# Laboratorio di Algoritmi e Strutture Dati

Docenti: M. Torelli, S. Aguzzoli

Appello del 12 giugno 2007

Progetto "Domino"
Consegna entro il 1º luglio 2007

## Il problema

Il gioco del domino consiste nel disporre una sequenza di tessere, ciascuna delle quali suddivisa in due parti aventi colori distinti, in modo che i lati a contatto di due tessere adiacenti abbiano lo stesso colore.

Formalmente, sia $\mathcal{A}$ l'insieme delle lettere minuscole dell'alfabeto. Una *tessera* $t$ è definita da una tripla $(\alpha, \beta, \sigma)$, dove $\alpha$, $\beta$ e $\sigma$ sono parole di lunghezza *arbitraria* sull'alfabeto $\mathcal{A}$; $\alpha$ e $\beta$ rappresentano i colori con cui è colorata ciascuna delle due metà di $t$, $\sigma$ il nome di $t$. Per ottenere un domino, possiamo immaginare di disporre le tessere in una sequenza. Ogni tessera può essere disposta in due modi diversi. Denotiamo una *disposizione* di una tessera $t$ con la coppia $(t, k)$, con $k \in \{-1, +1\}$. Data una tessera $t = (\alpha, \beta, \sigma)$ e una sua disposizione $(t, k)$, il *colore sinistro* $Col_S$ di $(t, k)$ e il *colore destro* $Col_D$ di $(t, k)$ sono definiti nel modo seguente, a seconda del valore di $k$:

- $Col_S((t, 1)) = \alpha$ e $Col_D((t, 1)) = \beta$;
- $Col_S((t, -1)) = \beta$ e $Col_D((t, -1)) = \alpha$.

Un *domino* $\tau$ è dato da una sequenza di disposizioni di tessere

$$(t_1, k_1),\ (t_2, k_2),\ \ldots,\ (t_n, k_n)$$

tali che:

- per ogni $1 \leq i \leq n$, la tessera $t_i$ non appartiene ad alcun altro domino precedentemente definito;
- per ogni $1 \leq i \leq n$ e $1 \leq j \leq n$, $i \neq j$ implica $t_i \neq t_j$;
- per ogni $1 \leq i \leq n - 1$, $Col_D((t_i, k_i)) = Col_S((t_{i+1}, k_{i+1}))$.

La *lunghezza* di $\tau$ è $n$ (numero di tessere usate in $\tau$). Se $Col_S((t_1, k_1)) = \alpha$ e $Col_D((t_n, k_n)) = \beta$, diciamo che $\tau$ è un domino dal colore $\alpha$ al colore $\beta$. Un domino *minimo* da $\alpha$ a $\beta$ è un domino da $\alpha$ a $\beta$ avente lunghezza minima fra tutti i possibili domini da $\alpha$ a $\beta$.

### Esempio

Siano date le seguenti tessere:

$$t_1 = (rosso, verde, pane),\ t_2 = (rosso, blu, cane),\ t_3 = (giallo, blu, casa),\ t_4 = (verde, giallo, cosa)$$

Allora i domini da *rosso* a *giallo* sono i seguenti:

$$(t_1, 1), (t_4, 1) \quad \text{e} \quad (t_2, 1), (t_3, -1)$$

entrambi di lunghezza minima 2. I domini da *rosso* a *verde* sono:

$$(t_1, 1) \quad \text{e} \quad (t_2, 1), (t_3, -1), (t_4, -1)$$

e il domino $(t_1, 1)$ che ha lunghezza 1, è l'unico domino da *rosso* a *verde* di lunghezza minima.

Si consideri un domino $(t_1, k_1)$, $(t_2, k_2)$, $\ldots$, $(t_n, k_n)$ e si denoti con $\sigma_i$ il nome della tessera $t_i$, per ogni $i = 1, \ldots, n$. Il *nome* del domino $(t_1, k_1)$, $(t_2, k_2)$, $\ldots$, $(t_n, k_n)$ è la stringa $\sigma_1\sigma_2 \cdots \sigma_n$ ottenuta concatenando i nomi delle singole tessere. La lettura del nome $\sigma_1\sigma_2 \cdots \sigma_n$ è tanto più *cacofonica* quanto più i nomi delle tessere adiacenti sono simili tra loro.

Formalmente, date due stringhe $\sigma_1 = a_1a_2 \cdots a_u$ e $\sigma_2 = b_1b_2 \cdots b_v$, diciamo che $\sigma = c_1c_2 \cdots c_z$ è una *sottostringa* di $\sigma_1$ e $\sigma_2$ se $\sigma$ verifica le seguenti proprietà:

1. Per ogni $h = 1, \ldots, z$ esistono indici $i(h)$ e $j(h)$ con $1 \leq i(h) \leq u$ e $1 \leq j(h) \leq v$ tali che $c_h = a_{i(h)} = b_{j(h)}$.
2. Per ogni coppia di indici $h_1, h_2$ con $1 \leq h_1 < h_2 \leq z$ si ha che $i(h_1) < i(h_2)$ e $j(h_1) < j(h_2)$.

Diciamo che $\sigma$ è una *sottostringa massima* di $\sigma_1$ e $\sigma_2$ se $\sigma$ è una sottostringa di $\sigma_1$ e $\sigma_2$ avente lunghezza massima fra tutte le sottostringhe di $\sigma_1$ e $\sigma_2$. Si noti che due stringhe possono ammettere più di una sottostringa massima, tutte della stessa lunghezza. Ad esempio, le sottostringhe massime di *abc* e *bac* sono *ac* e *bc*, di lunghezza 2. Con $\delta(\sigma_1, \sigma_2)$ denotiamo la lunghezza di una sottostringa massima di $\sigma_1$ e $\sigma_2$. Sia $\tau$ un domino e sia $\sigma_1\sigma_2 \cdots \sigma_n$ il suo nome; la *cacofonia* di $\tau$ è data da

$$\sum_{i=1}^{n-1} \delta(\sigma_i, \sigma_{i+1}) \,.$$

**Esempio**

Il nome del domino $(t_2, 1), (t_3, -1), (t_4, -1)$, costruito nell'esempio precedente, è *canecasacosa*. Si noti che $\delta(cane, casa) = 2$ (l'unica sottostringa massima è *ca*), mentre $\delta(casa, cosa) = 3$ (l'unica sottostringa massima è *csa*), dunque la cacofonia del domino risulta essere $2 + 3 = 5$.

Si richiede di implementare una struttura dati efficiente che permetta di eseguire le operazioni seguenti:

- **tessera** $(\alpha, \beta, \sigma)$

  Se esiste già una tessera di nome $\sigma$ oppure se $\alpha$ è uguale a $\beta$, non compie alcuna operazione. Altrimenti, crea la tessera $(\alpha, \beta, \sigma)$.

- **elimina**$(\sigma)$

  Elimina, se esiste, la tessera di nome $\sigma$.

- **domino**$(\alpha, \beta)$

  Crea un domino di lunghezza minima dal colore $\alpha$ al colore $\beta$. Se non è possible crearlo, stampa il messaggio:

  `non esiste domino da` $\alpha$ `a` $\beta$

- **stampaDomino**$(\sigma)$

  Se non esiste alcuna tessera di nome $\sigma$ oppure se la tessera di nome $\sigma$ non appartiene ad alcun domino, non compie alcuna operazione.

  Altrimenti, stampa il domino cui appartiene la tessera $\sigma$ secondo il formato specificato nell'apposita sezione.

- **cancellaDomino**($\sigma$)

  Se non esiste alcuna tessera di nome $\sigma$ oppure se la tessera di nome $\sigma$ non appartiene ad alcun domino, non compie alcuna operazione.

  Altrimenti, sia $\delta$ il domino cui appartiene la tessera di nome $\sigma$. Allora, $\delta$ è cancellato e tutte le tessere che compongono $\delta$ possono essere riutilizzate nella definizione di nuovi domini.

- **max**($\sigma_1, \sigma_2$)

  Se almeno una tra $\sigma_1$ e $\sigma_2$ non è un nome di tessera, non compie alcuna operazione. Altrimenti, stampa su una nuova linea una sottostringa massima $\sigma$ di $\sigma_1$ e $\sigma_2$ (se $\sigma$ è la stringa nulla, la linea rimane vuota).

- **cacofonia**($\sigma$)

  Se non esiste alcuna tessera di nome $\sigma$ oppure se la tessera di nome $\sigma$ non appartiene ad alcun domino, non compie alcuna operazione.

  Altrimenti stampa la cacofonia del domino cui appartiene la tessera di nome $\sigma$.

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 $\alpha$, $\beta$ e $\sigma$ sono stringhe finite sull'alfabeto $\mathcal{A}$ di lunghezza *arbitraria*. 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 |
|---|---|
| `t` $\alpha\ \beta\ \sigma$ | **tessera** $(\alpha, \beta, \sigma)$ |
| `e` $\sigma$ | **elimina** $(\sigma)$ |
| `d` $\alpha\ \beta$ | **domino** $(\alpha, \beta)$ |
| `s` $\sigma$ | **stampaDomino** $(\sigma)$ |
| `c` $\sigma$ | **cancellaDomino** $(\sigma)$ |
| `m` $\alpha\ \beta$ | **max** $(\alpha, \beta)$ |
| `C` $\sigma$ | **cacofonia** $(\sigma)$ |
| `f` | Termina l'esecuzione del programma |

Tabella 1: Specifiche del programma

**Note**

1. Non devono essere presenti vincoli sul numero di tessere e domini esistenti, e sulla lunghezza dei colori e dei nomi di tessere (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. **Formato per la stampa di un domino**

   Si supponga che il domino da stampare sia

   $$(t_1, k_1), (t_2, k_2), \ldots, (t_n, k_n)$$

   e che le tessere che lo costituiscono siano $t_i = (\alpha_i, \beta_i, \sigma_i)$ per ogni $1 \leq i \leq n$. Si ponga

   $$\gamma_i = \sigma_i : \alpha_i, \beta_i \qquad \text{se} \qquad k_i = +1\,,$$

   $$\gamma_i = \sigma_i : \beta_i, \alpha_i \qquad \text{se} \qquad k_i = -1\,.$$

   L'output da produrre è il seguente:

   (
   $\gamma_1$
   $\gamma_2$
   $\vdots$
   $\gamma_n$
   )

**Esempio**
Si supponga che le righe di input siano:

```
t nero rosso cammello
t celeste verde topolino
t arancione blu elefante
t celeste arancione cane
t nero rosa gatto
t giallo arancione cavallo
t nero giallo cappello
t blu rosso mela
t verde blu topo
t rosa verde tela
t verde celeste tavolo
m mela tela
m cappello cammello
m cane topo
m topo topolino
m gatto  topo
m tela elefante
```

```
d arancione nero
s cavallo
C cappello
d arancione nero
s elefante
C mela
e gatto
d blu nero
c cammello
d blu nero
s mela
C cammello
c mela
c cappello
t rosa nero pesca
d verde verde
s tavolo
C topolino
d verde verde
s cammello
C pesca
f
```

L'output prodotto dal programma deve essere:

```
ela
caello

topo
to
ela
(
cavallo: arancione, giallo
cappello: giallo, nero
)
5
(
elefante: arancione, blu
mela: blu, rosso
cammello: rosso, nero
)
6
non esiste domino da blu a nero
(
mela: blu, rosso
cammello: rosso, nero
)
3
(
topolino: verde, celeste
```

```
tavolo: celeste, verde
)
4
(
tela: verde, rosa
pesca: rosa, nero
cammello: nero, rosso
mela: rosso, blu
topo: blu, verde
)
7
```

## Presentazione del progetto

Il progetto deve essere inviato per posta elettronica all'indirizzo `aguzzoli@dsi.unimi.it` entro il 1° luglio 2007 (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/~torelli/algoritmi.html`).

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 2 luglio 2007 (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 lesame orale di Algoritmi e Strutture Dati si svolgeranno indicativamente nei giorni 3, 10, 13 luglio 2007.

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.