Laboratorio di Algoritmi e Strutture Dati

Docenti: M. Torelli, S. Aguzzoli

Progetto “Die Hard”

valido per gli appelli di gennaio e febbraio 2010

Questo progetto è valido per gli appelli dell'8 gennaio, del 28 gennaio e dell'11 febbraio 2010. Le date ultime di consegna per i rispettivi appelli sono:

- 11 gennaio per l'appello dell'8 gennaio;
- 1 febbraio per l'appello del 28 gennaio;
- 15 febbraio per l'appello dell'11 febbraio.

1 Il problema

Nel film “Die hard 3”, i due eroi John (Bruce Willis) e Zeus (Samuel L. Jackson) devono disinnescare una bomba risolvendo il gioco seguente. Vicino ad una fontana ci sono due contenitori aventi capacità 3 e 5 galloni. Per bloccare il timer dell'ordigno è necessario appoggiare sopra la bomba il contenitore da 5 galloni con esattamente 4 galloni di acqua al suo interno. Il gioco consiste quindi nel misurare esattamente 4 galloni d'acqua utilizzando solamente i due contenitori e le operazioni di riempimento, svuotamento e travaso da un contenitore all'altro.

Ovviamente siamo in un film americano e i due eroi riescono nell'intento partendo dalla configurazione inizialmente vuota `(0[3],0[5])` e utilizzando la seguente sequenza di operazioni (lo stato di ogni contenitore è indicato con un intero non negativo seguito dalla sua capacità tra parentesi quadre):

1. riempire il contenitore da 5 galloni: `(0[3],5[5])`;
2. travasare il contenitore da 5 in quello da 3: `(3[3],2[5])`;
3. svuotare il contenitore da 3: `(0[3],2[5])`;
4. travasare il contenitore da 5 in quello da 3: `(2[3],0[5])`;
5. riempire il contenitore da 5: `(2[3],5[5])`;
6. travasare il contenitore da 5 in quello da 3: `(3[3],4[5])`;

Attenzione: Come avrete notato dall'esempio, non è possibile travasare una quantità arbitraria di acqua. Il riempimento va fatto fino all'orlo, lo svuotamento deve essere completo e il travaso dal contenitore A al contenitore B si conclude con il contenitore A vuoto (ad esempio l'operazione 4 della sequenza) o col contenitore B pieno (ad esempio, l'operazione 6 della sequenza).

In questo progetto si chiede di risolvere il problema dei contenitori nel caso generale. Si deve gestire una sequenza di n contenitori di capacità $c_1, c_2, \ldots, c_n$, dove c_i è un intero positivo per ogni $i \in \{1, 2, \ldots . n\}$, inizialmente vuoti.

Denotiamo con un vettore $(a_1, a_2, \ldots, a_n)$ di interi la *configurazione* dei livelli d'acqua nei contenitori, dove $0 \leq a_i \leq c_i$ per ogni $i = 1, \ldots, n$.

Le operazioni di *riempimento* di un contenitore, *svuotamento* di un contenitore, *travaso* di un contenitore in un altro sono dette *operazioni elementari.*

Una configurazione $b = (b_1, b_2, \ldots, b_n)$ è *direttamente raggiungibile* da una configurazione $a = (a_1, a_2, \ldots, a_n)$ se a è trasformata in b da un'operazione elementare.

Una configurazione $b = (b_1, b_2, \ldots, b_n)$ è *raggiungibile* da una configurazione $a = (a_1, a_2, \ldots, a_n)$ se esiste una sequenza finita di configurazioni

$$x_1, x_2, \ldots, x_m$$

tali che $x_1 = a$, $x_m = b$ e, per ogni $i \in \{1, 2, \ldots, m-1\}$ la configurazione x_{i+1} è direttamente raggiungibile dalla configurazione x_i. Ogni sequenza $x_1, x_2, \ldots, x_m$ con queste proprietà è detta sequenza *ammissibile* di *lunghezza* m.

In alcuni casi stabiliremo che una configurazione $a = (a_1, \ldots, a_n)$ è *pericolosa* (poiché, per qualche motivo non precisato, tale configurazione indurrebbe lo scoppio della bomba). Ovviamente, una configurazione pericolosa deve essere sempre evitata. Una configurazione non pericolosa è detta *innocua*.

In alcuni casi supporremo che versare acqua fuori dai contenitori induca lo scoppio della bomba. In queste situazioni l'operazione elementare di svuotamento non potrà essere effettuata.

2 Specifiche di implementazione

Il programma deve leggere dallo standard input (`stdin`) una sequenza di righe (separate da `\n`), ciascuna delle quali corrisponde a una riga della prima colonna della Tabella 1, dove $c_1, c_2, \ldots, c_n$, i, j sono numeri interi positivi maggiori di 0 mentre d, k, h e $a_1, a_2, \ldots, a_n$ sono numeri interi ≥ 0.

I vari elementi sulla riga sono separati da uno o più spazi. Quando una riga è letta, viene eseguita l'operazione associata; le operazioni di stampa sono effettuate sullo standard output (`stdout`), e ogni operazione deve iniziare su una nuova riga.

Note e suggerimenti

1. Non si richiede – anzi si sconsiglia – l'uso di grafica, se non per test personali: in modo particolare, non si usi `conio.h` e neppure `clrscr()`.
2. Per semplicità si suppone che l'input sia sempre conforme alle specifiche di Tabella 1, per cui non è necessario controllare la correttezza dell'input. Per leggere l'input si usino le funzioni standard ANSI C `getchar()` e/o `scanf()`.

3 Operazioni da implementare

Le operazioni da implementare variano in modo incrementale a seconda dell'appello per il quale avviene la consegna: in particolare, chi consegnerà entro l'11 gennaio dovrà implementare le operazioni specificate nella Sezione 3.1; chi consegnerà entro l'1 febbraio dovrà implementare le operazioni specificate nelle Sezioni 3.1 e 3.2; chi consegnerà entro il 15 febbraio dovrà implementare le operazioni specificate nelle Sezione 3.1, 3.2 e 3.3. Per i primi due appelli, l'implementazione delle operazioni specificate nelle sezioni relative alle consegne successive è da considerare facoltativa.

Si noti che le operazioni richieste sono liberamente implementabili; in particolare, non vanno necessariamente intese come prototipi di funzioni.

3.1 Per l'appello dell'8 gennaio (consegna entro l'11 gennaio)

- **crea_contenitori**(c)

Riga di input	Operazione
`N` $c_1\ c_2\ \cdots\ c_n$	**crea_contenitori**(n, c), dove $c = (c_1, \ldots, c_n)$
`R` i	**riempi**(i)
`S` i	**svuota**(i)
`T` $i\ j$	**travasa**(i, j)
`v`	**visualizza**()
`e` k	**esiste**(k)
`r` $a_1\ a_2\ \cdots\ a_n$	**raggiungibile**(a), dove $a = (a_1, \ldots, a_n)$
`c` d	**configurazioni**(d)
`f`	esce dal programma.
`w` k	**contenenti**(k)
`m` k	**mosse**(k)
`p` $a_1\ a_2\ \cdots\ a_n$	**pericolosa**(a), dove $a = (a_1, \ldots, a_n)$
`i` $a_1\ a_2\ \cdots\ a_n$	**innocua**(a), dove $a = (a_1, \ldots, a_n)$
`k` $a_1\ a_2\ \cdots\ a_n$	**critica**(a), dove $a = (a_1, \ldots, a_n)$
`A`	**attiva**()
`D`	**disattiva**()
`m` $k\ h$	**mosse**(k, h)

Tabella 1: Specifiche del programma

Dato un vettore $c = (c_1, c_2, \ldots, c_n)$ di interi positivi, inizializza una sequenza di n contenitori inizialmente vuoti aventi capacità $c_1, c_2, \ldots, c_n$. Ad esempio, **crea_contenitori**$(2, 3, 6)$ inizializzerà la configurazione `(0[2],0[3],0[6])`. Se esiste già una sequenza di contenitori, la cancella e ne crea una nuova.

- **riempi**(i)

  Il contenitore i viene riempito fino all'orlo: il suo livello diventa c_i. Ad esempio, **riempi**(2) sulla configurazione `(0[2],2[3],0[6])` porta alla configurazione `(0[2],3[3],0[6])`. L'operazione non è valida se il contenitore i è già pieno. In tal caso stampa `OPERAZIONE NON VALIDA`.

- **svuota**(i)

  Il contenitore i viene svuotato completamente: il suo livello diventa 0. Ad esempio, **svuota**(2) sulla configurazione `(0[2],2[3],0[6])` porta alla configurazione `(0[2],0[3],0[6])`. L'operazione non è valida se il contenitore i è già vuoto. In tal caso stampa `OPERAZIONE NON VALIDA`.

- **travasa**(i, j)

L'acqua del contenitore i viene versata nel contenitore j fino al completo riempimento di j o al completo svuotamento di i.
Ad esempio, data la configurazione `(0[2],3[3],2[6])`, **travasa**$(2, 1)$ porta alla configurazione `(2[2],1[3],2[6])` mentre **travasa**$(2, 3)$ porta alla configurazione `(0[2],0[3],5[6])` L'operazione non è valida se i è già vuoto o se j è già pieno. In tal caso stampa `OPERAZIONE NON VALIDA`.

- **visualizza**()

  Visualizza la configurazione attuale.

- **esiste**(k)

  Stampa `SI'` oppure `NO` a seconda che dalla configurazione attuale sia possibile o meno raggiungere una configurazione in cui almeno un contenitore ha livello k.

- **raggiungibile**(a)

  Dato un vettore $a = (a_1, a_2, \ldots, a_n)$ di interi positivi, stampa `SI'` oppure `NO` a seconda che dalla configurazione attuale sia possibile raggiungere una configurazione in cui, per ogni $i \in \{1, 2, \ldots, n\}$, il contenitore i ha livello a_i.

- **configurazioni**(d)

  Stampa tutte le configurazioni raggiungibili dalla configurazione attuale eseguendo esattamente d operazioni elementari. Ad esempio, dalla configurazione `(2[3],0[5])` con una mossa si possono raggiungere le configurazioni `(3[3],0[5])` con **riempi**(1), `(2[3],5[5])` con **riempi**(2), `(0[3],0[5])` con **svuota**(1) e `(0[3],2[5])` con **travasa**$(1, 2)$.

3.2 Per l'appello del 28 gennaio (consegna entro l'1 febbraio)

- **contenenti**(k)

  Stampa tutte le configurazioni raggiungibili dalla configurazione attuale e contenenti il valore k come livello di almeno un contenitore. Se non esistono configurazioni soddisfacenti la condizione, stampa `NON PRESENTI!`.

- **mosse**(k)

  Stampa una sequenza ammissibile di lunghezza minima tra quelle che portano dalla configurazione attuale ad una configurazione contenente il valore k come livello di almeno un contenitore. Se non esiste nessuna configurazione raggiungibile che contiene il valore k, allora stampa `IRRAGGIUNGIBILE!`.
  Ad esempio, John e Zeus hanno seguito la più corta sequenza di operazioni che porta ad una configurazione contenente un 4: `(0[3],0[5])`, `(0[3],5[5])`, `(3[3],2[5])`, `(0[3],2[5])`, `(2[3],0[5])`, `(2[3],5[5])`, `(3[3],4[5])`.

- **pericolosa**(a)

  Dato un vettore $a = (a_1, a_2, \ldots, a_n)$ di interi positivi, dichiara "pericolosa" la configurazione in cui il contenitore i ha livello a_i. Questo implica che questa configurazione non potrà più essere utilizzata nel calcolo di altre operazioni (quali ad esempio **contenenti**(k) o **mosse**(k)) fino a quando non sia nuovamente dichiarata come innocua con l'operazione **innocua**(a) descritta sotto. In particolare se una operazione elementare trasforma la configurazione attuale in una configurazione pericolosa, l'operazione non deve essere eseguita, e deve essere stampato il messaggio `OPERAZIONE PERICOLOSA`.

- **innocua**(a)

  Dato un vettore $a = (a_1, a_2, \ldots, a_n)$ di interi positivi, dichiara "innocua" la configurazione in cui il contenitore i ha livello a_i. Questo implica che questa configurazione potrà essere nuovamente utilizzata nel calcolo di altre operazioni (quali ad esempio **contenenti**(k) o **mosse**(k)).

3.3 Per l'appello dell'11 febbraio (consegna entro il 15 febbraio)

- **critica**(a)

  Stampa, se esiste, una configurazione a' (distinta da a e dalla configurazione attuale) tale che ogni sequenza di operazioni elementari dalla configurazione attuale alla configurazione a contiene a'. Se tale configurazione non esiste, allora stampa `NON ESISTE CONFIGURAZIONE CRITICA`. (Si osservi che per una data configurazione c possono esistere più configurazioni critiche distinte. Basta stamparne una di queste.)

- **mosse**(k, h)

  Stampa una sequenza ammissibile di lunghezza minima tra quelle che portano da una configurazione avente un contenitore riempito a livello k a una avente un contenitore riempito a livello h. Se non esiste nessuna di tali configurazioni, allora stampa `NON E' POSSIBILE!`.

- **disattiva**()

  Disattiva la possibilità di svuotare i contenitori, fino a quando tale possibilità non viene riattivata tramite l'operazione **attiva**(). Questo implica che ogni operazione elementare **svuota**(i) dovrà essere ignorata, e che nel calcolo di altre operazioni (quali ad esempio **mosse**(k) o **mosse**(k, h)) l'operazione **svuota**(i) non potrà essere utilizzata.

- **attiva**()

  Riattiva la possibilità di svuotare i contenitori.

4 Esempi

Di seguito sono mostrati tre esempi di esecuzione. Le righe che iniziano con il simbolo `>` sono da intendersi come righe di input.

Esempio 1 (Operazioni richieste per l'appello dell'8 gennaio).

```
>N 3 5
>S 1

OPERAZIONE NON VALIDA.

>c 2

(0[3],0[5])
(3[3],5[5])
(3[3],2[5])
(0[3],3[5])

>e 4

SI'

>e 1

SI'

>e 7
```

```
NO

> r 2 0

SI'

> r 1 2

NO

>R 1
>T 1 2
>v

(0[3],3[5])

>R 1
>T 1 2
>S 2
>v

(1[3],0[5])

>T 1 2
>R 1
>T 1 2
>v

(0[3],4[5])

>N 2 3 6
>R 1
>T 1 2
>v

(0[2],2[3],0[6])

>R 1
>T 1 3
>v

(0[2],2[3],2[6])

>R 1
>R 2
>R 3
>v

(2[2],3[3],6[6])

>c 2
```

```
(2[2],3[3],6[6])
(0[2],0[3],6[6])
(0[2],3[3],0[6])
(2[2],1[3],6[6])
(2[2],3[3],4[6])
(2[2],0[3],0[6])
(0[2],2[3],6[6])
(2[2],3[3],3[6])
(0[2],3[3],2[6])
(2[2],0[3],3[6])

>S 2
>S 3
>T 1 2
>R 1
>T 1 2
>v

(1[2],3[3],0[6])

>f
```

Esempio 2. (Operazioni richieste per l'appello del 28 gennaio).

```
>N 3 5
>m 4

(0[3],0[5])
(0[3],5[5])
(3[3],2[5])
(0[3],2[5])
(2[3],0[5])
(2[3],5[5])
(3[3],4[5])

>R 1
>T 1 2
>R 1
>T 1 2
>v

(1[3],5[5])

>m 2

(1[3],5[5])
(0[3],5[5])
(3[3],2[5])

>p 0 5
>S 2
```

```
>S 1
>m 4

(0[3],0[5])
(3[3],0[5])
(0[3],3[5])
(3[3],3[5])
(1[3],5[5])
(1[3],0[5])
(0[3],1[5])
(3[3],1[5])
(0[3],4[5])

>p 3 0
>m 4

IRRAGGIUNGIBILE!

>p 3 5
>i 3 0

>m 4

(0[3],0[5])
(3[3],0[5])
(0[3],3[5])
(3[3],3[5])
(1[3],5[5])
(1[3],0[5])
(0[3],1[5])
(3[3],1[5])
(0[3],4[5])

>w 2

(3[3],2[5])
(0[3],2[5])
(2[3],0[5])
(2[3],5[5])

>p 2 0
>w 2

(2[3],5[5])

>f
```

Esempio 3. (Operazioni richieste per l'appello dell'11 febbraio).

```
>N 3 5
>R 1
>R 2
```

```
>D
>e 0

NO

>A
>m 0

(3[3],5[5])
(0[3],5[5])

>D
>k 0 0

NON ESISTE CONFIGURAZIONE CRITICA

>A
>k 0 0

NON ESISTE CONFIGURAZIONE CRITICA

>S 2
>D
>k 1 5

(3[3],3[5])

>m 1 2

(3[3],1[5])
(0[3],4[5])
(3[3],4[5])
(2[3],5[5])

>m 2 1

(2[3],0[5])
(3[3],0[5])
(0[3],3[5])
(3[3],3[5])
(1[3],5[5])

>p 2 0
>m 2 1

NON E' POSSIBILE!

>i 2 0
>A
>m 1 2

(1[3],5[5])
```

```
(0[3],5[5])
(3[3],2[5])

>m 2 1

(2[3],5[5])
(3[3],4[5])
(0[3],4[5])
(3[3],1[5])

>f
```

5 Presentazione del progetto

Il progetto deve essere inviato per posta elettronica all'indirizzo `aguzzoli@dsi.unimi.it` entro la data di consegna (inclusa) relativa all'appello a cui si partecipa.

La discussione del progetto e l'esame orale si svolgeranno in data e luogo da specificarsi (consultare al riguardo il sito: `http://homes.dsi.unimi.it/∼aguzzoli/algo.htm`).

Occorre presentare:

1. il codice sorgente (rigorosamente ANSI C, compilabile con **gcc**);
2. una sintetica relazione (formato pdf o rtf) che illustra le strutture dati utilizzate e analizza il costo delle diverse operazioni richieste dalla specifica.

I due o più file (file sorgenti C + relazione) devono essere contenuti in un unico file `.zip` il cui nome dovrà essere `cognome.zip`. La relazione e il codice devono riportare il vostro nome, cognome e matricola.

Una copia cartacea della relazione e del codice deve inoltre essere consegnata al dr. Aguzzoli entro la data di consegna (lasciandola eventualmente nella sua casella postale presso il dipartimento in via Comelico).

Si ricorda infine di presentarsi alla prova orale con una copia stampata della relazione e del codice.

Il calendario degli esami orali sarà disponibile sulla pagina del corso
`http://homes.dsi.unimi.it/∼aguzzoli/algo.htm`)
qualche giorno dopo il termine di consegna del progetto.

Per ogni ulteriore chiarimento:
E-mail: `aguzzoli@dsi.unimi.it`
Ricevimento: il mercoledì, ore 15-16, stanza S204.

5.1 Avvisi

La versione aggiornata del progetto è pubblicata in .pdf sul sito:
`http://homes.dsi.unimi.it/∼aguzzoli/algo.htm`.
Si consiglia di consultare periodicamente questo sito per eventuali correzioni e/o precisazioni relative al testo del progetto.

Si richiede allo studente di effettuare un adeguato collaudo del proprio progetto su numerosi esempi diversi per verificarne la correttezza e valutarne le prestazioni.

La realizzazione del progetto è una prova d'esame da svolgersi **individualmente**. I progetti giudicati frutto di **collaborazioni** saranno **estromessi** d'ufficio dalla valutazione.