Laboratorio di Algoritmi e Strutture Dati

Docenti: M. Torelli, S. Aguzzoli

Appello del 19 febbraio 2008

Progetto "Same Game 3"
Consegna entro l'11 marzo 2008

Il problema

Si tratta di una generalizzazione del gioco *Same Game* che nella sua versione originale si gioca su una scacchiera rettangolare di dimensioni $r \times c$. All'inizio ogni casella della scacchiera contiene una *biglia* colorata. Lo scopo del gioco consiste nel cercare di rimuovere tutte le biglie dalla scacchiera per mezzo di semplici regole che permettono di rimuovere certi insiemi di biglie dello stesso colore.

Nella generalizzazione proposta la scacchiera è il piano $\mathbb{Z}^2$ degli interi

$$\mathbb{Z}^2 = \{\, (x, y) \mid x, y \in \mathbb{Z} \,\},$$

dove una *casella* è una coppia di interi $(x, y) \in \mathbb{Z}^2$.

Una casella (x, y) può essere vuota oppure contenere una (e una sola) biglia. A ogni biglia è associato un *colore* σ, dove σ è una parola di lunghezza arbitraria sull'alfabeto $\{a, \ldots, z\}$, e un *valore* intero $v > 0$. Se una biglia b si trova in (x, y), diciamo che b si trova alla riga y e colonna x di $\mathbb{Z}^2$.

Le casella (x, y) di $\mathbb{Z}^2$ è *adiacente* alla casella (x', y') se e solo se (x', y') è una delle caselle

$$(x-1, y),\ (x+1, y),\ (x, y-1),\ (x, y+1)\,.$$

Due caselle (x', y') e (x'', y'') sono *connesse* se esiste una sequenza di caselle $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ tale che:

1. per ogni $i \in \{1, \ldots, n-1\}$, la casella (x_i, y_i) contiene una biglia,
2. $(x_1, y_1) = (x', y')$ e $(x_n, y_n) = (x'', y'')$,
3. (x_i, y_i) è adiacente a (x_{i+1}, y_{i+1}) per ogni $i \in \{1, \ldots, n-1\}$.

Un *blocco* di colore σ è un insieme B di caselle connesse che è massimale rispetto a σ, vale a dire:

1. ogni casella di B contiene una biglia di colore σ,
2. se C è un altro insieme di caselle connesse contenenti solo biglie di colore σ e $B \cap C \neq \emptyset$, allora $C \subseteq B$.

Dato un blocco B, una *colonna* x_0 di B è l'insieme delle celle (x_0, y) tali che (x_0, y) è una cella di B. La *cancellazione* di una colonna x_0 di B consiste nel cancellare tutte le biglie in B che si trovano nella colonna x_0.

Un blocco B può essere *ridotto* eliminando una o più colonne secondo la seguente regola:

- se una colonna x_0 di B è cancellata, non è possibile cancellare né la colonna $x_0 - 1$ né la colonna $x_0 + 1$.

Il *punteggio* ottenuto dalla *riduzione* di B è dato dalla somma dei valori delle biglie cancellate. L'obiettivo della riduzione è quello di eliminare le colonne in modo da ottenere il massimo punteggio possibile. Quindi, se il punteggio ottenuto riducendo B è V, non è possibile ridurre B ottenendo un punteggio $V' > V$. È invece possibile che vi sia più di un modo per ridurre B con punteggio massimo. In tal caso, la scelta dell'insieme di colonne da cancellare, fra quegli insiemi che ottengono tale punteggio, è arbitraria.

Quando le colonne sono cancellate, le biglie del blocco non eliminate vengono risistemate nel modo seguente. Sia x_{min} la colonna di indice minimo nel blocco B prima della eliminazione delle colonne e siano $b_1, \ldots, b_n$ le biglie di B non cancellate. È possibile spostare una biglia b_k $(1 \leq k \leq n)$ applicando la seguente regola di spostamento:

(s) Se b_k si trova nella casella (x, y), con $x > x_{min}$, e se la casella $(x-1, y)$ è vuota, allora b_k si sposta in $(x-1, y)$.

Le biglie $b_1, \ldots, b_n$ vanno spostate applicando la regola (s) fintanto che è possibile. Si noti che, indipendentemente dall'ordine in cui vengono effettuati gli spostamenti, la configurazione finale è la stessa. Le biglie che non appartengono a B non si spostano.

Esempio 1

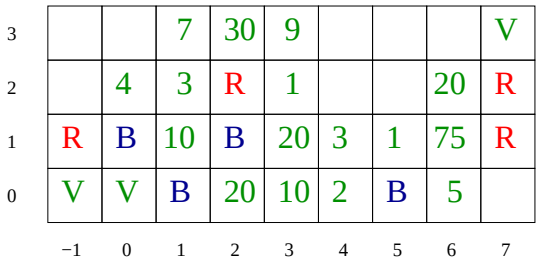
Supponiamo che le biglie siano disposte come nella figura, dove B indica una biglia di colore `blu` (per semplicità non viene indicato il valore), R `rosso` e V per `verde`. Una cella contrassegnata da un intero n indica che in essa si trova una biglia di colore `verde` e valore n. Tutte le altre celle del piano sono vuote.

3			7	30	9				V
2		6	3	R	1			20	R
1	R	B	10	B	20	3	1	75	R
0	V	V	B	20	10	2	B	5	
	−1	0	1	2	3	4	5	6	7

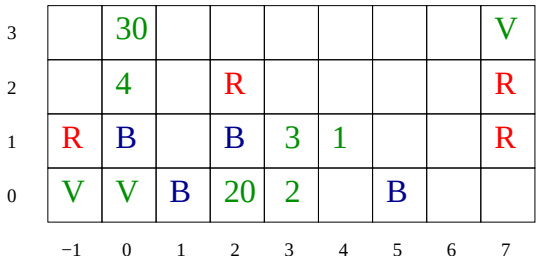
Supponiamo di voler ridurre il blocco B cui appartiene la cella $(1, 1)$. Tale blocco è formato dalle 16 celle contenenti biglie di colore `verde` di cui nella figura è riportato il valore (si noti che le celle $(-1, 0)$, $(0, 0)$ e $(7, 3)$ non appartengono a B). Il punteggio massimo si ottiene eliminando le colonne 0, 2, 4 e 6. Infatti, con tale scelta il punteggio ottenuto è 161, con qualunque altra scelta si ottiene un punteggio minore o uguale a 160. Dopo la eliminazione delle biglie nelle colonne selezionate e lo spostamento delle rimanenti si ha:

3		7	9						V
2		3		R	1				R
1	R	B	10	B	20	1			R
0	V	V	B	10			B		
	−1	0	1	2	3	4	5	6	7

Supponiamo ora che nel blocco B di prima il valore della biglia in $(0, 2)$ sia 4 anziché 6, come nella seguente figura.



In questo caso, il punteggio ottenuto riducendo B è 160, ottenuto cancellando le colonne 1, 3 e 6. Dopo la riduzione si ha:



Si noti che, se il valore della biglia in $(0, 2)$ nel blocco B fosse 5, il punteggio ottenuto riducendo B sarebbe 160, ottenibile cancellando le colonne 0,2,4,6 oppure, equivalentemente, le colonne 1,3,6.

Fra i colori è definita una *relazione d'ordinamento parziale stretto* $<$. Questo significa che $<$ deve verificare le seguenti proprietà degli ordinamenti stretti:

(P1). Per ogni colore σ_1 e σ_2, se $\sigma_1 < \sigma_2$, allora non vale $\sigma_2 < \sigma_1$.

(P2). Per ogni colore σ_1, σ_2 e σ_3, se $\sigma_1 < \sigma_2$ e $\sigma_2 < \sigma_3$, allora $\sigma_1 < \sigma_3$.

L'ordinamento dei colori è stabilito dall'utente e il programma deve controllare che l'introduzione di nuove asserzioni del tipo $\sigma_1 < \sigma_2$ non vada a violare le proprietà scritte sopra.

Diciamo che due blocchi B_1 e B_2 sono *adiacenti* se esistono una cella $(x_1, y_1) \in B_1$ e una cella $(x_2, y_2) \in B_2$ tali che (x_1, y_1) è adiacente a (x_2, y_2). La *fusione* di due blocchi B_1 e B_2 di colore rispettivamente σ_1 e σ_2 può avvenire se sono verificate le seguenti condizioni:

1. B_1 è adiacente a B_2
2. $\sigma_1 < \sigma_2$ oppure $\sigma_2 < \sigma_1$.

La fusione consiste nel colorare le biglie di entrambi i blocchi B_1 e B_2 con il colore $\max(\sigma_1, \sigma_2)$ in modo da ottenere un unico blocco contenente sia B_1 che B_2. Si noti che il blocco risultante può *contenere propriamente* $B_1 \cup B_2$. Ciò accade quando il blocco del colore minore fra B_1 e B_2 risulta adiacente ad altri blocchi del colore maggiore.

Esempio 2

Supponiamo che la relazione fra i colori sia così definita:

```
azzurro < giallo    azzurro < rosso    blu < rosso

giallo < nero       rosso < nero       verde < nero
```

Allora, non è possibile fondere un blocco di colore `azzurro` con uno di colore `verde`, in quanto non vale `azzurro < verde` e neppure `verde < azzurro`. È invece possibile fondere un blocco di colore `azzurro` con uno di colore `nero` e, dopo la fusione, il blocco ottenuto ha colore `nero`, essendo `azzurro < nero`.

Si noti che l'utente non può aggiungere all'ordinamento l'asserzione `nero < azzurro`, in quanto vale già `azzurro < nero`. È possibile invece inserire, ad esempio, `giallo < verde`, `azzurro < verde`, ecc.

Si richiede di implementare una struttura dati efficiente che permetta di eseguire le operazioni seguenti: (si tenga presente che la minima porzione rettangolare di piano contenente tutte le biglie può essere molto grande rispetto al numero di biglie presenti nel piano, quindi *non è sicuramente efficiente rappresentare l'insieme delle biglie mediante un'*unica *matrice*).

- **input** $(r, c, x, y, nomefile)$

  Legge dal file $nomefile$ una tabella di biglie colorate di r righe e c colonne e colloca le biglie nelle caselle corrispondenti dell'insieme $\{(x+h, y+k) \mid 0 \leq h < c,\ 0 \leq k < r\}$ secondo le regole specificate nell'apposita sezione.

- **biglia** (x, y, v, α)

  Pone in (x, y) una biglia di colore α e di valore v, indipendentemente dal contenuto precedente di (x, y).

- **riduzione** (x, y)

  Se la casella (x, y) contiene una biglia, esegue la riduzione del blocco contenente la biglia nella casella (x, y) e stampa il punteggio ottenuto. Altrimenti non esegue alcuna operazione.

- **minore** (α, β)

  Se l'asserzione $\alpha < \beta$ non viola i requisiti (P1) e (P2) allora la introduce nella relazione d'ordinamento stretto $<$. Altrimenti stampa:

  `Non posso inserire` α `<` β`.`

- **fusione** (x_0, y_0, x_1, y_1)

  Se la fusione fra i due blocchi contenenti rispettivamente le biglie (x_0, y_0) e (x_1, y_1) può avvenire, la effettua. Altrimenti non esegue alcuna operazione.

- **numeroBlocchi** ()

  Stampa il numero di blocchi attualmente presenti nel piano.

- **stampaBlocco** (x, y)

  Se la casella (x, y) contiene una biglia, allora stampa il blocco che la contiene secondo il formato specificato nell'apposita sezione. Altrimenti non esegue alcuna operazione.

All'inizio del programma il piano è vuoto e non vi sono relazioni tra i colori. Si noti che le operazioni richieste sono liberamente implementabili; in particolare, non vanno necessariamente intese come prototipi di funzioni.

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 $nomefile$ è il nome di un file, α e β sono stringhe finite sull'alfabeto $a, b, \ldots, z$ di lunghezza *arbitraria*, x, y, x_0, y_0, x_1, y_1 sono interi e v è un intero > 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.

Riga di input	Operazione
`i` $r\ c\ x\ y\ nomefile$	**input** $(r, c, x, y, nomefile)$
`b` $x\ y\ v\ \alpha$	**biglia** (x, y, v, α)
`r` $x\ y$	**riduzione** (x, y)
`<` $\alpha\ \beta$	**minore** (α, β)
`F` $x_0\ y_0\ x_1\ y_1$	**fusione** (x_0, y_0, x_1, y_1)
`n`	**numeroBlocchi** ()
`s` $x\ y$	**stampaBlocco** (x, y)
`f`	Termina l'esecuzione del programma

Tabella 1: Specifiche del programma

Note

1. Non devono essere presenti vincoli sul numero di colori, blocchi, biglie e sulla lunghezza dei nomi di colori (se non quelli determinati dal tipo di dato intero). 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. **Specifiche per la lettura di una tabella di biglie colorate da un file**

   Si consideri il comando:

   `i` $r\ c\ x\ y\ nomefile$.

   Allora il file di nome $nomefile$ contiene una sequenza di $r \times c$ coppie α, v_α dove, α è una stringa finita sull'alfabeto $\{a, b, \ldots, z\}$ oppure `0`, mentre v_α è un intero ≥ 0. Se α è la stringa `0` allora v_α deve essere 0. Se α non è la stringa `0` allora $v_\alpha > 0$. I colori delle biglie nel piano, e i loro rispettivi valori, vengono modificati in questo modo: Per ogni $0 \leq h < c$ e ogni $0 \leq k < r$:

   - se la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h+1)$ nel file è diversa da `0` allora si pone in $(x+h, y+k)$ una biglia di colore $\alpha_{h,k}$ e gli si assegna valore $v_{\alpha_{h,k}}$;

- se invece la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h+1)$ nel file è uguale a `0` non si modifica il contenuto della casella $(x+h, y+k)$.

Si assume che la prima stringa nel file sia in posizione 1 e l'ultima sia in posizione $r \cdot c$.

Ad esempio, se il comando è `i 2 3 4 5 tabella.txt`, e il contenuto del file `tabella.txt` è il seguente:

```
blu,5 rosso,4 0,0
bianco,7 0,0 nero,11
```

allora i colori delle biglie vengono modificati in questo modo:

$(4,5) =$ `blu`, $(5,5) =$ `rosso`, $(4,6) =$ `bianco`, $(6,6) =$ `nero`.

Nessun'altra casella viene modificata. In particolare le celle $(6,5)$ e $(5,6)$ non vengono toccate, in quanto le posizioni a loro corrispondenti in `tabella.txt` contengono la stringa `0`.

Il valore della biglia nella casella $(4,5)$ viene impostato a 5, quello della biglia nella casella $(5,5)$ a 4, quello della biglia nella casella $(4,6)$ a 7, infine 11 è il valore assegnato alla biglia nella casella $(6,6)$.

4. **Formato per la stampa di un blocco**

   Siano $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ tutte e sole le caselle formanti un blocco da stampare di colore σ, elencate in un ordine arbitrario. Allora il blocco deve essere stampato come segue:

   (σ
   x_1, y_1
   x_2, y_2
   $\vdots$
   x_k, y_k
)

Esempio

Si supponga che le righe di input siano:

```
i 6 3 0 0 f1.txt
i 4 5 -4 -2 f2.txt
i 6 3 0 -6 f1.txt
b 0 -1 6 blu
n
i 1 5 -4 2 f3.txt
i 1 5 -5 -3 f3.txt
b 0 -4 2 giallo
n
s 0 1
r 0 1
< rosso bianco
< bianco blu
< blu rosso
s 0 1
F 0 0 0 1
```

```
s 0 1
r 0 1
b 0 -1 8 blu
F 0 -1 0 -2
F 1 -3 -3 -1
r 0 -1
n
r 0 2
< giallo azzurro
< azzurro bianco
F -2 -1 -2 -2
n
b 1 -2 12 blu
b -4 -4 3 blu
r -3 0
i 1 5 -6 1 f3.txt
F -1 1 -6 1
F 1 3 2 5
r 1 3
r 1 3
n
r -1 1
r -6 -3
r -6 -3
r -6 1
r -6 1
r -6 1
n
f
```

dove il file `f1.txt` contiene

```
blu,7 blu,3 0,0
0,0 blu,1 giallo,4
0,0 giallo,6 giallo,1
giallo,6 rosso,1 giallo,2
0,0 rosso,2 0,0
0,0 blu,2 blu,8
```

il file `f2.txt` contiene

```
giallo,3 giallo,2 giallo,1 blu,2 rosso,2
0,0 rosso,4 rosso,7 blu,6 0,0
0,0 0,0 rosso,1 0,0 rosso,9
giallo,1 0,0 giallo,3 0,0 blu,1
```

mentre il file `f3.txt` contiene

```
giallo,1 giallo,1 giallo,1 giallo,1 giallo,1
```

L'output prodotto dal programma deve essere il seguente

```
14
9
( blu
0,1
1,1
1,0
-1,-1
0,-1
1,-1
2,-1
-1,-2
)
16
Non posso inserire blu < rosso.
( blu
-1,1
0,1
)
( blu
-1,1
0,1
0,0
1,0
-1,-1
0,-1
)
12
18
9
20
5
38
8
5
1
37
6
1
3
2
1
0
```

Presentazione del progetto

Il progetto deve essere inviato per posta elettronica all'indirizzo `aguzzoli@dsi.unimi.it` entro l'11 Marzo 2008 (incluso). 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 l'11 Marzo 2008 (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.

La discussione del progetto e l'esame orale di Algoritmi e Strutture Dati si svolgeranno indicativamente nei giorni 14, 18 e 19 Marzo 2008.

Alla consegna del progetto, indicare nel testo della e-mail la data in cui si preferisce sostenere la prova orale; nei limiti del possibile si cercherà di tener conto di tali indicazioni (se non si hanno preferenze, non dare alcuna indicazione).

Il calendario degli esami orali sarà disponibile sulla pagina del corso 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.

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.