Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Prova di laboratorio - Appello del 13 giugno 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!
- 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-riordina.c`, `es2-interruttori.c`, `es3-luciGrafo.txt`, `es4-spegnere.txt`.

1 Riordina i bit

Si consideri un array `A`, non vuoto, di n interi, in cui ciascun valore può essere esclusivamente 0 oppure 1. I valori sono presenti senza alcun ordine; è anche possibile che tutti i valori siano uguali.

Si scriva un algoritmo avente complessità ottima che ordini l'array `A` spostando tutti i valori 0 prima di tutti i valori 1. Verrà assegnato punteggio pieno ad algoritmi che scambiano elementi in `A`, che richiedono memoria ulteriore $O(1)$ (quindi che non sfruttano array ausiliari), e che non sono basati sul conteggio del numero di 0 e 1 presenti.

Si scriva un programma che implementa l'algoritmo. Il programma deve leggere da standard input un intero n seguito da una sequenza di n valori 0/1, riordinare l'array e stamparlo.

Note per la consegna. Si scriva il programma in un file di nome `es1-riordina.c`.

Esempio di esecuzione 1

Ricevendo da standard input

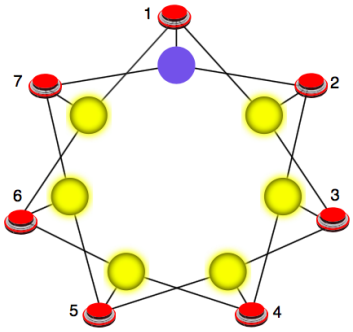
```
7
0 1 1 0 0 0 1
```

il programma deve stampare

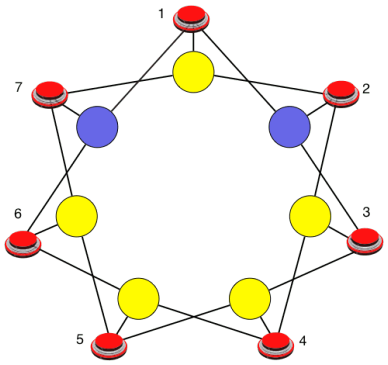
```
0 0 0 0 1 1 1
```

2 Interruttori

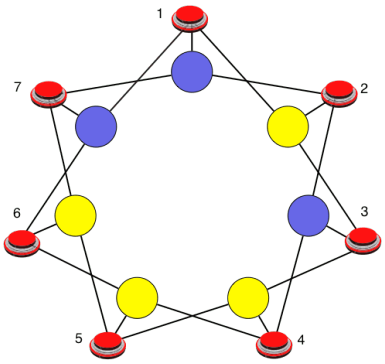
Nella rete di luci rappresentata in figura, ogni interruttore agisce sulle tre luci che collega (le luci sono disegnate in giallo o blu a seconda che siano accese o spente, rispettivamente, mentre gli interruttori sono in rosso).

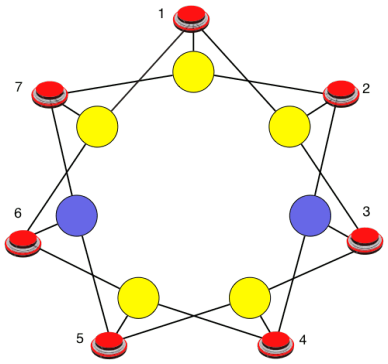


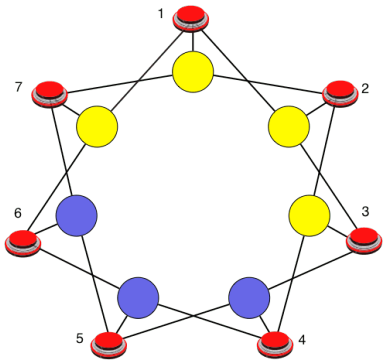
Ad esempio, se si parte con una sola luce spenta come nella figura sopra e si preme l'interruttore 1, si ottiene



Se poi si premono nell'ordine gli interruttori 2, 7 e 4, la situazione evolve come raffigurato qui sotto:







Premendo infine l'interruttore 5, tutte le luci risulteranno accese.

Generalizziamo la situazione considerando un numero arbitrario n di interruttori e di luci (nell'esempio sopra n è pari a 7). Indichiamo sia le luci che gli interruttori con numeri progressivi da 1 a n. Se i è compreso tra 2 e $n-1$, allora l'interruttore i è collegato con le luci $i-1$, i e $i+1$. Inoltre, l'interruttore 1 è collegato con le luci n, 1 e 2, mentre l'interruttore n è collegato con le luci $n-1$, n e 1. L'intero n è chiamato *dimensione della rete*.

Si scriva un programma che simuli l'uso di interruttori in questo tipo di reti, tenendo in memoria in ogni istante lo *stato di accensione delle luci della rete*. Lo stato di accensione delle luci è descritto da una sequenza di bit: il bit di posizione i indica se la luce i è accesa o spenta. Il programma deve leggere da standard input una sequenza di istruzioni, scritte una per riga, secondo il formato nella seguente tabella, dove n e m sono interi positivi, $\ell_1 \dots \ell_n$ sono bit, $i, i_1, \dots i_m$ sono numeri compresi tra 1 e n.

Istruzione	**Operazione**
`+` n	Crea una rete con n interruttori e n luci tutte spente. Se esiste già una rete, la elimina e ne costruisce una nuova.
`o` $\ell_1\ell_2\dots\ell_n$	Imposta la rete allo stato di accensione $\ell_1\ell_2\dots\ell_n$.
`p`	Stampa lo stato di accensione della rete.
`s` i	Preme l'interruttore i.
`S` $m\ i_1 i_2 \dots i_m$	Preme in sequenza gli interruttori $i_1\ i_2\ \dots i_m$.
`f`	Termina l'esecuzione

Per tutti i comandi diversi da `+` e `f`, se non è stata creata ancora nessuna rete, si deve stampare un messaggio d'errore.

Si noti che ilprimo argomento del comando *S* indica la lunghezza della sequenza di interruttori successiva. Quindi, in particolare, il comando `S 1 4` è equivalente al comando `s 4`.

Esempio di esecuzione

Simuliamo il caso descritto precedentemente (vedi figure): ricevendo da standard input

```
+7
p
o 0 1 1 1 1 1 1
p
s 1
p
S 3 2 7 4
p
s 5
p
f
```

il programma deve stampare

```
0 0 0 0 0 0 0
0 1 1 1 1 1 1
1 0 1 1 1 1 0
1 1 1 0 0 0 1
1 1 1 1 1 1 1
```

Si noti che l'effetto del comando `S 3 2 7 4` è quello di premere in sequenza i 3 interruttori 2, 7 e 4.

Note per la consegna. Si scriva il programma in un file con nome `es2-interruttori.c`.

3 Luci e grafi

Usando il concetto di grafo, si descriva la topologia della rete di luci presentata nell'esercizio 2, specificando in particolare quali sono gli insiemi dei nodi e degli archi, e illustrandone le caratteristiche interessanti (orientato, connesso, pesato, colorato, aciclico, bipartito...)

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

4 Spegnere tutte le luci

Si consideri ora il seguente problema di *spegnimento delle luci*: data una rete di luci in un certo stato di accensione iniziale, determinare una sequenza di interruttori che consenta di spegnere tutte le luci.

Si risponda quindi ai seguenti punti.

1. Si usi un grafo per modellare l'evolvere dello stato di accensione della rete.

   Nota bene: questo grafo non ha nulla a che vedere con la topologia della rete descritta dal grafo dell'esercizio precedente!!!

2. Si sfrutti il grafo definito al punto precedente per progettare un algoritmo che risolva il problema dello spegnimento delle luci. Si descriva l'algoritmo specificando anche quali informazioni vanno tenute in memoria e come vanno rappresentate.

3. La sequenza trovata dall'algoritmo descritto al punto precedente è di lunghezza minima? Se sì, si spieghi perché, altrimenti si fornisca un esempio in cui l'algoritmo non fornisce una sequenza ottimale.

4. Si progetti e descriva un algoritmo che trovi una delle sequenze di linghezza minima che consenta di spegnere tutte le luci.

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

5 Spegnere le luci - implementazione (Esercizio facoltativo)

Si scriva un programma che implementa l'algoritmo progettato al punto 4 dell'esercizio precedente.

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