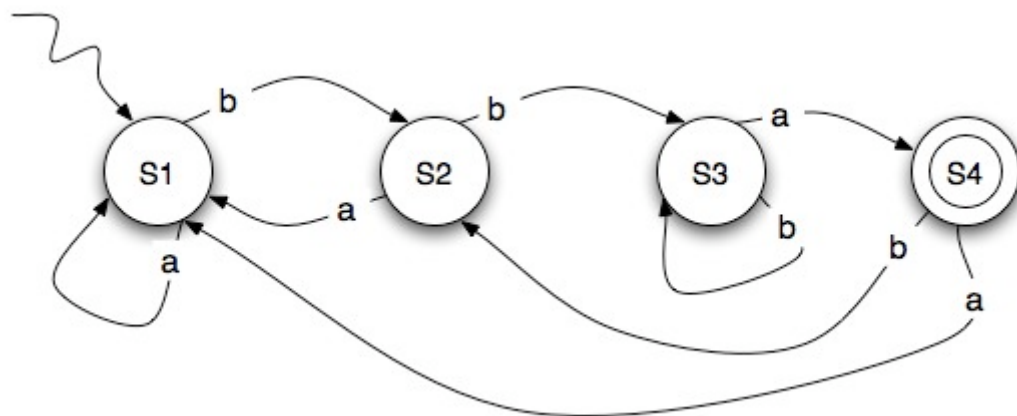


## MCC: esercitazione del 17 gennaio 2008

### Esercizio 1

Scrivere l'ASF che riconosce il linguaggio:  $(a+b)^*bba$

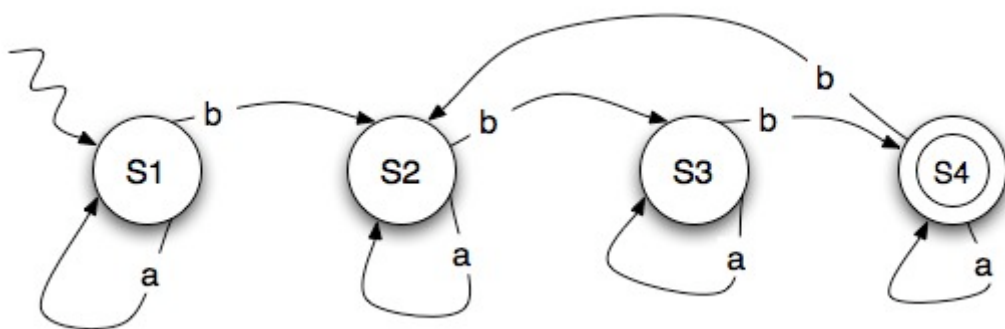
Soluzione:



### Esercizio 2

Scrivere l'ASF che riconosce il linguaggio:  $(a^*ba^*ba^*ba^*)^+$

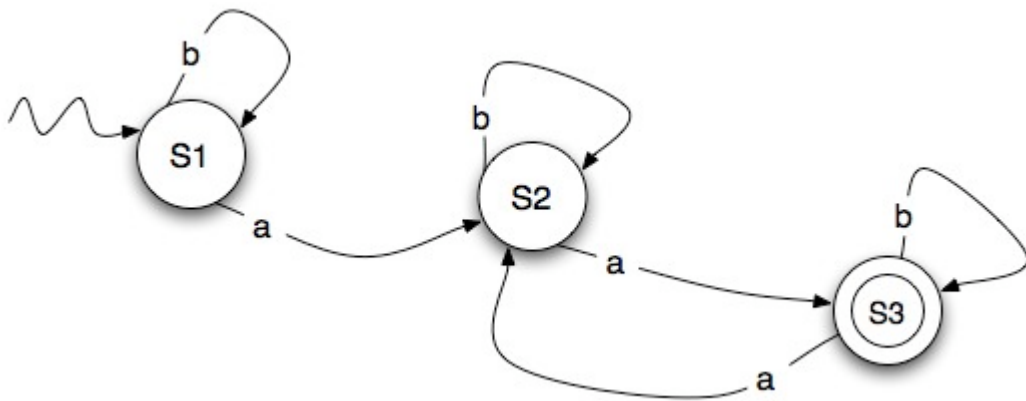
Soluzione:



### Esercizio 3

Scrivere l'ASF che riconosce il linguaggio:  $(b^*ab^*ab^*)^+$

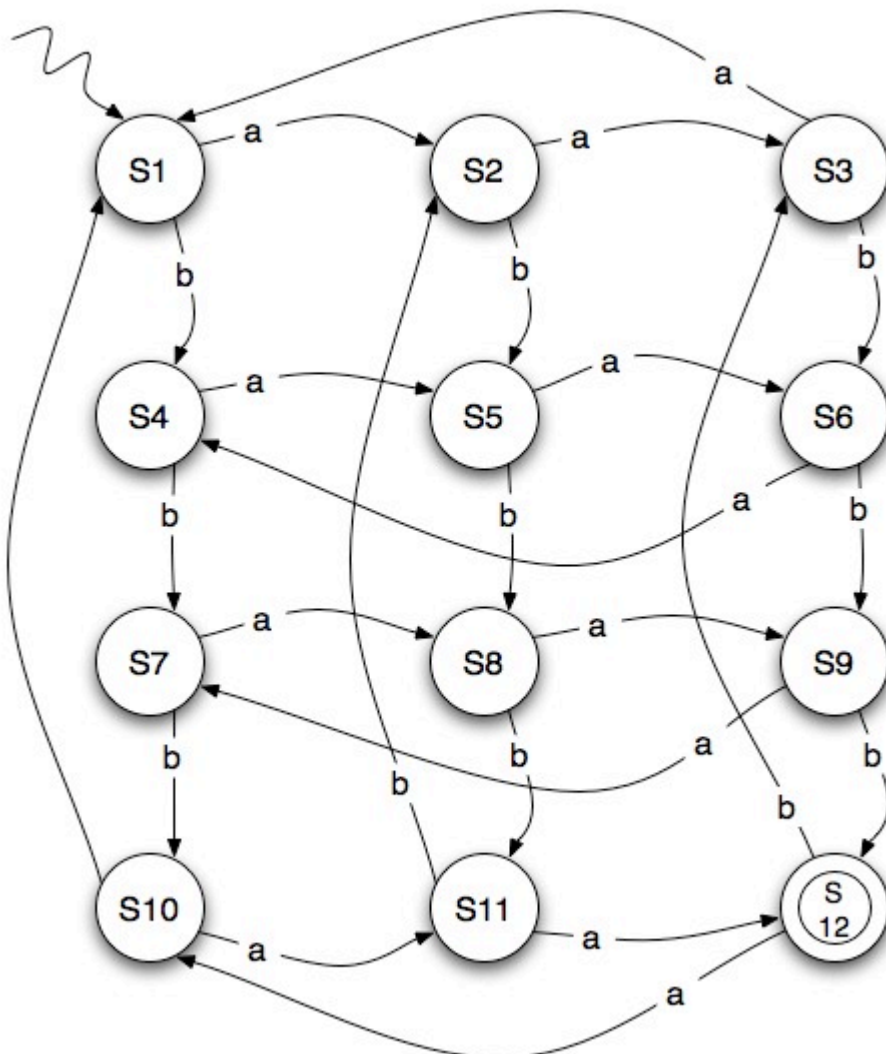
Soluzione:



### Esercizio 4

Scrivere l'ASF che riconosce il linguaggio L composto da tutte e sole le stringhe  $w$  sull'alfabeto  $\{a, b\}$  aventi un numero di 'a' pari a  $2k$  ( $k \geq 1$ ) ed un numero di 'b' pari a  $3j$  ( $j \geq 1$ ):  $L = \{w \mid \#(a,w) = 2k \text{ (} k \geq 1 \text{) e } \#(b,w) = 3j \text{ (} j \geq 1 \text{)}\}$

Soluzione banale (12 stati):



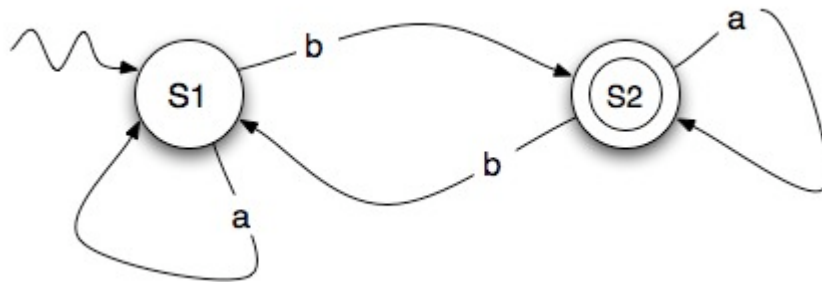
### Esercizio 4/bis

Trovare una soluzione con un numero inferiore di stati

### Esercizio 5

Scrivere la grammatica che genera il linguaggio, sull'alfabeto  $\{a,b\}$ , delle stringhe aventi un numero dispari di 'b'.

Soluzione: si parte dall'ASF



Dall'automa a stati finiti è relativamente semplice ricavare la grammatica corrispondente. Regola pratica: ad ogni stato corrisponde un simbolo NON TERMINALE; alle etichette di un ramo corrisponde un simbolo TERMINALE; lo stato iniziale è l'assioma della grammatica; ad ogni ramo con etichetta 'x' che collega lo stato A allo stato B corrisponde la produzione:  $A \rightarrow xB$ . Se lo stato B è anche uno stato finale, si aggiunge la produzione  $A \rightarrow x$ .

Quindi la grammatica corrispondente al precedente automa è:

$S1 \rightarrow aS1 \mid bS2 \mid b$

$S2 \rightarrow aS2 \mid bS1 \mid a$

## Esercizio 6

Scrivere le grammatiche che generano i linguaggi degli esercizi 1 e 2.

### Grammatica esercizio 1:

$S_1 \rightarrow aS_1 \mid bS_2$

$S_2 \rightarrow aS_1 \mid bS_3$

$S_3 \rightarrow aS_4 \mid a \mid bS_3$

$S_4 \rightarrow aS_1 \mid bS_2$

### Grammatica esercizio 2:

$S_1 \rightarrow aS_1 \mid bS_2$

$S_2 \rightarrow aS_2 \mid bS_3$

$S_3 \rightarrow aS_3 \mid bS_4 \mid b$

$S_4 \rightarrow aS_4 \mid a \mid bS_2$

## MCC: esercitazione del 30 gennaio 2008

Macchine di Turing deterministiche, non deterministiche, ad un solo nastro e con più nastri.

### Notazione

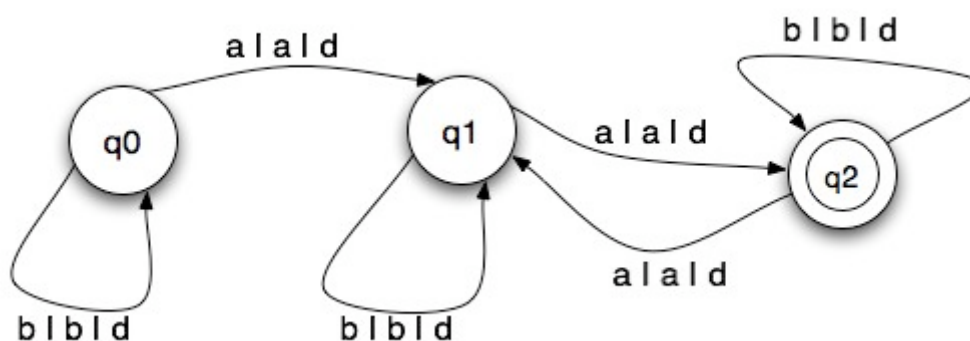
Rappresentiamo le MdT mediante grafi: i nodi rappresentano gli stati che la MdT può assumere; le transizioni da uno stato all'altro sono rappresentate da archi etichettati. Le etichette, per MdT a nastro singolo, sono triple del tipo "a|b|m", dove 'a' è il carattere letto sul nastro, 'b' il carattere scritto e 'm' indica lo spostamento della testina sul nastro ('d' per lo spostamento a destra, 's' per lo spostamento a sinistra, 'i' per nessuno spostamento).

Per le MdT ad n nastri usiamo delle etichette del tipo "a<sub>1</sub>, a<sub>2</sub>, ... , a<sub>n</sub> | b<sub>1</sub>, b<sub>2</sub>, ... , b<sub>n</sub> | m<sub>1</sub>, m<sub>2</sub>, ... , m<sub>n</sub>", dove 'a<sub>x</sub>' (x = 1 .. n) è il carattere letto sul nastro numero x, 'b<sub>x</sub>' è il carattere scritto sul nastro numero x, 'm<sub>x</sub>' è lo spostamento della testina sul nastro x.

### Esercizio 1

Scrivere la macchina di Turing deterministica ad un nastro che riconosce le stringhe  $w \in \{a, b\}^+$  che contengono un numero pari, non nullo, di 'a'.

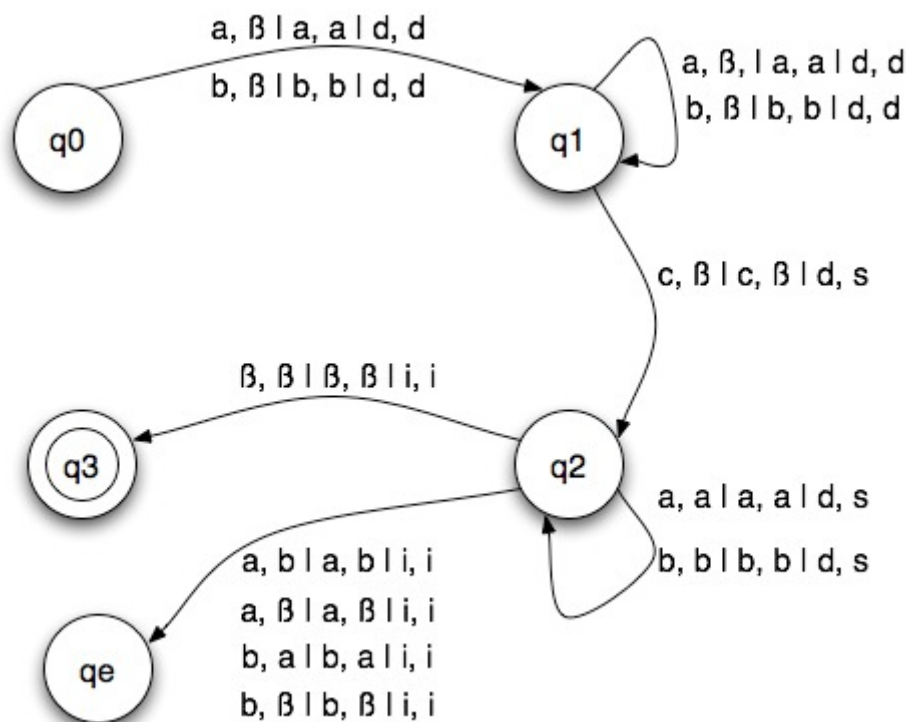
Soluzione:



### Esercizio 2

Scrivere la macchina di Turing deterministica a due nastri che riconosce le stringhe del tipo  $wcw'$  (dove  $w'$  è l'inversa di  $w$ ) con  $w \in \{a, b\}^+$ .

Soluzione: il primo nastro contiene la stringa di input, mentre il secondo è un nastro di lavoro contenente inizialmente solo simboli blank ( $\beta$ ).



Spiegazione: ogni volta che la macchina legge sul nastro di input una 'a' o una 'b', copia il carattere letto sul nastro di lavoro e quindi passa a destra (su entrambi i nastri). Dopo aver incontrato una 'c', inizia a percorrere il nastro di lavoro al contrario per confrontare il carattere letto (dopo la 'c') sul nastro di input con quello letto sul nastro di lavoro. Fin quando i caratteri sono uguali, procede (a destra sul nastro di input, a sinistra sul nastro di lavoro). Se incontra due caratteri differenti, la macchina passa in uno stato di errore; se incontra due 'β', termina passando nello stato finale. Lo stato di errore ( $q_e$ ) e le relative transizioni non sono strettamente necessarie in quanto un eventuale input non previsto provoca l'arresto della macchina su uno stato non finale. La MdT, infatti, accetta una data stringa solo se termina in uno stato finale.

Lo stato  $q_1$  è necessario in quanto senza di esso la MdT avrebbe accettato anche stringhe composte dalla sola lettera 'c'.

### Esercizio 3

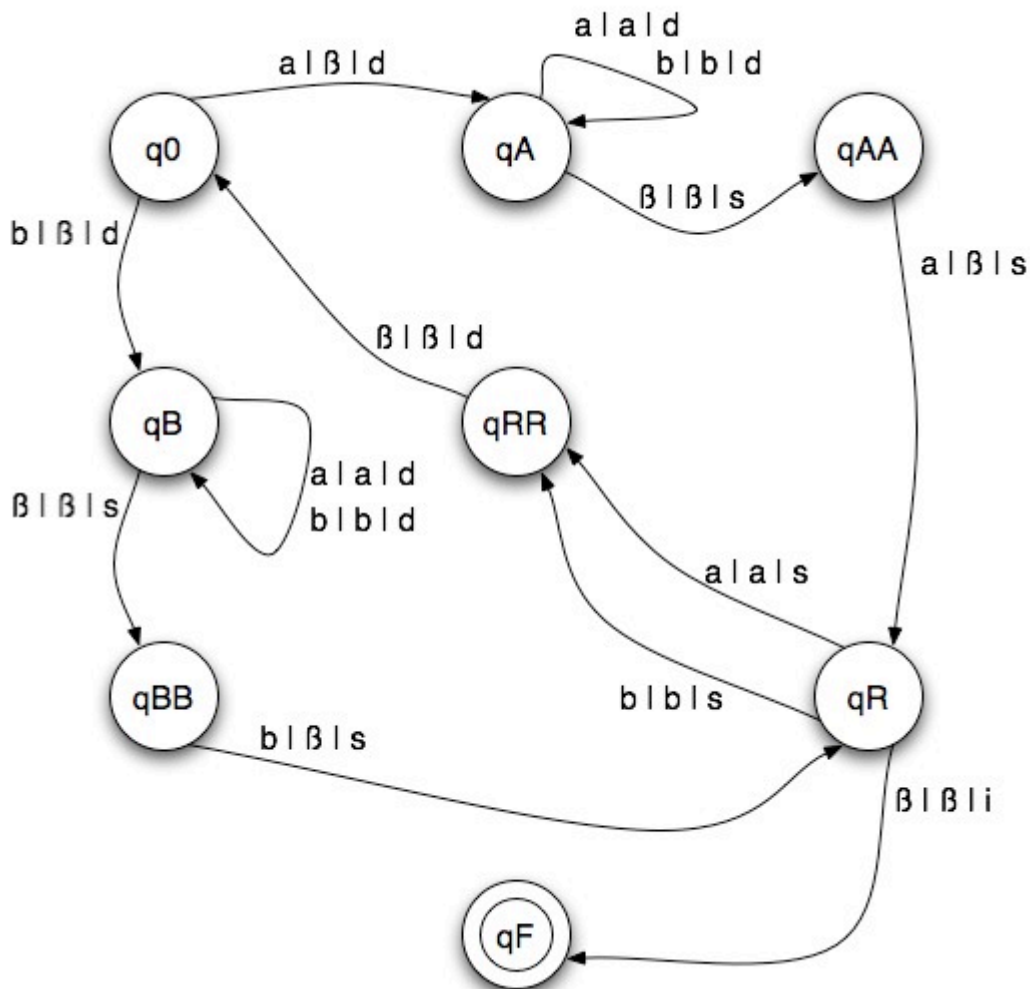
Scrivere la macchina di Turing deterministica ad un solo nastro che riconosce le stringhe del tipo  $ww'$ , con  $w \in \{a, b\}^+$  ( $w'$  rappresenta l'inversa di  $w$ ). Ci si può limitare a descrivere il funzionamento della macchina, senza scriverne la realizzazione completa.

Soluzione: l'idea è quella di avere uno stato iniziale  $q_0$ , uno stato  $q_A$  che ci ricorda che l'ultimo carattere letto è una 'a', uno stato  $q_B$  che ci ricorda che l'ultimo carattere letto è una 'b' e uno stato finale  $q_F$ . Dopo aver letto un carattere, lo cancello e passo nello stato  $q_A$  o  $q_B$  (in base al carattere letto); quindi si inizia a scandire tutta la stringa per verificare che l'ultimo carattere della stessa sia uguale a quello letto: in caso positivo, lo cancello e torno all'inizio per ripetere la computazione (se ci sono ancora caratteri nella stringa)

oppure termino nello stato di accettazione. Se invece l'ultimo carattere della stringa non corrisponde a quello letto, mi fermo (stringa rifiutata).

La realizzazione della macchina richiede alcuni stati in più.

Quella che segue è la mia realizzazione; probabilmente differisce da quella mostrata alla lavagna durante l'esercitazione.



Spiegazione: dallo stato iniziale, se leggo una 'a' la cancello e mi sposto a destra nello stato  $q_A$ ; analogamente, se leggo una 'b' la cancello e mi sposto a destra nello stato  $q_B$ . ' $q_A$ ' mi ricorda che ho letto una 'a', ' $q_B$ ' mi ricorda che ho letto una 'b'. Negli stati  $q_A$  e  $q_B$  non faccio altro che scorrere verso destra la stringa, senza modificarla (cioè riscrivendo sempre il carattere letto); quando incontro il simbolo  $\beta$  (blank), se ero nello stato  $q_A$  vado in  $q_{AA}$ , se ero in  $q_B$  vado in  $q_{BB}$ . In entrambi i casi mi sposto a sinistra sulla stringa, per poterne leggere l'ultimo carattere. ' $q_{AA}$ ' vuol dire che avevo letto una 'a' all'inizio e ora devo cercare una 'a' alla fine; analogamente, ' $q_{BB}$ ' vuol dire che avevo letto una 'b' all'inizio e ora devo cercare una 'b' alla fine. In  $q_{AA}$ , se leggo una 'a' i due estremi della stringa coincidono, quindi cancello il carattere, vado nello stato  $q_R$  e mi sposto a sinistra. Lo stesso accade se in  $q_{BB}$  leggo una 'b'.

' $q_R$ ' è lo stato di verifica della fine dell'input: se leggo un blank ho finito e vado nello stato finale, altrimenti vado in  $q_{RR}$  che è lo stato di riavvolgimento del

nastro (cioè torno indietro al primo carattere della stringa, che non è ancora vuota). Itero su qRR finché non trovo il blank all'inizio della stringa, quindi torno in q0 e ricomincio.

Avvertenza: potrei aver commesso degli errori, quindi ricontrollate con attenzione sia il ragionamento che la realizzazione.

#### **Esercizio 4**

Scrivere la macchina di Turing deterministica ad un solo nastro che riconosce le stringhe del tipo  $ww$ , con  $w \in \{a, b\}^+$ .

Soluzione1 (solo il ragionamento): devo simulare con una MdT deterministica una MdT non deterministica, ovvero devo assumere come stringa  $w$  i primi  $k$  caratteri ( $k = 1 \dots |ww'|$ ), inserire un marcatore e quindi confrontare la stringa prima del marcatore con quella che lo segue.

Soluzione2 (solo il ragionamento): inserisco dei marcatori all'inizio e alla fine della stringa, quindi inizio a spostarli verso il centro; quando i due marcatori si incontrano, ho diviso la stringa in ingresso in due parti. A questo punto si inizia il confronto carattere per carattere delle due parti.

#### **Esercizio 5**

Ci sono  $N$  prigionieri, in fila indiana davanti ad un muro; ogni prigioniero ha un cappello in testa, che può essere bianco o nero. Ciascun prigioniero può vedere soltanto il prigioniero che gli sta davanti, vedendo di che colore ha il cappello, ma non può sapere il colore del proprio cappello. Ogni prigioniero deve indovinare il colore del proprio cappello; se sbaglia, viene fucilato. Quanti prigionieri si possono salvare? Trovare un algoritmo della soluzione e scrivere un'automa a stati finiti che lo esegue (per casa).

#### **Esercizio 6**

Scrivere una macchina di Turing deterministica ad un solo nastro che riconosce una stringa avente lo stesso numero di 'a' e di 'b' (per casa).

# MCC: esercitazione del 7 febbraio 2008

## Introduzione al linguaggio LISP

### Esercizio 1

Definire la funzione Fattoriale

Soluzione

```
(defun fattoriale (n)
  (cond ((= n 0) 1)
        (T (* n (fattoriale (- n 1))))))
```

### Esercizio 2

Scrivere la funzione che restituisce l'ultimo elemento di una lista (si tralascino i controlli sulla lunghezza della lista ingresso, supponendo che abbia almeno un elemento).

Soluzione

```
(defun ultimo (L)
  (cond ((eq nil (cdr L)) (car L))
        (T (ultimo(cdr L)))))
```

### Esercizio 3

Date due liste, scrivere la funzione che ne fa l'intersezione.

**Nota:** quando si lavora su due liste, la ricorsione va eseguita su una soltanto di esse per evitare di complicare troppo il programma.

Soluzione

```
(defun inter (L1 L2)
  (cond ((eq L1 nil) nil)
        ((member (car L1) L2) (cons (car L1) (inter (cdr L1) L2)))
        (T (inter (cdr L1) L2))))
```

## Utilizzo dell'interprete

Per gli utenti GNU/Linux, l'interprete Common Lisp è disponibile per tutte (o quasi) le distribuzioni installando il pacchetto clisp. Per Mac OS X lo stesso software è installabile mediante [Fink](#).

Per usare l'interprete è sufficiente digitare il comando clisp da terminale; all'interno dell'interprete potete definire le funzioni e, successivamente, richiamarle con i relativi parametri.

Un utilizzo più efficiente consiste nello scrivere i programmi con un editor di testo, compilarli con il comando `clisp -c nomeprogramma.lisp` (verificando che il compilatore non restituisca errori), avviare l'interprete e caricare il programma compilato con il comando

```
(load "nomeprogramma.fas")
```

A questo punto potete invocare il programma con i relativi parametri.

In realtà la cosa funziona ugualmente caricando il programma sorgente con

```
(load "nomeprogramma.lisp")
```

Nota: sicuramente ci sono modi migliori di utilizzare l'interprete, il compilatore e i programmi compilati; io al momento ho provato solo questi.

## Link utili

- [Common Lisp Home](#)
- [Breve tutorial su CLISP](#)
- [Presentazione ppt dal sito dell'ing. Laura](#)

*Materiale prodotto da Gica78R per [FORUMiNG](#)*

*Segnalare eventuali errori a: [Gica78R](#)*

## MCC: esercitazione del 14 febbraio 2008

Ancora sul LISP

### Esercizio 1

Definire la funzione che esegue l'unione di due insiemi. Un insieme è una lista senza elementi ripetuti.

```
(defun unione (a b)
  (cond ((null a) b)
        ((member (car a) b) (unione (cdr a) b))
        (T (cons (car a) (unione (cdr a) b)))))
```

### Esercizio 2

Scrivere la funzione 'Appartiene' che stabilisce se un elemento appartiene ad una lista. La funzione deve prevedere anche il caso in cui gli elementi della lista non siano atomici (cioè siano a loro volta delle liste). Non utilizzare la funzione MEMBER.

```
(defun appartiene (a b)
  (cond ((null b) nil)
        ((listp (car b)) (or (appartiene a (car b))
                              (appartiene a (cdr b))))
        ((eq a (car b)) T)
        (T (appartiene a (cdr b)))))
```

### Esercizio 3

Scrivere la funzione che, ricevuta in ingresso una matrice quadrata M (lista contenente N liste di N elementi), restituisce la lista composta dagli elementi della diagonale principale di M. Si può fare uso di funzioni ausiliarie.

```
(defun sub (M)
  (cond ((null M) nil)
        (T (cons (cdr (car M)) (sub (cdr m))))))

(defun diagonale (M)
  (cond ((null M) nil)
        (T (cons (caar M) (diagonale (sub (cdr M))))))
```

## Esercizio 4

Scrivere la funzione che restituisce il prodotto cartesiano di due insiemi.

```
(defun COPPIE (A L)
  (cond ((null L) nil)
        (T (cons (list A (car L)) (COPPIE A (cdr L))))))

(defun PC (L1 L2)
  (cond ((null L1) nil)
        (T (append (COPPIE (car L1) L2) (PC (cdr L1) L2)))))
```

## Esercizio per casa: problema della festa di compleanno

Ad una festa di compleanno ogni invitato saluta e stringe la mano solo alle persone che conosce. Si consideri la proposizione: *alla fine della festa, ci sono almeno due persone che hanno stretto lo stesso numero di mani*. Questa proposizione è vera?

Materiale prodotto da Gica78R per [FORUMiNG](#)  
Segnalare eventuali errori a: [Gica78R](#)

## MCC: esercitazione del 21 febbraio 2008

LISP e problemi NP-Completi

### Esercizio 1

Scrivere la funzione LISP che, ricevuta in ingresso una lista di liste, restituisce la lista rappresentante l'intersezione delle liste. Si consideri definita la funzione INTERS che restituisce l'intersezione tra due liste (semplici).

Soluzione:

```
(defun INT2 (L)
  (cond ((NULL (CDR L)) (CAR L))
        (T (INTERS (CAR L) (INT2 (CDR L))))))
```

### Esercizio 2

Compito A del 25 marzo 2004 – esercizio 4

Soluzione:

```
(defun ASF (Q L)
  (COND ((NULL L) (FINALE Q))
        (T (ASF (DELTA (CAR L) Q) (CDR L)))))
```

## Problemi NP-completi

### Vertex Cover (VC)

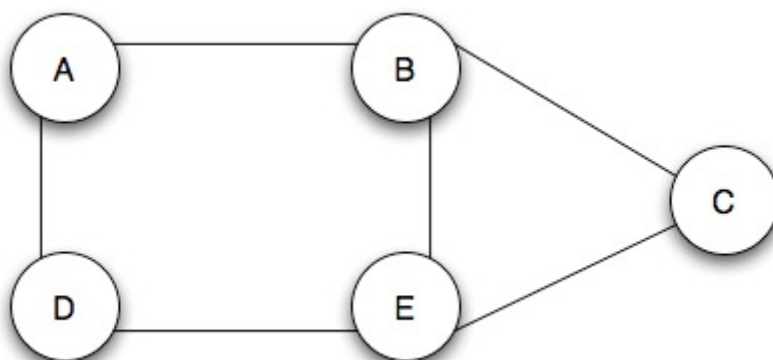
Dato un grafo  $G=(V, E)$  e un intero  $k>0$ ,

$$\exists V' \subseteq V \text{ t.c. } |V'| \leq k \wedge \forall e=(v_i, v_j) \in E \ v_i \in V' \vee v_j \in V' ?$$

In parole povere: dato un grafo  $G=(V, E)$  e un  $k>0$ , riusciamo a trovare un sottoinsieme  $V'$  di  $V$  di vertici, in numero minore o uguale a  $k$ , tale che ogni arco  $e$  di  $E$  abbia almeno un estremo in  $V'$ ?

### Esempio di Vertex-Cover

Si consideri il seguente grafo



Una possibile soluzione è rappresentata dall'insieme di vertici  $V'=\{B, C, D\}$ . Infatti tutti i rami del grafo, ovvero  $E=\{(A, B), (A, D), (B, C), (B, E), (C, E), (D, E)\}$  hanno almeno un vertice in  $V'$ . Appare evidente che, in questo caso, una soluzione con  $k<3$  non è possibile.

## Independent-Set (IS)

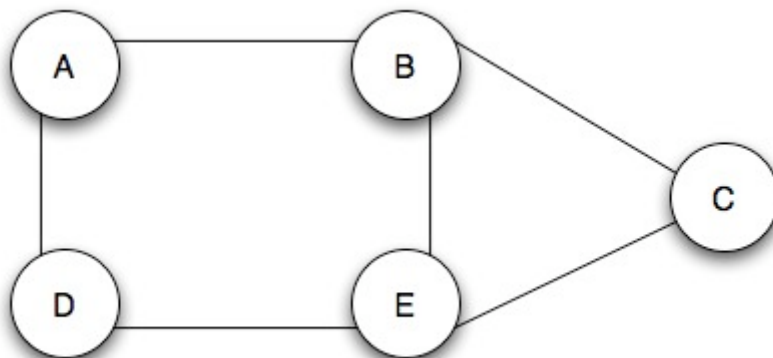
Dato un grafo  $G=(V, E)$  e un intero  $k>0$ ,

$$\exists V' \subseteq V \text{ t.c. } |V'| \geq k \wedge \forall (v_i, v_j) \in V' \text{ non } \exists e=(v_i, v_j) \in E \text{ ?}$$

In parole povere: dato un grafo  $G$  e un  $k>0$ , è possibile individuare un insieme  $V'$  di almeno  $k$  vertici tali che, comunque presi due vertici in  $V'$ , non esiste un arco nel grafo che li colleghi direttamente?

## Esempio di Independent-Set

Si consideri il seguente grafo (uguale al precedente):



Una possibile soluzione è costituita dall'insieme di vertici  $V'=\{B, D\}$ .

Infatti, nel grafo non esiste alcun arco tra i nodi B e D. Se aggiungessimo un qualsiasi altro vertice a  $V'$ , ciò non sarebbe più vero. Un'altra possibile soluzione è costituita dall'insieme  $V''=\{A, E\}$ .

## Relazione tra le soluzioni del problema VC e del problema IS

I problemi VC e IS relativi ad uno stesso grafo sono indipendenti o hanno una qualche relazione? Dagli esempi precedenti si può dedurre, intuitivamente, che i nodi che non fanno parte della soluzione del problema VC costituiscono una soluzione del problema IS sullo stesso grafo.

Dimostrazione: supponiamo che  $V'$  sia una soluzione del problema VC e consideriamo  $V''=V-V'$ . Se  $V''$  è tale che tra i suoi vertici non vi siano archi,

allora  $V'$  è soluzione del problema IS, altrimenti in  $V''$  ci sono due elementi connessi da un arco che non sono presenti in  $V'$  e dunque  $V'$  non è una soluzione di VC.

### Esercizio 3

Scrivere la funzione che, ricevuti in ingresso un insieme  $V$  ed un grafo  $G$  (rappresentato come coppie di nodi tra i quali esiste un arco), verifica se  $V$  è una soluzione del problema Vertex Cover.

Soluzione:

```
(defun ISVC (V G)
  (cond ((NULL G) T)
        ((OR (MEMBER (CAAR G) V) (MEMBER (CADAR G) V)) (ISVC V
(CDR G)))
        (T nil)))
```

### Esercizio 4

Scrivere la funzione che, ricevuti in ingresso un insieme  $V$  ed un grafo  $G$  (rappresentato come coppie di nodi tra i quali esiste un arco), verifica se  $V$  è una soluzione del problema Independent Set.

Soluzione:

```
(defun ISIS (V G)
  (cond ((null G) T)
        ((not (AND (MEMBER (caar G) V) (MEMBER (cadar G) V)))
(ISIS V (cdr G)))
        (T nil)))
```

*Materiale prodotto da Gica78R per [FORUMiNG](#)*

*Segnalare eventuali errori a: [Gica78R](#)*

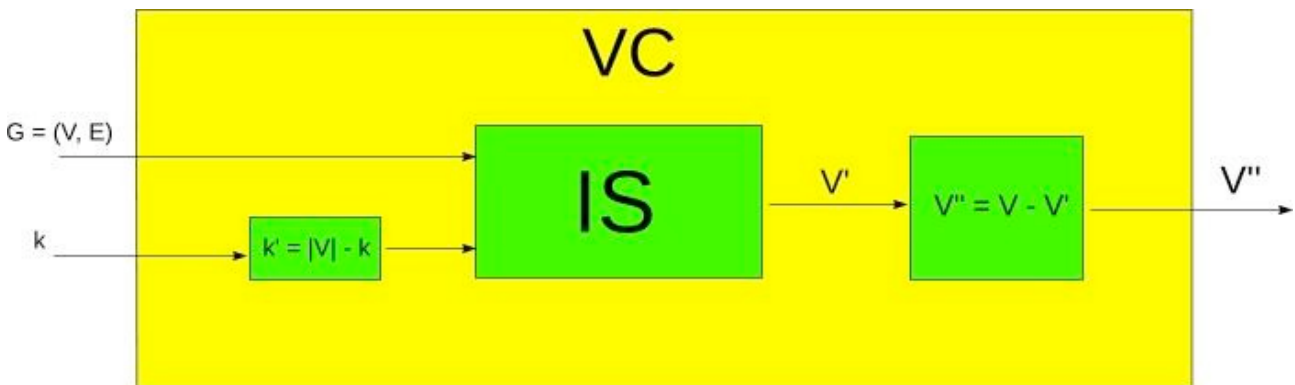
## MCC: esercitazione del 28 febbraio

### Problemi NP-Completi e riduzioni

#### Esempio di riduzione

Supponendo di avere a disposizione una *scatola nera* che risolve il problema Independent Set, mostrare come si può usarla per risolvere il problema Vertex Cover.

La scatola che risolve Independent Set riceve in ingresso un grafo  $G=(V, E)$  e un intero  $k$  e restituisce in uscita il sottoinsieme  $V'$  di  $V$  contenente  $k$  nodi di  $G$  non interconnessi tra loro. Il problema Vertex Cover, invece, consiste nel trovare un sottoinsieme  $V'$  di  $V$  con al massimo  $k$  nodi di  $G$  che coprono tutti gli archi  $E$ . Quindi, per risolvere il problema VC usando la scatola che risolve il problema IS dobbiamo adattare l'input di VC a quello di IS. In questo caso basta notare che, dati  $G=(V, E)$  e  $k$ , affinché esista una soluzione al problema VC è sufficiente che esista una soluzione al problema IS avente come ingresso lo stesso grafo  $G$  e come intero  $k'=(|V| - k)$ . Per ottenere la soluzione del problema VC, dunque, diamo in input alla scatola nera il grafo  $G$  e  $k'$  e all'uscita prenderemo come soluzione l'insieme  $V''=V - V'$ , dove  $V'$  è la soluzione del problema IS fornita dalla scatola nera.



La notazione con cui si indica questa riduzione è  $VC \leq IS$ .

È abbastanza semplice eseguire la riduzione nel verso opposto ( $IS \leq VC$ ), nel qual caso la scatola nera è rappresentata da VC.

#### Esercizio 1 (7 luglio 2003, compito A, esercizio 1)

$VC \leq CM$

Dobbiamo trasformare un'istanza del problema VC in un'istanza del problema CM

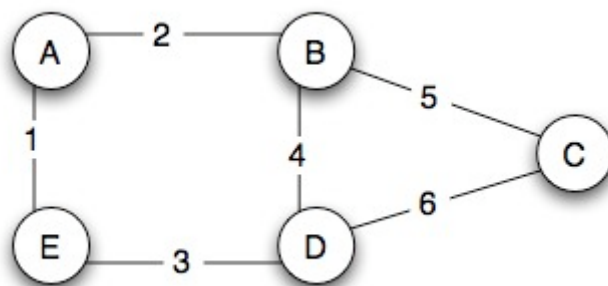
Soluzione: supponiamo di avere una scatola nera, chiamata CM, che riceve in

ingresso l'insieme dei capi cosca, l'insieme dei picciotti e il valore  $k$ , fornisce in uscita l'eventuale insieme (se esiste) dei capi cosca che risolve il problema.

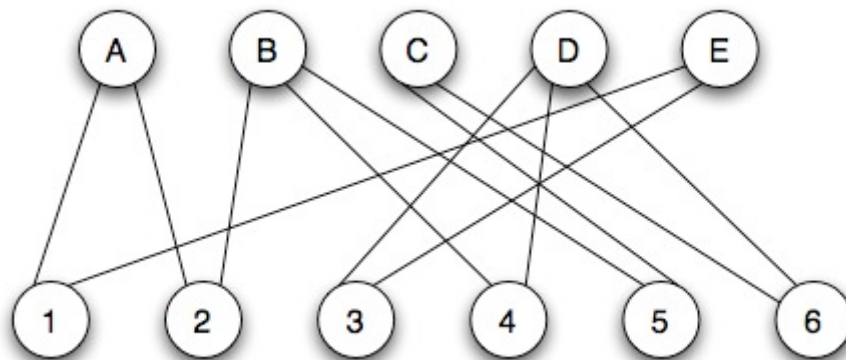
Il problema VC, come sappiamo, riceve in ingresso un grafo  $G=(V, E)$  e il valore  $k$ , restituendo in uscita (se esiste) l'insieme di vertici  $V'$  incluso in  $V$  che "coprono" tutti gli archi  $E$ . Per risolvere il problema VC usando la scatola nera CM dobbiamo adattare l'input di VC a CM. In particolare, l'insieme  $V$  dei vertici di  $G$  diventa l'insieme dei capi cosca, mentre l'insieme  $E$  degli archi rappresenterà i picciotti. Il valore  $k$  mantiene immutato il suo significato. A questo punto, l'output di CM costituirà effettivamente l'insieme di nodi in grado di coprire tutti gli archi.

Esempio

Consideriamo la seguente istanza di un problema VC:



Essa si trasforma, secondo le regole definite prima, nella seguente istanza di CM:



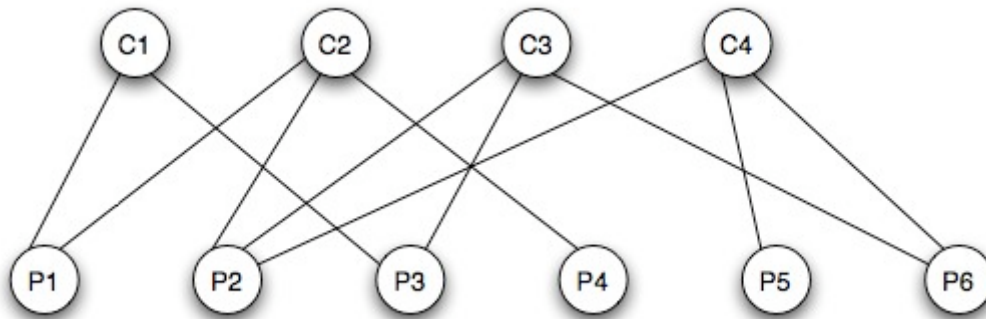
Considerazioni: ci potremmo domandare se sia possibile eseguire la riduzione nel verso opposto, cioè  $CM \leq VC$ . Si nota però che una generica istanza di un problema CM non è rappresentabile con un semplice grafo. In particolare, se ci fosse un picciotto che prende ordine da un solo capo cosca o da più di due capi cosca, l'istanza non sarebbe un grafo. Effettivamente un grafo rappresenta solo un tipo particolare di istanza di CM, cioè quelle istanze in cui ogni picciotto prende ordini esattamente da due capi cosca. Questo genere di riduzione è particolarmente complessa ed è fattibile soltanto ricorrendo al concetto di ipergrafo.

## Esercizio 2 (7 luglio 2003, compito B, esercizio 1)

Premesso che il problema CM è noto essere NP-completo, mostrare che anche HS è NP-completo.

Soluzione: dobbiamo far vedere che, disponendo di una scatola nera che risolve il problema HS, possiamo farne uso per risolvere il problema della cupola mafiosa, ovvero dobbiamo mostrare che  $CM \leq HS$ . Il problema Hitting Set riceve in ingresso l'insieme  $S$ , l'intero  $k$  e i sottoinsiemi  $C_1, C_2, \dots, C_m$ , restituendo in uscita l'insieme  $S'$  contenuto in  $S$  che ha cardinalità minore o uguale a  $k$  e che contiene almeno un elemento per ciascun  $C_i$ . Affinché questa scatola nera HS risolva il problema CM stabiliamo che l'insieme  $S$  coincide con l'insieme dei capi cosca, mentre consideriamo come insieme collezione  $C_i$  l'insieme dei capi cosca che danno ordini al picciotto  $i$ -esimo. Il valore  $k$  resta immutato per entrambi i problemi.

Esempio



In questo caso abbiamo che  $S = \{C1, C2, C3, C4\}$ , mentre gli insieme collezione, che chiameremo  $P_i$  per evitare confusione, sono:

$P_1 = \{C1, C2\}$ ;  $P_2 = \{C2, C3, C4\}$ ;  $P_3 = \{C1, C3\}$ ;  $P_4 = \{C2\}$ ;  
 $P_5 = \{C4\}$ ;  $P_6 = \{C3, C4\}$

Come risultato, la scatola nera HS restituirà  $S' = \{C2, C3, C4\}$

Osservazione: possiamo eseguire la riduzione anche nel verso opposto, ovvero  $HS \leq CM$ . In che modo?

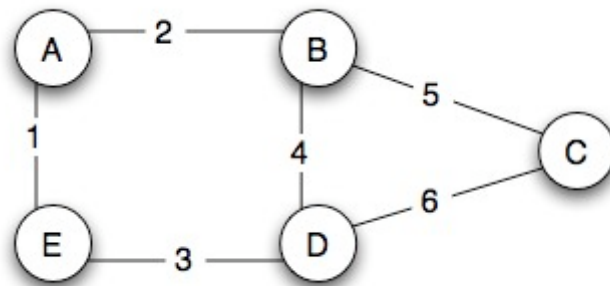
## Esercizio 3 (5 luglio 2006, compito A, esercizio 3)

Problema Set Cover (SC): dati in ingresso un insieme  $S$ ,  $r$  sottoinsiemi  $C_1 \dots C_r$  di  $S$  ed un intero  $k$ , posso prendere al più  $k$  sottoinsiemi  $C_i$  tali che la loro unione sia uguale a  $S$ ?

Sapendo che il problema Vertex Cover è NP-completo, mostrare che  $VC \leq SC$ .

Soluzione: in ingresso alla scatola che risolve il problema SC si mette come insieme  $S$  l'insieme degli archi di  $G$  e come insieme collezione  $C_i$  gli insiemi di

archi connessi al vertice i-esimo. Esempio:



Si sceglie come insieme  $S$  l'insieme  $\{1, 2, 3, 4, 5, 6\}$  e come collezioni:

$C_a = \{1, 2\}$ ;  $C_b = \{2, 4, 5\}$ ;  $C_c = \{5, 6\}$ ;  $C_d = \{3, 4, 6\}$ ;  $C_e = \{1, 3\}$

L'intero  $k$  resta invariato. All'uscita di  $SC$  avremo, ad esempio,

$C' = \{C_a, C_c, C_d\}$ , ovvero i vertici  $V' = \{A, C, D\}$  che sono una soluzione del problema  $VC$  sull'istanza considerata.

*Segnalare eventuali errori a [gica78r@tele2.it](mailto:gica78r@tele2.it)*

## MCC: esercitazione del 6 marzo 2008

Riduzione di problemi NP-completi

### Esercizio 1

Sapendo che il problema Set Cover è NP-completo, dimostrare che il problema Hitting Set è NP-completo, cioè che **SC**  $\leq$  **HS**.

In pratica dobbiamo mostrare come trasformare un'istanza del problema SC in un'istanza del problema HS. Ricordiamo che il problema SC riceve in ingresso un insieme  $S$  di elementi, un insieme  $C = \{C_1, C_2, \dots, C_n\}$  di collezioni (ovvero di sottoinsiemi) di elementi di  $S$  ed un intero  $k \geq 1$ , fornendo in uscita un insieme  $C'$  incluso in  $C$  di al più  $k$  collezioni tali che la loro unione è uguale all'insieme  $S$ . Il problema HS, invece, riceve in ingresso un insieme  $T$  di elementi, un insieme  $D = \{D_1, \dots, D_m\}$  di collezioni di elementi di  $T$  ed un intero  $k \geq 1$ , fornendo in uscita un insieme  $T'$  incluso in  $T$  di al più  $k$  elementi tale che l'intersezione tra  $T'$  ed ogni  $D_i$  è non vuota.

### Soluzione

L'insieme  $C$  delle collezioni in ingresso a SC diventa l'insieme  $T$  in ingresso a HS; come insieme di collezioni  $D$  in ingresso a HS prendo gli insiemi  $D_1, \dots, D_j, \dots, D_z$  aventi come elementi le collezioni di  $C$  contenenti il  $j$ -esimo elemento di  $S$  ( $z = |S|$ ). L'intero  $k$  resta invariato.

**Esempio** di istanza di SC e relativa conversione in istanza di HS:

$S = \{a, b, c, d, e, f\}$

$C_1 = \{a, b\}; C_2 = \{c, d\}; C_3 = \{d, e\}; C_4 = \{a, f\}; C_5 = \{c, d, f\};$

$C_6 = \{b, e\}$

Esempio di soluzione di SC per  $k=3$ :

$S' = \{C_1, C_5, C_6\}$ .

Infatti  $(C_1 \cup C_5 \cup C_6) = \{a, b, c, d, f, b, e\}$ .

Verifichiamo che la conversione funziona:

nel nostro caso l'insieme  $T$  in ingresso a HS diventa:

$T = \{C_1, C_2, C_3, C_4, C_5, C_6\}$

mentre le collezioni in ingresso a HS saranno:

$D_1 = \{C_1, C_4\}$  (collezioni contenenti l'elemento 'a')

$D_2 = \{C_1, C_6\}$  (collezioni contenenti l'elemento 'b')

$D_3 = \{C_2, C_5\}$  (collezioni contenenti l'elemento 'c')

$D_4 = \{C_2, C_3, C_5\}$  (... elemento 'd')

$D5 = \{C3, C6\}$  (... elemento 'e')

$D6 = \{C4, C5\}$  (... elemento 'f')

In uscita dal blocco HS avremo quindi un insieme  $T'$  incluso in  $T$  di collezioni avente intersezione non nulla con ciascun  $D_i$ ; nel caso in esame, un possibile risultato è:

$$T' = \{C1, C5, C6\}$$

$T'$  è tale da avere intersezione non nulla con ogni  $D_i$ , quindi  $T'$  contiene almeno una collezione contenente l'elemento 'a', almeno una collezione contenente l'elemento 'b', e così via...

## Esercizio 2 (20 aprile 2006, esercizio 3, compito A)

Dimostrare che  $VC \leq HS$

### Soluzione

Come insieme di ingresso  $S$  di HS si usa l'insieme dei nodi  $V$ , mentre come collezioni  $C1, \dots, C_n$  si usano i nodi collegati all'arco  $i$ -esimo (ogni arco si può pensare come collezione di due nodi).

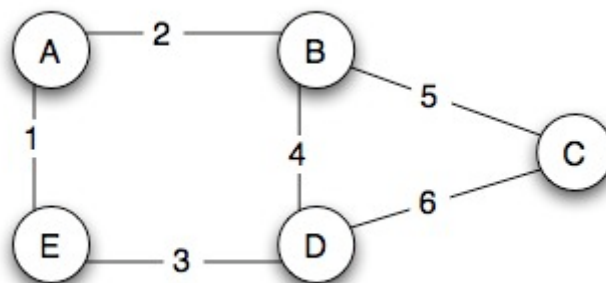
## Esercizio 3 (20 aprile 2006, esercizio 3, compito B)

Dimostrare che  $DS \leq HS$

### Soluzione

Come insieme di ingresso  $S$  a HS si passa l'insieme  $V$  dei nodi di  $G$ , mentre come collezioni  $C1, C2, \dots, C_i, \dots, C_n$  si passano gli insiemi di nodi collegati al nodo  $i$ -esimo, compreso lo stesso nodo  $i$ -esimo. L'intero  $k$  resta inalterato.

### Esempio



$V = \{A, B, C, D, E\}; E = \{1, 2, 3, 4, 5, 6\}$

$S = V; C1 = \{A, B, E\}; C2 = \{A, B, C, D\}; C3 = \{B, C, D\}; C4 = \{B, C, D, E\};$

$C5 = \{A, D, E\}$

Una possibile soluzione è, per esempio,  $S' = \{B, E\}$ .

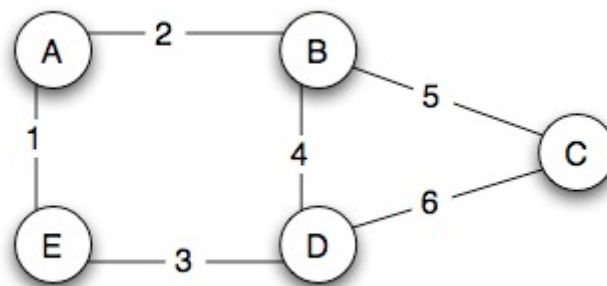
### Esercizio 4 (20 aprile 2006, esercizio 3, compito B)

Mostrare che  $IS \leq AUC$

#### Soluzione

Osserviamo che trasformando il grafo  $G$  in una matrice di incidenza e passando questa matrice di incidenza e l'intero  $k$  come input ad AUC, il risultato è un Vertex Cover. Quindi, per avere in uscita un Independent Set dobbiamo passare in ingresso ad AUC la matrice di incidenza e un intero  $k' = (|V| - k)$ ; a tal proposito si rivedano le riduzioni  $IS \leq VC$  e  $VC \leq IS$ .

Ricordiamo che la matrice di incidenza di un grafo  $G$  avente  $m$  nodi e  $n$  archi è una matrice  $m \times n$  che ha sulla colonna  $i$ -esima un 1 in corrispondenza dei nodi collegati dall'arco  $i$ -esimo. Esempio:



In questo caso,  $m=5$  e  $n=6$ . La corrispondente matrice di incidenza è allora:

	1	2	3	4	5	6
A	1	1	0	0	0	0
B	0	1	0	1	1	0
C	0	0	0	0	1	1
D	0	0	1	1	0	1
E	1	0	1	0	0	0

Ovvero: l'arco 1 (prima colonna) collega i nodi A e E, l'arco 2 (seconda colonna) collega i nodi A e B, e così via...

E' evidente che AUC restituisce una matrice di  $k$  righe associabili a  $k$  nodi di  $V$  che coprono tutti gli archi del grafo, in quanto tale matrice presenta almeno un 1 in corrispondenza di ogni colonna (associata agli archi), quindi tutti gli archi sono coperti. La soluzione complementare, allora, è un IS.

**ATTENZIONE: la presente esercitazione non è completa; mancano gli esercizi di riduzione  $VC \leq PL01$  e  $CM \leq PL01$  perché dai miei appunti non riesco a ricostruire il testo esatto del problema PL01 (Programmazione Lineare 0 1). Chiunque volesse contribuire al completamento dell'esercitazione è il benvenuto!!!**

**Nota: probabilmente per un errore, tra i compiti di esame scaricabili dal sito del corso ci sono più compiti datati 20 aprile 2006. Attenzione quindi a verificare i testi degli esercizi.**

*Segnalare eventuali errori o osservazioni a [Gica78R](#).*

## MCC: esercitazione del 12 marzo 2008

Riduzione di problemi NP-completi

### Esercizio 1

Dimostrare che  $IS \leq PL01$

Ricordiamo cosa dice il testo del problema PL01

*Problema Programmazione Lineare 0-1: dato un insieme di variabili  $Z=\{z_1, \dots, z_n\}$  con dominio in  $\{0, 1\}$ , un'insieme di disuguaglianze su  $Z$  e un intero  $k \geq 1$ , esiste un'assegnazione di almeno  $k$  variabili ad 1 tale che tutte le disuguaglianze siano verificate?*

### Soluzione

Dobbiamo mostrare come adattare un'istanza del problema IS in un'istanza del problema PL01. Mappiamo i nodi del grafo sulle variabili di  $Z$  nel seguente modo: per ogni arco tra due nodi  $x$  e  $y$  abbiamo la disequazione

$$z_x + z_y \leq 1$$

dove il segno di minore o uguale a uno significa che i due nodi non devono essere entrambi presenti nella soluzione di IS perché sono tra loro collegati.

A tutte le disequazioni del tipo precedente si aggiunge il vincolo su  $k$ , ovvero:

$$\sum z_i \geq k$$

La soluzione di questa istanza di PL01, in cui determinate variabili  $z$  sono poste a uno, si traduce nella soluzione di IS in cui l'insieme dei nodi indipendenti è costituito dai nodi associati alle variabili poste a uno.

### Esercizio 2 (12 settembre 2005, esercizio 3, compito B)

Dimostrare che  $Sudoku \leq GraphColoring$

### Soluzione

Un'istanza di Sudoku è una matrice  $(n^2) \times (n^2)$  con alcune celle preassegnate (vincoli). Si hanno a disposizione  $n^2$  diverse cifre da assegnare alle celle. Su una stessa riga, una stessa colonna e all'interno di un blocco  $n \times n$  non vi devono essere cifre uguali.

Esempio per  $n=3$ :


L'istanza di Sudoku si mappa su GC facendo corrispondere ogni cella ad un nodo del grafo, per un totale di  $(n^2) \times (n^2)$  nodi, avendo a disposizione per la colorazione  $n^2$  colori diversi. Ciascun nodo è collegato con un arco a tutti i nodi della stessa riga, della stessa colonna e dello stesso blocco. Nell'esempio, il nodo associato alla cella (1, 1) è collegato con un arco a tutti i nodi associati alle celle evidenziate. Il vincolo sulle cifre del Sudoku si traduce nel vincolo che un nodo del grafo abbia colore diverso da tutti i nodi ad esso collegati.

### **Esercizio 3 (13 aprile 2005, esercizio 3, compito B)**

Dimostrare che  $SP \leq CL$

#### **Soluzione**

Semplicemente, le collezioni di SP diventano i nodi  $V$  del grafo di input a SP, mentre gli archi del grafo sono costituiti dalle coppie di collezioni disgiunte tra loro.

#### **Esempio**

La seguente istanza del problema SP:

$$S = \{a, b, c, d, e, f\}$$

$$C1 = \{a, b\}; C2 = \{b, c\}; C3 = \{c, d\}; C4 = \{d, e\}; C5 = \{e, f\}; C6 = \{a, f\}$$

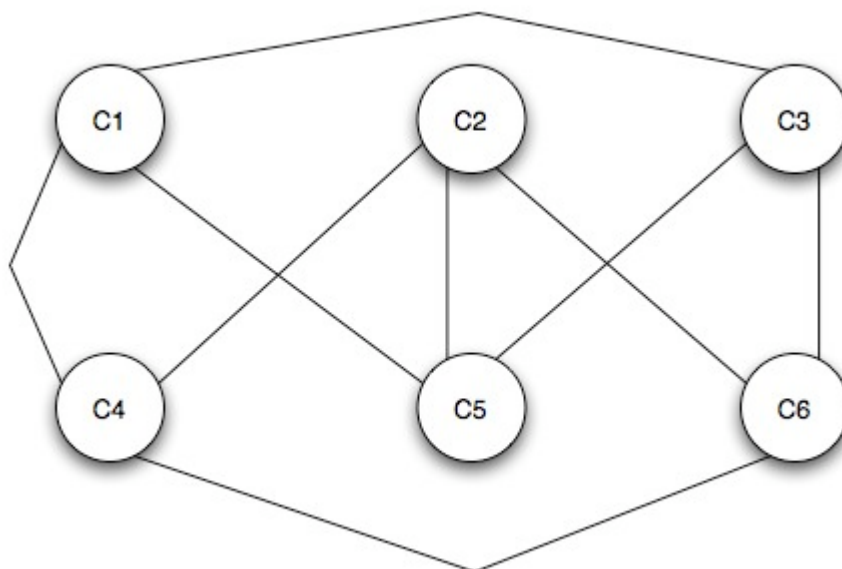
$$k = 3$$

diventa, nel problema CL

$V = \{C1, C2, C3, C4, C5, C6\}$

$E = \{(C1, C3), (C1, C4), (C1, C5), (C2, C4), (C2, C5), (C2, C6), (C3, C5), (C3, C6), (C4, C6)\}$

$k = 3$



Una soluzione possibile è  $V' = \{C1, C3, C5\}$ .

Segnalare eventuali errori o osservazioni a [Gica78R](#).