Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Prova di laboratorio - Appello del 19 gennaio 2022

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!
- 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-insetti.c`
  `es2-funzioni.txt`
  `es3-caverna.txt`
  `es4-caverna.c`.
- Per superare la prova è necessario svolgere (almeno parzialmente) gli esercizi 1, 2 e 3, oppure gli esercizi 3 e 4.

1 Popolazioni di insetti

Un entomologo sta studiando lo sviluppo di una piccola colonia di insetti. Periodicamente rileva quanti individui sono nella colonia e ora sta preparando una relazione sull'andamento della popolazione nella colonia. L'entomologo innanzitutto calcola quante volte si è osservato un aumento di individui rispetto alla misurazione precedente. Considera poi finestre di m rilevazioni consecutive e calcola quante volte osserva un aumento della somma degli individui nella finestra, rispetto alla finestra precedente.

Ad esempio, i dati delle ultime 8 rilevazioni sono le seguenti: 5689, 6258, 4923, 3926, 5916, 6101, 5804, 5707, Il numero di rilevazioni per cui si è osservato un aumento è 3; se consideriamo finestre di 3 rilevazioni consecutive, il numero di volte in cui si è osservato un aumento in una finestra di 3 rilevazioni è invece 2. Le somme degli individue nelle 6 finestre di ampiezza 3 sono infatti:

$$\begin{aligned} 5689+6258+4923 &= 16870 \\ 6258+4923+3926 &= 15107 \\ 4923+3926+5916 &= 14765 \\ 3926+5916+6101 &= 15943 \\ 5916+6101+5804 &= 17821 \\ 6101+5804+5707 &= 17612 \end{aligned}$$

Scrivete un programma che

- legge un intero n che indica il numero di rilevazioni;
- legge un numero m che indica l'ampiezza della finestra ($m < n$);
- stampa il numero di volte in cui si è osservato un aumento rispetto alla misurazione precedente;
- stampa il numero di volte in cui si è osservato un aumento in una finestra di m rilevazioni.

Stimate la complessità in tempo degli algoritmi usati, in funzione di n e m; stimate inoltre quante somme vengono eseguite.

Esempio di esecuzione. Se il programma riceve da standard input:

```
8 3
5689
6258
4923
3926
5916
6101
5804
5707
```

allora il programma stamperà su standard output:

```
3 2
```

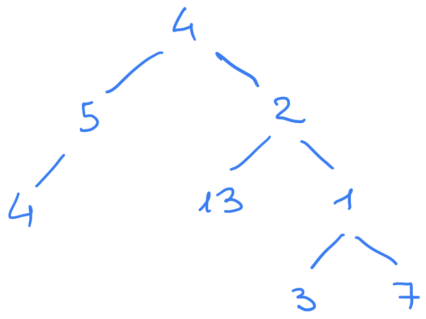
Note per la consegna. Consegnate un programma C con nome `es1-insetti.c`. Inserite in un commento le stime del tempo di esecuzione e del numero di somme eseguite dal programma.

2 Funzioni misteriose - comprensione di codice, strutture dati

Considerate la porzione di codice qui sotto, in cui il tipo `Bit_node` è usato per implementare i nodi di un albero binario:

```
void f( Bit_node p, int x ) {
        if ( !p )
                return;
        p -> y = x + p -> item;
        f( p -> l, p -> y);
        f( p -> r, p -> y);
}
```

1. Scrivete una definizione per il tipo `Bit_node` che sia compatibile con quanto scritto nella funzione.
2. Analizzate gli effetti della funzione `f` se viene invocata con secondo argomento 0, e rispondete alle domande seguenti:
   a) Qual è l'effetto della funzione `f` se viene invocata sulla radice dell'albero disegnato sotto?
   b) Di quanto si alza al massimo lo stack per effetto questa chiamata?
   c) Quante altre chiamate della funzione `f` vengono effettuate per effetto di questa chiamata?
   d) Descrivete a parole cosa fa la funzione `f` se viene invocata sulla radice di un albero binario qualunque.
   e) Date un nome più significativo a `y`.
   f) In generale, di quanto si alza al massimo lo stack se la funzione viene invocata sulla radice di un albero binario qualunque?
   g) In generale, quante chiamate della funzione `f` vengono effettuate quando `f` viene invocata sulla radice di un albero binario qualunque?



Note per la consegna. Scrivete le vostre risposte in un file di testo e salvatelo con il nome `es2-albero.txt`.

3 Caverna ammuffita - modellazione e progettazione

Un robot usato per esplorazioni speleologiche si muove in caverna strettissima, di forma quadrata. L'entrata della caverna si trova in alto a sinistra, l'uscita nell'angolo in basso a destra. Il robot ha due modalità di movimento:

Modalita M1 Il robot può spostarsi *in orizzontale* e *in verticale* (ma non in diagonale).

Modalita M2 Il robot può spostarsi solo *in orizzontale verso destra* oppure *in verticale verso il basso*.

La grotta è ricoperta da muffe, e il contatto con le muffe rischia di danneggiare il robot. Usando i suoi sensori, il robot ha stimato il rischio di venire a contatto con le muffe nelle varie parti della caverna, producendo una mappa dei livelli di rischio (indicata con una matrice di valori da 1 a 9). Il rischio di un percorso all'interno della caverna è dato dalla somma dei rischi delle posizioni in cui il robot entra (il rischio della posizione iniziale non conta).

Considerate i seguenti 2 problemi:

Problema P1 Calcolare il rischio minimo r_1 tra tutti i percorsi che vanno dall'entrata all'uscita della caverna, in modalità M1.

Problema P2 Calcolare il rischio minimo r_2 tra tutti i percorsi che vanno dalla dall'entrata all'uscita della caverna, in modalità M2.

Esempio

Consideriamo la seguente matrice dei livelli di rischio:

7	1	8	4	4
9	1	8	4	4
1	1	8	4	4
1	9	8	4	4
1	1	1	1	1

In questo caso $r_1 = 10$, mentre $r_2 = 16$, che si ottengono rispettivamente con questi cammini (notate che il rischio 7 della posizione d'entrata non viene contato).

7	**1**	8	4	4
9	**1**	8	4	4
1	**1**	8	4	4
1	9	8	4	4
1	**1**	**1**	**1**	**1**

7	**1**	8	4	4
9	**1**	8	4	4
1	**1**	8	4	4
1	**9**	8	4	4
1	**1**	**1**	**1**	**1**

3.1 Modellazione e progettazione

Considerate i seguenti cinque approcci:

1. Modellare la situazione con grafi
2. Modellare la situazione con alberi
3. Usare un approccio ricorsivo
4. Usare la tecnica della programmazione dinamica
5. Usare una tecnica greedy.

Per ciascuno dei due problemi P1 e P2, valutate quali degli approcci suggeriti possono essere utili a risolvere il problema, giustificando la risposta. In particolare seguite i seguenti schemi.

- Se scegliete gli approcci 1 o 2:
  a) descrivete come si può rappresentare la situazione usando grafi/alberi;
  b) riformulate il problema usando la terminologia relativa a grafi/alberi;
  c) indicate eventuali proprietà notevoli di questi grafi/alberi;

d) descrivete come è opportuno implementare la struttura dati scelta;

e) progettate un algoritmo per risolvere il problema, che faccia uso di grafi/alberi.

- Se scegliete l'approccio 3:
  a) descrivete la relazione di ricorrenza individuata;
  b) progettate un algoritmo ricorsivo che risolve il problema.
- Se scegliete l'approccio 4 o 5: progettare e descrivere un algoritmo che risolve il problema usando la programmazione dinamica / una tecnica greedy, discutendone la correttezza.

Se avete trovato soluzioni diverse per uno stesso problema, confrontate le soluzioni e valutate quale considerate migliore e perché.

Note per la consegna. Scrivete le vostre risposte in un file chiamato `es3-caverna.txt`.

4 Caverna ammuffita - Implementazione

Considerate i problemi descritti nell'esercizio precedente. Scrivete un programma C che legge un intero n seguito da una matrice $n \times n$ di cifre con i livelli di rischio, e stampa i rischi r_1 e r_2.

Esempio di esecuzione. Se il programma riceve da standard input:

```
5
7 1 8 4 4
9 1 8 4 4
1 1 8 4 4
1 9 8 4 4
1 1 1 1 1
```

allora il programma stamperà su standard output:

```
r1 = 10
r2 = 16
```

Note per la consegna. Consegnate un programma con nome `es4-caverna.c`.