Laboratorio di Linguaggi Formali e Traduttori
LFT lab

Turno T2: cognomi **A-K** (corso A), **penultima** cifra del numero di matricola **pari**

Corso di Laurea in Informatica

a.a. 2021/2022

Docente

- Docente di LFT lab T2: Jeremy Sproston
- E-mail: sproston@di.unito.it (utilizzate vostro indirizzo istituzionale)
- Homepage: http://www.di.unito.it/~sproston
- Ricevimento:
  - Il ricevimento studenti avviene **previo appuntamento** da **prenotare via email.**
  - Il ricevimento studenti è **sempre sospeso** nei 3 giorni lavorativi precedenti un appello di esame (teoria/lab).
  - Il ricevimento via email è ammesso solo per dubbi e domande semplici.
- Assistente per LFT lab: Gianni Forlastro

Orario turno T2

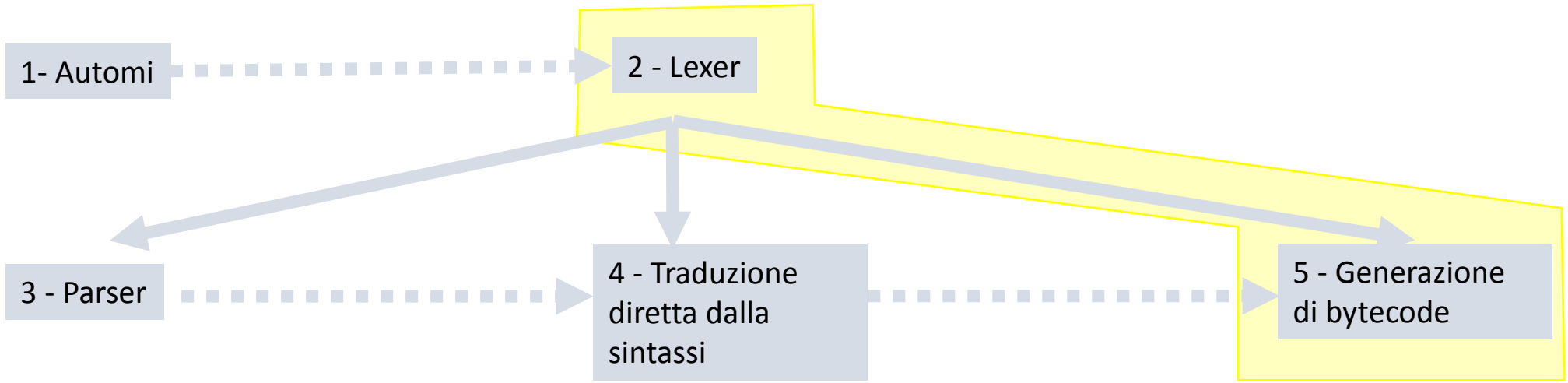
- Orario: giovedì 11.00 – 14.00
- Aula: laboratorio Dijkstra
- Date lezioni:
  - Ogni giovedì dal 14 ottobre 2021 al 9 dicembre 2021 (9 lezioni)
  - Giovedì 13 gennaio 2022 (1 lezione)
- Utilizziamo la stessa stanza WebEx per tutto il corso (link sulla pagina moodle del corso).

Pagina moodle/I-Learn

- File degli esercizi (pdf, aggiornato durante il corso):
  - Serie di esercizi da fare durante il corso.
  - Materiali aggiuntivi per aiutare lo svolgimento degli esercizi.
- Forum Annunci: utilizzato dal docente per mandare comunicazioni agli studenti.
- Forum di discussione:
  - Gli studenti possono iniziare una discussione.
  - Il docente e altri studenti possono partecipare alla discussione.
- Codice relativo agli esercizi (spesso *frammenti* di codice da completare).
- Slide (su un numero limitato di argomenti).
- Altri risorse: FAQ, esempi di input, link, ecc.

Progetto di laboratorio LFT lab

- Il progetto di laboratorio consiste in una serie di *esercitazioni* assistite mirate allo *sviluppo di un semplice traduttore*.
- Il corretto svolgimento di tali esercitazioni presuppone una buona conoscenza
  1. degli *argomenti di teoria di LFT* e
  2. del linguaggio di programmazione *Java*.



Modalità dell’esame di laboratorio

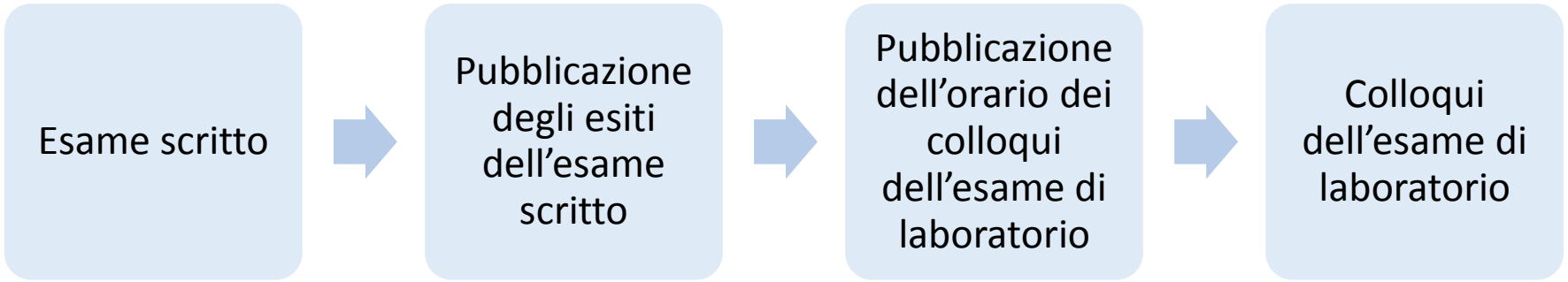
- Per sostenere l’esame a un appello è necessario *prenotarsi* su unito.it.
- Ogni studenti deve presentarsi all’esame con il codice di *tutti* gli esercizi obbligatori presentati durante il corso di laboratorio.
  - Il codice deve essere consegnato al docente (caricato sulla pagina moodle del laboratorio LFT T2) prima dell’esame di laboratorio.
- Il progetto di laboratorio può essere svolto individualmente o in gruppi formati da al massimo tre studenti.
- L’esame di laboratorio consiste in un colloquio *orale* ed è *individuale*.
- Anche se il codice è stato sviluppato in collaborazione con altri studenti, i punteggi ottenuti dai singoli studenti sono *indipendenti*.

Modalità dell’esame di laboratorio

- Per poter discutere il laboratorio è necessario aver prima superato la prova scritta relativa al modulo di teoria.
- L’esame di laboratorio deve essere superato nella stessa sessione d’esame in cui viene superato lo scritto, altrimenti lo scritto deve essere sostenuto nuovamente.
- Le sessioni d’esame in un a.a. (per LFT) sono tre:
  - gennaio/febbraio (due appelli)
  - giugno/luglio (due appelli)
  - settembre (un appello)

Modalità dell’esame di laboratorio

- Utilizzeremo *più giorni* per fare i colloqui dell’esame di laboratorio: il giorno ufficiale dell’appello e i giorni lavorativi successivi, fino al esaurimento degli studenti.
- La suddivisione degli studenti tra i giorni dei colloqui sarà comunicata (via email agli studenti iscritti all’appello) *dopo* la pubblicazione degli esiti dell’ultimo esame scritto di LFT che precede l’appello di laboratorio.
- Nel caso in cui non potete presentarvi in uno o più giorni lavorativi successivi al giorno ufficiale dell’appello: scrivere al docente *prima* della pubblicazione degli esiti dell’esame scritto.



Modalità dell’esame di laboratorio

- Durante l’esame vengono accertati:
  - il corretto svolgimento della prova di laboratorio;
  - la comprensione della sua struttura e del suo funzionamento;
  - la comprensione delle parti di teoria correlata al laboratorio.
- La presentazione di codice “funzionante” non è condizione sufficiente per il superamento della prova di laboratorio.
- Dal momento che durante la prova è possibile che venga richiesto di apportare modifiche al codice del progetto, è opportuno presentarsi all’esame con un’adeguata conoscenza del progetto e degli argomenti di teoria correlati.

Calcolo del voto finale

- I voti della prova scritta e della prova di laboratorio sono espressi in trentesimi (devono essere entrambi ≥ 18).
- Il voto finale di LFT è determinato calcolando la *media pesata* del voto della prova scritta e del laboratorio, *secondo il loro contributo in CFU* (con una eventuale modifica nel caso in cui lo studente sostiene una prova orale):

$$\text{voto finale} = \frac{\text{voto dello scritto} \times 2 + \text{voto del laboratorio}}{3} \pm \text{eventuale esito orale}$$

- La prova orale è obbligatoria se lo studente desidera ottenere la lode.

Sezione 1 - Implementazione di un DFA in Java

- Analisi lessicale: riconoscere sequenze di caratteri che rappresentano elementi atomici del programma.
- Esempio:

```
/* legge numero, scrive 123 poi
il numero sommato con 1 */
(do
        (read x1)
        (print 123 (+ x1 1))
)
```

Sezione 1 - Implementazione di un DFA in Java

- Analisi lessicale: riconoscere sequenze di caratteri che rappresentano elementi atomici del programma.
- Esempio:

```
/* legge numero, scrive 123 poi
il numero sommato con 1 */
(do
        (read x1)
        (print 123 (+ x1 1))
)
```

commento
parole chiavi
identificatore
costanti numeriche
operatore aritmetico
parentesi

Sezione 1 - Implementazione di un DFA in Java

- Analisi lessicale: riconoscere sequenze di caratteri che rappresentano elementi atomici del programma.
- Esempio:

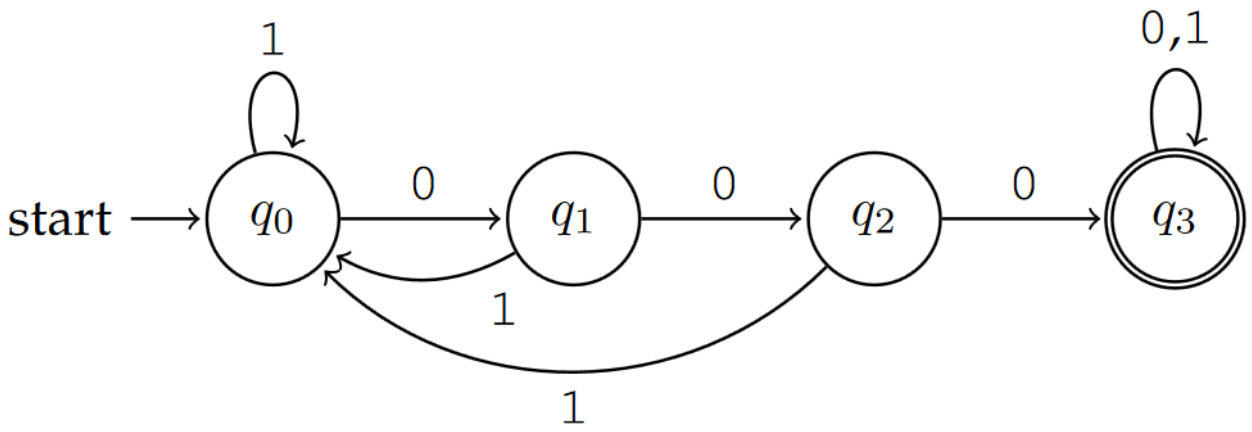
```
/* legge numero, scrive 123 poi
il numero sommato con 1 */
(do
        (read x1)
        (print 123 (+ x1 1))
)
```

commento
parole chiavi
identificatore
costanti numeriche
operatore aritmetico
parentesi

- Automi: macchine per riconoscere stringhe

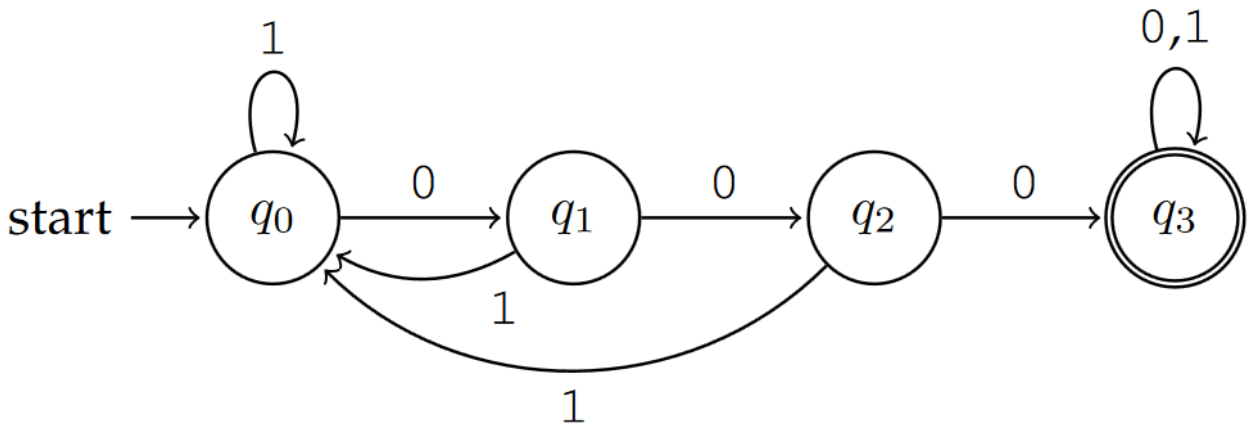
Sezione 1 - Implementazione di un DFA in Java

- Implementare un metodo Java per distinguere tra le stringhe accettate da un DFA e le stringhe non accettate.
- Dato il DFA $A=(Q,\Sigma,\delta,q_0,F)$ (ci ricordiamo che $\delta : Q \times \Sigma \rightarrow Q$), scrivere un metodo `scan` Java che «simula» *A* mentre legge una stringa:
  - Parametro del metodo: stringa `s`.
  - Valore restituito: `true` se la stringa `s` è accettata da *A*, altrimenti `false`.
- La parte principale del metodo `scan` è un ciclo, dove il corpo del ciclo corrisponde a:
  - la lettura di un singolo simbolo dalla sequenza di input, e
  - al passaggio dallo stato attuale del DFA a un altro stato secondo la funzione di transizione (cioè, se lo stato attuale è *q*, il simbolo letto è *a*, il DFA passa allo stato $\delta(q,a)$).
- Rappresentare lo stato attuale del DFA con un variabile intero `state`.



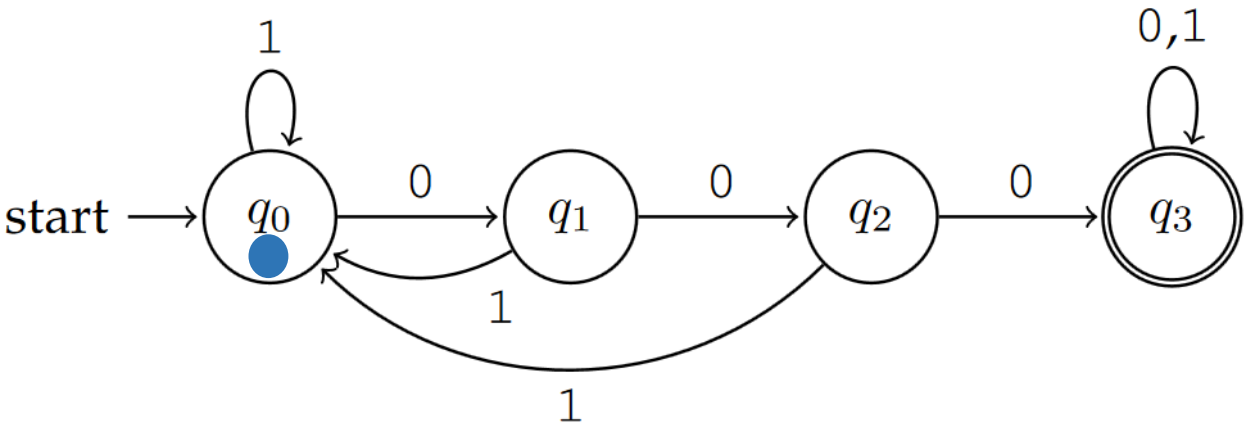

```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

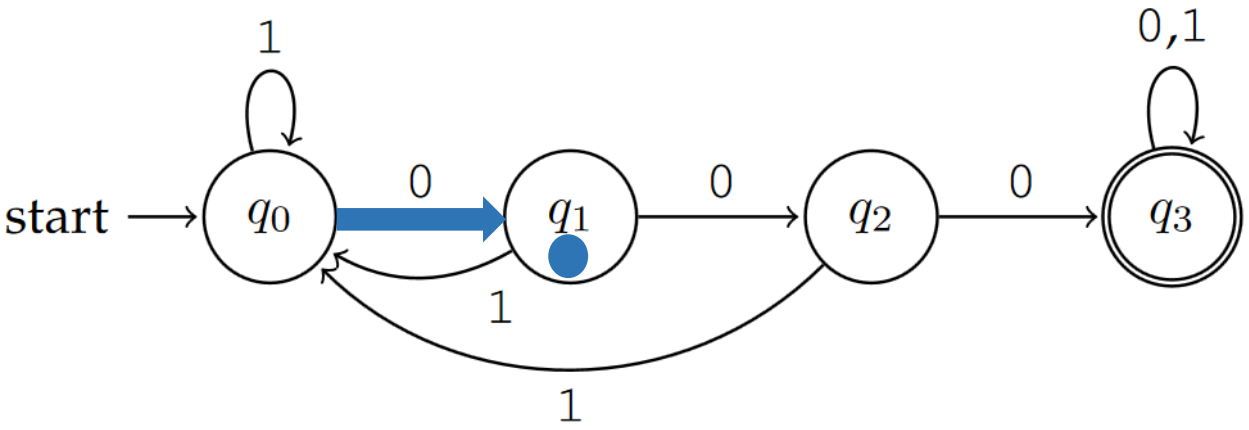
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 0
i = 0

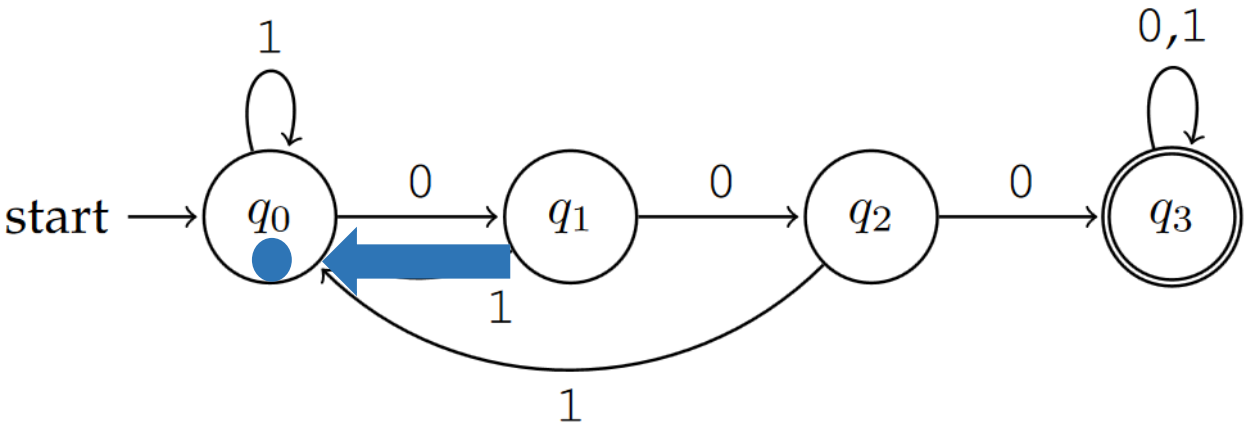
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 1
i = 1

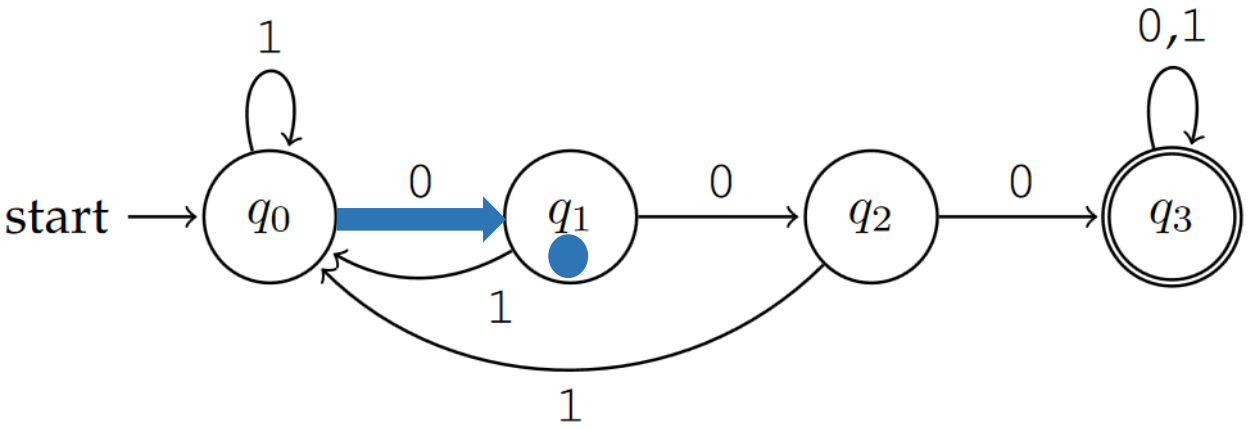
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 0
i = 2

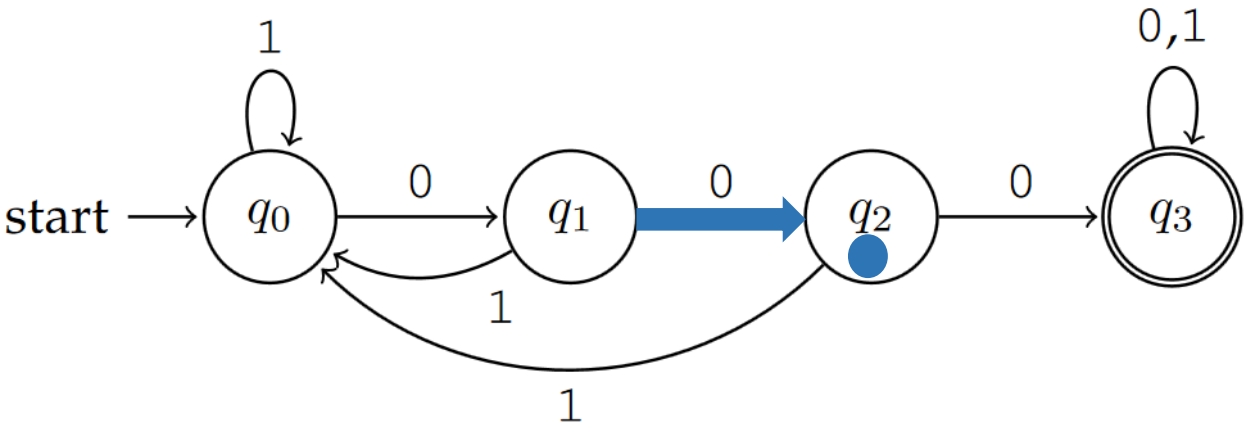
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 1
i = 3

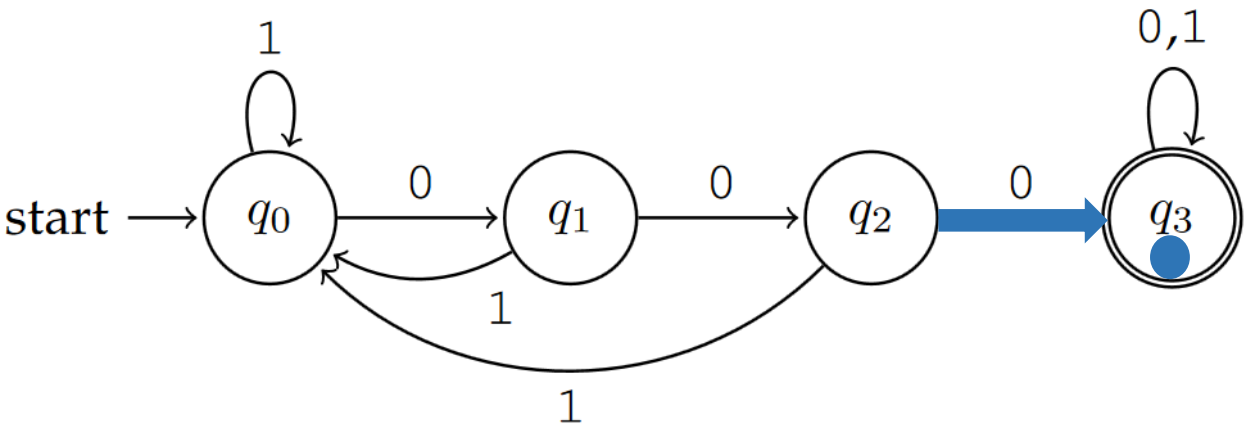
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 2
i = 4

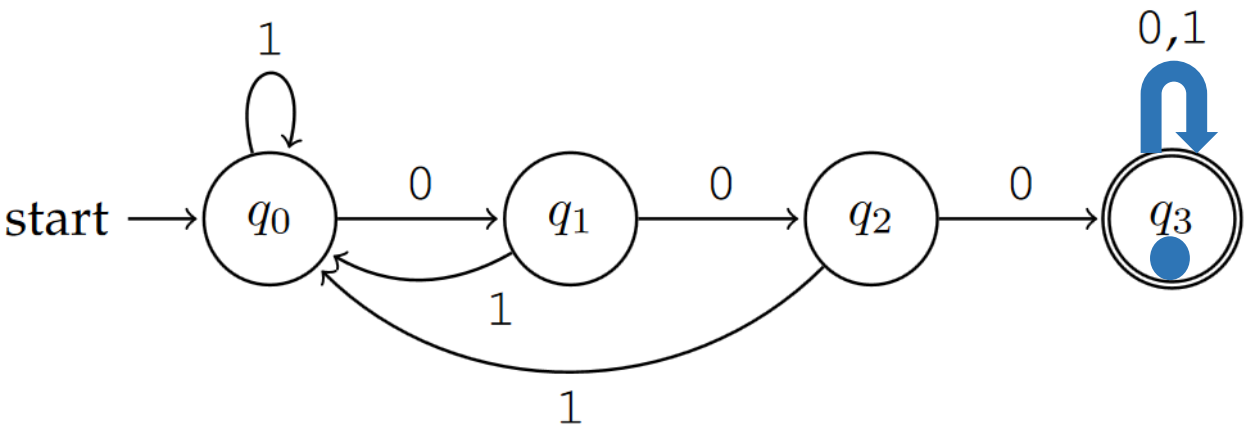
Stringa: 0 1 0 0 0 1




```
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 3
i = 5

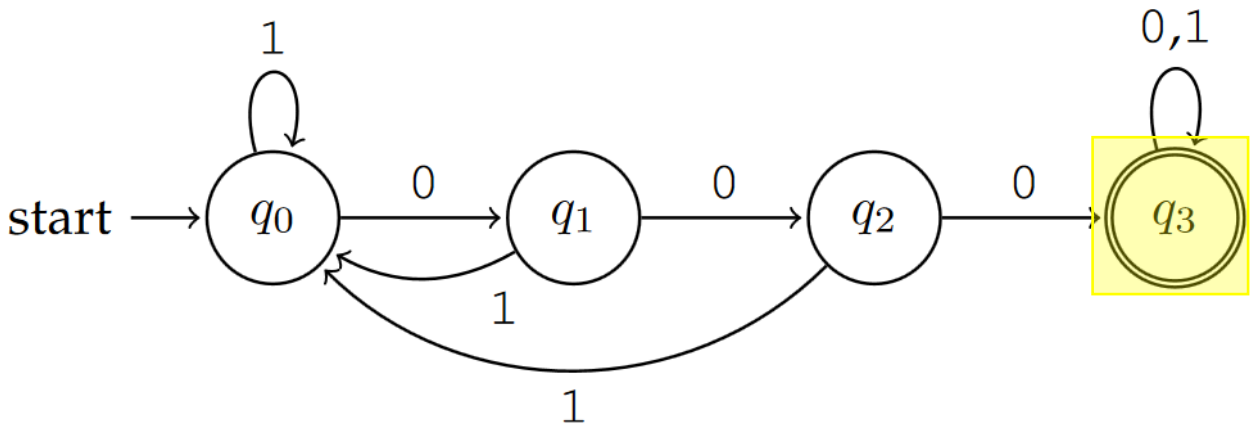
Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

state = 3
i = 6

Stringa: 0 1 0 0 0 1




```java
public static boolean scan(String s)
{
    int state = 0;
    int i = 0;

    while (state >= 0 && i < s.length()) {
        final char ch = s.charAt(i++);

        switch (state) {
        case 0:
            if (ch == '0')
                state = 1;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 1:
            if (ch == '0')
                state = 2;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 2:
            if (ch == '0')
                state = 3;
            else if (ch == '1')
                state = 0;
            else
                state = -1;
            break;

        case 3:
            if (ch == '0' || ch == '1')
                state = 3;
            else
                state = -1;
            break;
        }
    }
    return state == 3;
}
```

Sezione 1 - Implementazione di un DFA in Java

- Viene assegnato il valore -1 alla variabile `state` se viene incontrato un simbolo diverso da 0 e 1.
- Il valore -1 rappresenta una condizione di «errore» per il caso in cui l'ultimo simbolo letto non fa parte del alfabeto Σ del DFA.
- Nel caso in cui `state` = -1, il metodo `scan` restituisce subito il valore `false` (non legge altri simboli dal input).
- Si nota che il metodo `scan` restituisce:
  - `true` se l'input (1) consiste solo da simboli che appartiene al alfabeto, e (2) è accettato dal DFA.
  - `false` se l'input (1) consiste solo da simboli che appartiene al alfabeto, e (2) non è accettato dal DFA.
  - `false` se l'input contiene almeno un simbolo che non appartiene al alfabeto.

Sezione 1 - Implementazione di un DFA in Java

- Output del programma: stampa `OK` sul terminale se la stringa di input è accettato da *A*, altrimenti `NOPE`.
- Input: dalla linea di comando, come argomento (tra virgolette):

```
C:\Users\sproston\Desktop\LFT2122\Lab\CommandLine>java TreZeri.java "010"
NOPE

C:\Users\sproston\Desktop\LFT2122\Lab\CommandLine>java TreZeri.java "010001"
OK
```

- Approccio alternativo: modificare il `main` per passare la stringa di input direttamente a `scan`

```
public static void main(String[] args) {
    System.out.println(scan("010") ? "OK" : "NOPE");
    System.out.println(scan("010001") ? "OK" : "NOPE");
}
```

```
C:\Users\sproston\Desktop\LFT2122\Lab\CommandLine>java TreZeri.java
NOPE
OK
```

Sezione 1 - Implementazione di un DFA in Java

- Dall'esercizio 1.2 in avanti, gli esercizi chiedono di:
  - progettare un DFA;
  - implementare il DFA utilizzando l'approccio descritto tramite l'esempio di `TreZeri;`
  - testare il funzionamento della implementazione su un insieme significativo di esempi (non solo quelli scritti nel documento degli esercizi!).
- Alcuni esercizi richiedono che i DFA devono essere minimi: per il momento si può ignorare quella parte degli esercizi.