Fondamenti di Programmazione - CdL in MATEMATICA

I Prova di verifica del 2/4/2012

num. eserc.	1	2	3	4	5
punt. tot	5	5	7	5	8

N.B.: Negli esercizi di programmazione, viene valutata anche la leggibilità del codice proposto. Inoltre, non è consentito l'uso di istruzioni che alterino il normale flusso dell'esecuzione all'interno di cicli e provochino l'uscita forzata.

ESERCIZIO 1 (5 punti)
Costruire un NFA N che accetti stringhe sull'alfabeto $\Sigma = \{1, 2, 3\}$ tali che l'ultimo simbolo appaia almento due volte, senza che però tra le due occorrenze vi sia l'occorrenza di un numero più grande del simbolo ripetuto. Ad esempio 22, 3113, 123123 appartengono a $L(N)$, mentre 232, 1231 non vi appartengono. Attenzione: la sequenza 23211212 appartiene a $L(N)$.

ESERCIZIO 2 (5 punti)
Dato l'NFA N, descritto dalla seguente tabella di transizione:

		0	**1**	**2**
$\rightarrow$	$\mathbf{q_0}$	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
$\star$	$\mathbf{q_1}$	$\emptyset$	$\emptyset$	$\emptyset$

dimostrare formalmente che N accetta il linguaggio $L = \{w \in \Sigma^* | w = x2 \text{ con } x \in \Sigma^*\}$.

ESERCIZIO 3 (7 punti)
Data una sequenza di interi di dimensione *dim*, determinare se esistono esattamente *cont* occorrenze del valore *val*, utilizzando per questo una variabile booleana *check*.

Stato iniziale: {dim $\rightsquigarrow$ K, cont $\rightsquigarrow$ C, ..., $c[0] \rightsquigarrow V_0$, ...} con K > 0
Stato finale: { ...

- Completare la specifica in termini di stato iniziale e finale.
- Fornire la specifica di un algoritmo risolutivo utilizzando lo pseudocodice visto a lezione.

ESERCIZIO 4 (5 punti)

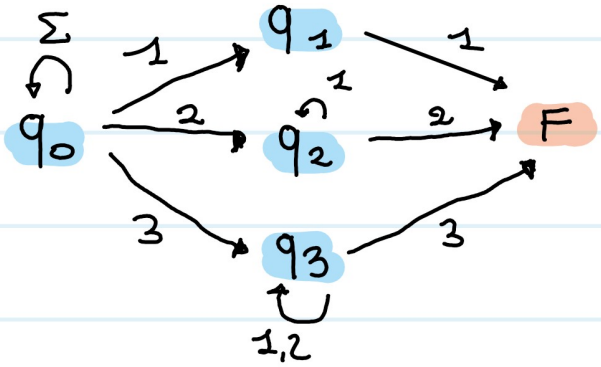
- Due numeri sono *co-primi* o *primi tra loro* se e solo se essi non hanno nessun divisore comune eccetto 1.
- Scrivere una funzione C che dati due numeri interi positivi controlli se sono *primi tra loro*.

ESERCIZIO 5 (8 punti)

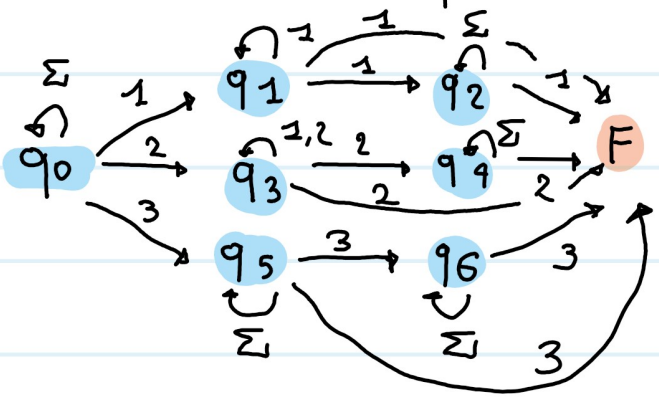
- Definiamo una sequenza *pari-dispari* come una sequenza di interi positivi distinti dove i pari si alternano con i dispari. La sequenza **termina** non appena si rompe l'alternanza, ovvero si trovano o due numeri pari consecutivi o due numeri dispari consecutivi. L'intero che viola la sequenza non si considera parte della sequenza.
- Scrivere un programma C che legga un valore *val* e una sequenza *pari-dispari* e stampi il massimo numero della sequenza che sia *co-primo* rispetto a *val*.
- Per controllare se un numero è *co-primo* rispetto a *val* usare la funzione dell'esercizio 4.

es. 1. 2012 $\Sigma = \{1,2,3\}$

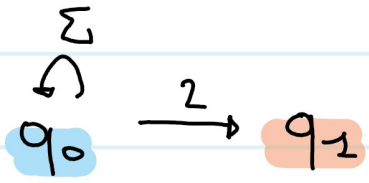
(i) solo l'ultima coppia valida



(ii) una coppia qualsiasi



es. 2. 2012 $\Sigma = \{0,1,2\}$



$L = \{ w \in \Sigma^* \mid w = x2, x \in \Sigma^* \}$

(i) $q_0 \in \hat{\delta}(q_0, w) \quad \forall w \in \Sigma^*$

- $q_0 \in \hat{\delta}(q_0, \varepsilon)$ (base)
- $\hat{\delta}(q_0, xa) = \bigcup_{p \in \hat{\delta}(q_0,x)} \delta(p,a) \supseteq \delta(q_0,a) \ni q_0$ (passo induttivo) (per ip. induttiva)

□

(ii) $\hat{\delta}(q_0, x2) \cap F \neq \emptyset \quad \forall x \in \Sigma^*$

- $\hat{\delta}(q_0, x2) = \bigcup_{p \in \hat{\delta}(q_0,x)} \delta(p,2) \supseteq \delta(q_0,2) =$ (per (i))

$= \{q_0, q_1\} \ni q_1 \quad \wedge \quad q_1 \in F \Rightarrow$

$\Rightarrow \hat{\delta}(q_0, x2) \cap F \neq \emptyset$ □

(iii) $\hat{\delta}(q_0, xk) \cap F = \emptyset \quad \forall x \in \Sigma^*, k \in \Sigma \setminus \{2\}$

- $\hat{\delta}(q_0, xk) = \bigcup_{p \in \hat{\delta}(q_0,x)} \delta(p,k) \subseteq \delta(q_0,k) \cup$

$$\cup\ \delta(q_1, k) = \{q_0\} \cup \emptyset = \{q_0\} \not\ni q_1 \Rightarrow$$
$$\Rightarrow\ \hat{\delta}(q_0, xk) \cap \underbrace{F}_{\{q_1\}} = \emptyset \qquad \square$$

Per la (ii), l'automa accetta L; per la (iii) l'automa non può accettare che sottoinsiemi di L. Quindi l'automa accetta esattamente L. ☑

es. 3.2012

(i)

Stato iniziale: {dim ↝ k, cont ↝ c, ..., c[0] ↝ v_0, ...}

con $k > 0$.

Stato finale: { check ↝ true se #{$0 \le i < k$ | c[i] è uguale a val} è uguale a c, false altrimenti }

(ii) occorrenze = 0; check = false;

Da i=0 a dim:

Se c[i] è uguale a val:

Incremento occorrenze

Se occorrenze è uguale a cont:

check = True

es. 4.2012

```
int are_coprime(int a, int b) {
    int max = a > b ? a : b; int min = a+b - max;
    int c = max % min;

    while (c != 0) {
        max = min;
        min = c;
        c = max % min;
    }

    return min == 1; // 1 se coprimi, 0 altrimenti
}
```

es. 5.2012

```
#include <stdio.h>
```

```
# define TRUE 1
# define FALSE 0

int are_coprime (inT, int); // si assume già implementata

int main () {
    int val;  scanf("%d", &val);

    int max; int last; scanf("%d", &last);
    max = are_coprime(last, val) ? last : -1;
    // -1 se non esistono coprimi
    int valid_seq = TRUE; int current;

    while (valid_seq) {
        scanf("%d", &current);

        if ((current + last) % 2 == 0) {
            valid_seq = FALSE;
        } else if (current > max && are_coprime(
            current, val)) {
            max = current;
```

```
        }
    }

    printf ("%d", max);
    return 0;
}
```