Laboratorio di Algoritmi e Strutture Dati

Docenti: M. Torelli, S. Aguzzoli

Appello del 4 settembre 2008

Progetto “Percorsi colorati”
Consegna entro il 24 settembre 2008

Il problema

L’obiettivo è quello di studiare i percorsi che degli agenti intelligenti effettuano su un piano composto da celle colorate.

Formalmente, il piano è definito dall’insieme delle coppie di interi. Ogni coppia di interi (x, y) identifica una *cella*. Data una cella (x, y) diciamo che:

- la cella $(x, y + 1)$ è a *nord* di (x, y);
- la cella $(x, y - 1)$ è a *sud* di (x, y);
- la cella $(x + 1, y)$ è a *est* di (x, y);
- la cella $(x - 1, y)$ è a *ovest* di (x, y).

Diciamo inoltre che nord e sud sono opposti, est e ovest sono opposti.

Un *colore* è una stringa di lunghezza arbitraria sull’alfabeto $a, b, \ldots, z$ delle lettere minuscole. All’inizio dell’esecuzione del programma, ogni cella del piano ha colore *bianco*; è possibile modificare il colore di una cella tramite un apposito comando. Scriviamo $Col(x, y)$ per denotare il colore della cella (x, y).

Un *agente* è identificato da un *nome*, che è una stringa di lunghezza arbitraria sull’alfabeto $a, b, \ldots, z$ delle lettere minuscole. Ad ogni agente α è associato un insieme di coppie di colori $\mathcal{R}_\alpha$. Due colori σ e σ' sono *equivalenti* per α ($\sigma \equiv_\alpha \sigma'$) se e solo se esiste una sequenza di colori $\gamma_1, \ldots, \gamma_n$ ($n \geq 1$) tali che:

- $\sigma = \gamma_1$ e $\sigma' = \gamma_n$;
- per ogni i tale che $1 \leq i \leq n - 1$, $(\gamma_i, \gamma_{i+1}) \in \mathcal{R}_\alpha$ oppure $(\gamma_{i+1}, \gamma_i) \in \mathcal{R}_\alpha$.

In altri termini, $\equiv_\alpha$ è la più piccola relazione di equivalenza contenente la relazione $\mathcal{R}_\alpha$. Si noti, che per ogni colore σ, vale $\sigma \equiv_\alpha \sigma$.

È possibile aggiungere a $\mathcal{R}_\alpha$ nuove coppie di colori tramite un apposito comando.

Esempio 1

Supponiamo che l’insieme $\mathcal{R}_\alpha$ sia definito come segue:

$$\mathcal{R}_\alpha \; = \; \{\, (giallo, verde), \;\; (rosso, blu), \;\; (blu, nero), \;\; (verde, azzurro)\,\}$$

Valgono, ad esempio, le seguenti relazioni:

$$verde \equiv_\alpha giallo\,, \qquad azzurro \equiv_\alpha giallo\,, \qquad viola \equiv_\alpha viola\,.$$

Rispetto a $\equiv_\alpha$, *rosso* e *verde* non sono equivalenti, e neppure *azzurro* e *rosso*.

Un agente effettua una *mossa* in direzione *Dir* (*Dir* $\in \{nord, sud, est, ovest\}$) se, dalla cella (x, y) in cui si trova, si sposta nella cella (x', y') in posizione *Dir* rispetto a (x, y). Un *percorso* π da (x, y) a (x', y') per un agente α è una sequenza di celle $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ $(n \geq 1)$ tali che:

1. $(x, y) = (x_1, y_1)$ e $(x', y') = (x_n, y_n)$.
2. Per ogni k tale che $1 \leq k \leq n - 1$, α può passare da (x_k, y_k) a (x_{k+1}, y_{k+1}) in una mossa.
3. Se α effettua una mossa in direzione *Dir* passando da una cella alla successiva, allora α non potrà più effettuare mosse in direzione opposta a *Dir*.

   Ad esempio, se a un certo punto α si muove verso ovest, non potrà più muoversi verso est.
4. Per ogni k tale che $1 \leq k \leq n - 1$, $Col(x_k, y_k) \equiv_\alpha Col(x_{k+1}, y_{k+1})$, oppure almeno una cella tra (x_k, y_k) e (x_{k+1}, y_{k+1}) ha colore *bianco*.

L'agente consuma energia quando passa da una cella a una di colore diverso, quindi, nella pianificazione di un percorso, deve cercare di minimizzare i cambi di colore. Formalmente, diciamo che una mossa da (x, y) a (x', y') ha costo 0 se $Col(x, y) = Col(x', y')$, ha costo 1 se $Col(x, y) \neq Col(x', y')$. Il *costo*

$$Costo(\pi)$$

di un percorso π è dato dalla somma dei costi delle mosse che l'agente effettua nel percorrere π. Ad esempio, se i colori delle celle che formano π sono

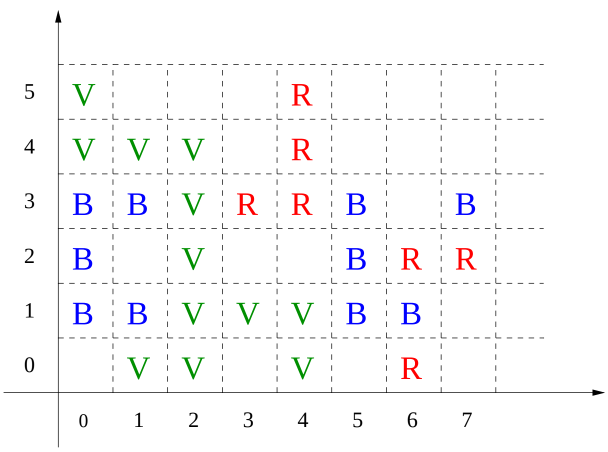
$$giallo, bianco, bianco, verde, verde, verde, rosso, rosso$$

il costo di π è 3.

Diciamo che un percorso π dalla cella (x, y) alla cella (x', y') ha *costo ottimo* per α se, per ogni altro percorso π' da (x, y) a (x', y') per α, $Costo(\pi') \geq Costo(\pi)$.

Esempio 2

Supponiamo che le celle del piano siano colorate come nella figura qui sotto, dove le celle contrassegnate da `B` hanno colore *blu*, quelle con `R` hanno colore *rosso*, quelle con `V` hanno colore *verde*, tutte le altre hanno colore *bianco*.



Consideriamo l'agente α tale che $\mathcal{R}_\alpha = \emptyset$. Un percorso di costo ottimo da $(0,5)$ a $(5,0)$ per α è

$$(0,5),\ (0,4),\ (1,4),\ (2,4),\ (2,3),\ (2,2),\ (2,1),\ (3,1),\ (4,1),\ (4,0),\ (5,0)$$

di costo 1 (il costo è ottimo in quanto per α non esistono percorsi da $(0,5)$ a $(5,0)$ di costo 0). Un percorso di costo ottimo da $(1,0)$ a $(0,3)$ per α è

$$(1,0),\ (0,0),\ (0,1),\ (0,2),\ (0,3)$$

di costo 2. Non esiste invece per α alcun percorso da $(1,0)$ a $(0,4)$.
Consideriamo ora l'agente β tale che

$$\mathcal{R}_\beta \ = \ \{\,(blu, verde)\,\}$$

Un percorso di costo ottimo da $(1,2)$ a $(7,0)$ per β è

$$(1,2),\ (2,2),\ (2,1),\ (3,1),\ (4,1),\ (5,1),\ (6,1),\ (7,1),\ (7,0)$$

di costo 3. Un percorso di costo ottimo da $(1,2)$ a $(1000,0)$ per β è

$$(1,2),\ (2,2),\ (2,1),\ (3,1),\ (4,1),\ (5,1),\ (6,1),\ (7,1),\ (7,0),\ (8,0),\ (9,0),\ \ldots\ (1000,0)$$

di costo 3. Per β non esiste alcun percorso da $(0,3)$ a $(8,2)$.
Infine, consideriamo ora l'agente γ tale che

$$\mathcal{R}_\gamma \ = \ \{\,(blu, verde),\ (rosso, blu)\,\}$$

(si noti che *verde* $\equiv_\gamma$ *rosso*). Un percorso di costo ottimo da $(0,0)$ a $(4,5)$ per γ è

$$(0,0),\ (1,0),\ (2,0),\ (2,1),\ (2,2),\ (2,3),\ (3,3),\ (4,3),\ (4,4),\ (4,5)$$

di costo 2.

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 celle colorate (non bianche) può essere molto grande rispetto al numero di celle colorate presenti nel piano, quindi *non è sicuramente efficiente rappresentare l'insieme delle celle colorate mediante un'*unica *matrice*).

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

  Legge dal file $nomefile$ una tabella di colori di r righe e c colonne e colora di conseguenza le celle dell'insieme $\{(x+h, y+k) \mid 0 \leq h < c,\ 0 \leq k < r\}$ secondo le regole specificate nell'apposita sezione.

- **colore** (x, y, σ)

  Assegna alla cella (x, y) il colore σ.

- **aggiungi** $(\alpha, \sigma, \sigma')$

  Aggiunge alla relazione $\mathcal{R}_\alpha$ la coppia di colori (σ, σ').

- **equivalente** $(\alpha, \sigma, \sigma')$

  Se $\sigma \equiv_\alpha \sigma'$, stampa

  α `:` σ `=` σ'

  Altrimenti, stampa

  α `:` σ `!=` σ'

- **unione** (α, β)

  Aggiunge tutte le coppie di $\mathcal{R}_\beta$ a $\mathcal{R}_\alpha$.

- **percorso** (α, x, y, x', y')

  Calcola un percorso di costo ottimo da (x, y) a (x', y') per α. Se non esiste alcun percorso da (x, y) a (x', y') per α stampa il messaggio:

  `Non esiste percorso da` (x, y) `a` (x', y') `per` α

  Altrimenti, sia $\pi = (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ un percorso di costo ottimo da (x, y) a (x', y') per α e sia $c = Costo(\pi)$. Allora l'output del comando deve essere:

  (c
  x_1, y_1
  x_2, y_2
  $\vdots$
  x_n, y_n
)

Si noti che le operazioni richieste sono liberamente implementabili; in particolare, non vanno necessariamente intese come prototipi di funzioni. Si ricorda inoltre che, all'inizio dell'esecuzione del programma, tutte le celle del piano hanno colore *bianco* e, per ogni agente α, l'insieme $\mathcal{R}_\alpha$ è vuoto.

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, dove $nomefile$ è il nome di un file, α, β, σ e σ' sono stringhe finite sull'alfabeto $\{a, b, \dots, z\}$ di lunghezza *arbitraria* (ossia, non deve essere definita nel programma alcuna costante k che limiti la lunghezza di una stringa a k); x, x', y e y' sono interi e r e c sono interi positivi. 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** $(r, c, x, y, nomefile)$
`c` $x\ y\ \sigma$	**colore** (x, y, σ)
`a` $\alpha\ \sigma\ \sigma'$	**aggiungi** $(\alpha, \sigma, \sigma')$
`e` $\alpha\ \sigma\ \sigma'$	**equivalente** $(\alpha, \sigma, \sigma')$
`u` $\alpha\ \beta$	**unione** (α, β)
`p` $\alpha\ x\ y\ x'\ y'$	**percorso** (α, x, y, x', y')
`f`	Termina l'esecuzione del programma

Tabella 1: Specifiche del programma

Note

1. Non devono essere presenti vincoli sul numero di celle colorate, di colori e di agenti sulla collocazione delle celle nel piano, sulla lunghezza dei nomi di colori e di agenti. 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 colori da un file**

   Si consideri il comando:

   `i` r c x y $nomefile$.

   Allora il file di nome $nomefile$ contiene una sequenza di $r \times c$ stringhe, ciascuna delle quali è una stringa finita sull'alfabeto $\{a, b, \ldots, z\}$ oppure `0`. I colori delle celle nel piano vengono modificati in questo modo: Per ogni h, k tali che $0 \leq h < c$ e $0 \leq k < r$, se la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h + 1)$ nel file è diversa da `0` allora si pone in $(x + h, y + k)$ il colore $\alpha_{h,k}$; se invece la stringa $\alpha_{h,k}$ in posizione $k \cdot c + (h + 1)$ nel file è uguale a `0` non si modifica il colore 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 rosso 0
   bianco 0 nero
   ```

   allora i colori delle celle vengono modificati in questo modo:

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

   Nessun'altra cella 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`.

Esempio

Si supponga che le righe di input siano:

```
i 2 5 0 0 f1.txt
i 5 2 5 -1 f2.txt
i 2 2 7 -1 f3.txt
c 1 2 verde
p anna 0 0 2 0
a bruno rosso azzurro
p anna 0 0 2 0
a anna azzurro giallo
p anna 0 0 2 0
u anna bruno
p anna 0 0 2 0
p bruno 0 3 5 -1
c 3 2 azzurro
c 2 -1 azzurro
```

```
c 2 2 azzurro
c 5 1 porpora
p bruno 0 3 5 -1
p anna 0 3 5 -1
a anna nero giallo
a bruno nero giallo
e anna giallo rosso
e bruno giallo rosso
c 7 1 marrone
c 7 3 marrone
a anna marrone verde
c 4 3 azzurro
p anna 0 0 7 3
a carlo verde azzurro
p carlo 8 -2 0 3
u carlo bruno
e carlo rosso verde
e dario rosso verde
p carlo 8 -2 0 3
a carlo marrone rosso
p carlo 10 3 -1 0
p dario 1 1 3 -1
f
```

dove il file `f1.txt` contiene

```
rosso giallo rosso giallo verde
verde giallo giallo giallo rosso
```

il file `f2.txt` contiene

```
0 rosso
rosso blu
rosso blu
rosso verde
verde verde
```

e il file `f3.txt` contiene

```
blu 0
rosso verde
```

L'output prodotto dal programma deve essere il seguente

```
Non esiste percorso da (0,0) a (2,0) per anna
Non esiste percorso da (0,0) a (2,0) per anna
Non esiste percorso da (0,0) a (2,0) per anna
( 2
0, 0
1, 0
```

```
2, 0
)
( 2
0, 3
1, 3
2, 3
3, 3
3, 2
3, 1
3, 0
3, -1
4, -1
5, -1
)
Non esiste percorso da (0,3) a (5,-1) per bruno
( 3
0, 3
1, 3
2, 3
3, 3
3, 2
3, 1
3, 0
3, -1
4, -1
5, -1
)
anna: giallo = rosso
bruno: giallo != rosso
Non esiste percorso da (0,0) a (7,3) per anna
( 5
8, -2
8, -1
8, 0
8, 1
8, 2
7, 2
6, 2
6, 3
5, 3
4, 3
3, 3
2, 3
1, 3
0, 3
)
carlo: rosso = verde
dario: rosso != verde
( 3
```

```
8, -2
7, -2
6, -2
5, -2
4, -2
3, -2
2, -2
1, -2
0, -2
0, -1
0, 0
0, 1
0, 2
0, 3
)
( 4
10, 3
9, 3
8, 3
7, 3
6, 3
5, 3
4, 3
3, 3
2, 3
1, 3
0, 3
-1, 3
-1, 2
-1, 1
-1, 0
)
( 1
1, 1
2, 1
3, 1
3, 0
3, -1
)
```

Presentazione del progetto

Il progetto deve essere inviato per posta elettronica all'indirizzo `aguzzoli@dsi.unimi.it` entro il 24 settembre 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 il 24 settembre 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 30 settembre, 3 e 7 ottobre 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.