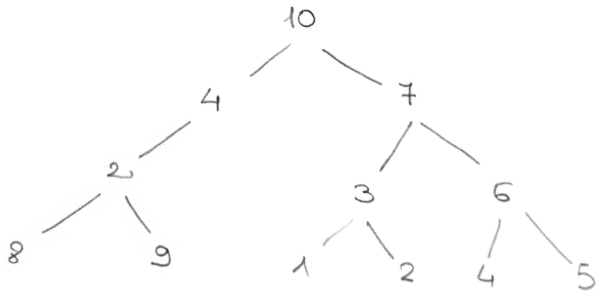
il programma deve stampare

```
7
```

2 Dalla radice alle foglie

Si consideri un albero binario in cui a ciascun nodo è associato un numero intero. Dato un qualsiasi cammino che porta dalla radice ad una foglia, definiamo il *costo del cammino* come la somma dei valori associati a tutti i nodi attraversati (inclusa la radice e la foglia di destinazione).

Ad esempio si consideri l'albero in figura.



Il cammino radice-foglia più a destra, che attraversa i nodi con associati i numeri 10, 7, 6, 5, ha costo 28.

Il programma allegato `es2-albero.c` contiene la definizione del tipo `Bit_node` e alcune funzioni per la gestione di alberi binari con chiavi intere. Esaminate il programma per capire cosa fanno le varie funzioni, quindi completate il programma scrivendo una funzione ricorsiva `costoMax` che, dato un albero, restituisca il costo massimo tra tutti i cammini radice-foglia nell'albero. Nel caso di albero vuoto, la funzione deve restituire 0.

Se lo si ritiene opportuno si può modificare anche il resto del programma e/o aggiungere altre funzioni ausiliarie. **La funzione `main` non deve però essere modificata.**

Il programma dovrà leggere da standard input un intero n seguito da una sequenza di n interi che rappresentano le n chiavi di un albero, quindi stampare l'albero in forma di sommario e il costo massimo tra tutti i cammini radice-foglia.

Esempio di esecuzione

Ricevendo da standard input

```
15
10 4 7 2 0 3 6 8 9 0 0 1 2 4 5
```

il programma deve stampare

```
*10
  *4
    *2
      *8
      *9
    *
  *7
    *3
      *1
      *2
    *6
      *4
      *5
28
```

infatti l'input rappresenta l'albero in figura e il cammino radice-foglia di peso massimo è quello a destra che attraversa i nodi di peso 10, 7, 6, 5.

Note per la consegna. Consegnate un file chiamato `es2-radiceFoglie.c` contenenente la funzione `costoMax`; il file deve contenere anche la funzione `main` già inclusa nell'allegato `es2-albero.c` (ricordo che la funzione `main` non deve essere modificata!).

3 Tubo metallico

Disponiamo di un tubo metallico di lunghezza L. Da questo tubo possono essere ricavati segmenti più corti, segando il tubo a partire da un'estremità. Abbiamo bisogno di n segmenti di tubo di lunghezza rispettivamente $\ell_1, \ell_2, \ldots \ell_n$, ma in generale la somma delle lunghezze dei segmenti supera la lunghezza L del tubo di partenza, quindi non è possibile ricavare tutti i n segmenti voluti.

Ad esempio, se si ha un tubo di lunghezza 24 e servono 6 segmenti di lunghezza rispettivamente 7 5 2 6 2 3, si possono ottenere al massimo 5 segmenti (ad esempio quelli di lunghezza 7 5 2 6 2). Se invece si ha un tubo di lunghezza 10 e servono 7 segmenti di lunghezza rispettivamente 7 4 3 5 1 2 6 si possono ottenere al massimo 4 segmenti (quelli di lunghezza 4 3 1 2).

Si scriva un algoritmo greedy che, dati gli interi $L, n, \ell_1, \ell_2, \ldots \ell_n$, determini il numero massimo di segmenti che è possibile ottenere.

Si implementi l'algoritmo in un programma che

- legga da standard input un intero L e un intero n, seguiti da una sequenza di interi $\ell_1, \ell_2, \ldots \ell_n$
- stampi il numero massimo di segmenti che è possibile ottenere a partire da un tubo di lunghezza L.

Oltre alla correttezza della funzione verrà valutata anche l'efficienza dell'algoritmo usato. Una soluzione che esamina tutte le possibili combinazioni è poco interessante.

Note per la consegna. Si scriva la risposta in un file di testo con nome `es3-tubo.c`.

Esempio di esecuzione 1

Ricevendo da standard input

```
24 6
7 5 2 6 2 3
```

il programma deve stampare

```
5
```

Esempio di esecuzione 2

Ricevendo da standard input

```
10 7
7 4 3 5 1 2 6
```

il programma deve stampare

```
4
```

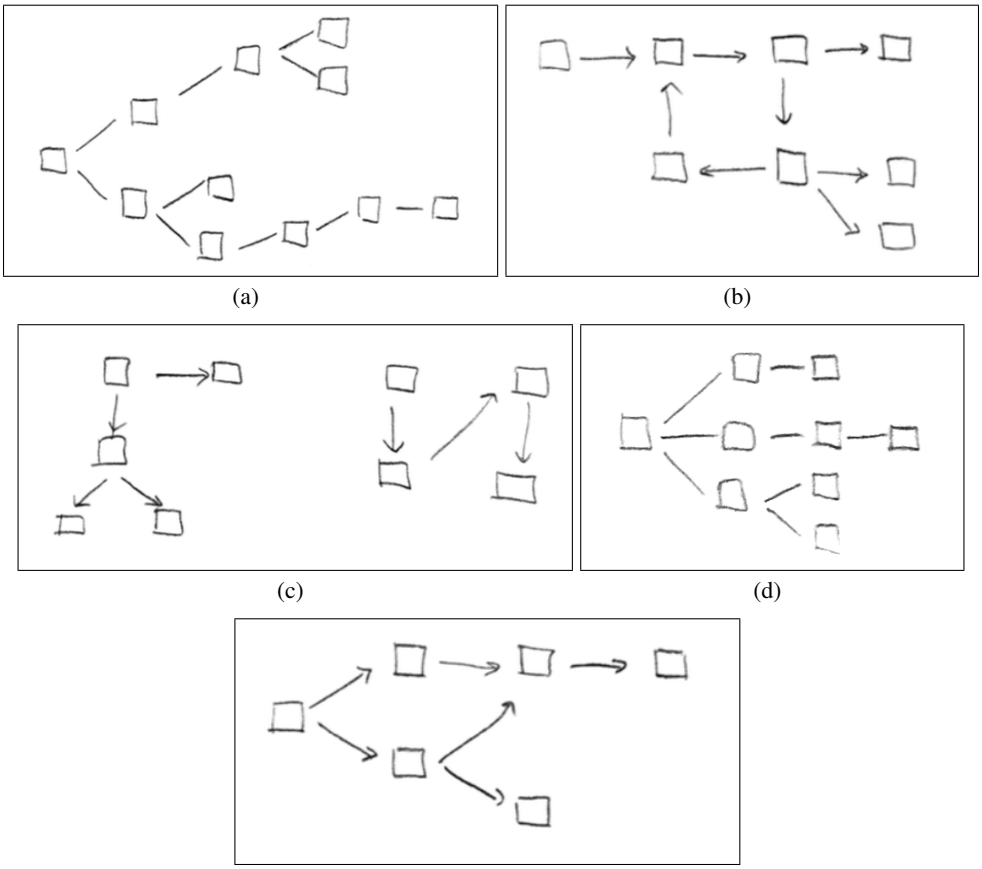
4 Reparti di produzione e collaudo

Un impianto di produzione produce diverse tipologie di prodotti. Il processo di produzione prevede la trasformazione dei prodotti in fasi successive, all'interno di reparti specializzati. I reparti sono individuati da un numero intero progressivo. Ogni reparto impiega un certo numero di lavoratori. Ci sono due tipi di reparto:

- Reparti di collaudo: sono dedicati al collaudo dei prodotti; conclusa la fase di collaudo, i prodotti escono dall'impianto perché il ciclo di produzione è concluso.
- Reparti di produzione: conclusa la fase di produzione in uno di questi reparti, il prodotto viene inviato ad un altro reparto (di produzione o collaudo). Da ogn reparto ci sono un massimo di 2 reparti possibili cui mandare il prodotto.

Ogni reparto riceve i prodotti da un solo altro reparto. La prima fase di produzione avviene sempre nello stesso reparto di produzione, chiamato Primo Reparto.

I processi produttivi dell'impianto sono descritti da uno schema che riporta, per ogni numero di reparto (di produzione o collaudo): il numero di lavoratori impiegati e il numero di reparto (di produzione) da cui provengono i prodotti.



(a) (b) (c) (d) (e)

1. Si considerino i diagrammi in figura. Se ogni quadratino corrisponde a un reparto, quale dei diagrammi è adatto a descrivere lo schema produttivo? Si giustifichi la risposta e si spieghi perché tutti gli altri diagrammi sono da escludere.
2. Si modelli lo schema dei processi produttivi usando una struttura dati opportuna, chiarendo come sia collegata alla situazione descritta; in particolare
   - si descriva, utilizzando una terminologia appropriata, come sono rappresentati i reparti, i passaggi da un reparto all'altro, il numero di lavoratori operativi nei vari reparti;
   - si definisca, con la terminologia appropriata, che cosa sono i reparti di collaudo e il Primo Reparto.
3. Utilizzando opportunamente la struttura dati definita al punto precedente, si progetti un algoritmo che, avendo in input lo schema dei processi produttivi e il numero di lavoratori operativi in ciascun reparto, calcoli quanti sono i lavoratori operativi nei reparti di collaudo.

Note per la consegna. Si scriva la risposta in un file di testo con nome `es4-reparti.txt`.

5 Reparti - implementazione (Esercizio facoltativo)

Si scriva un programma che implementa l'algoritmo progettato al punto 3 dell'esercizio precedente. Si specifichi in un commento al programma quale formato si usa per inserire da standard input lo schema dei processi produttivi e il numero di lavoratori in ogni reparto.

Note per la consegna. Si scriva il programma in un file con nome `es5-reparti.c`.