Corso di Laurea Ingegneria Informatica

Fondamenti di Informatica 1

Dispensa E09

Esercizi su Array

C. Limongelli

Novembre 2007

Contenuti

- **Esercizi:**
  - **Lettura e stampa di un array**
  - **Massimo elemento di un array**
  - **Verifica sequenza crescente**
  - **Verifica esistenza coppia elementi uguali**
  - **Inverso di una sequenza**
  - **Elemento più frequente in una sequenza**

Lettura e stampa di un array…

- Scrivere un programma che legge da input 5 elementi interi, li memorizza in un array e stampa il contenuto dell'array

```
import fiji.io.*;
class LeggiScrivi{

 public static void main(String[] args){
  final int L=5;
  int[] a;
  a = new int[L];
  leggi(a);
  System.out.println("** verifica acquisizione dati **");
  stampa(a);
  }
```

…Lettura e stampa di un array

```
  public static void leggi(int[] v){
      int i;
      System.out.println("scrivi " + v.length + “
            interi");
      for (i=0; i<v.length; i++)
            v[i] = Lettore.in.leggiInt();
      System.out.println();
  }

  public static void stampa(int[] v){
      int i;
      System.out.println("array letto: ");
      for (i=0; i<v.length; i++)
            System.out.print(v[i] + " ");
      System.out.println();
  }
}
```

Esempio d’esecuzione

```
scrivi 5 interi
2 4 6 8 10

** verifica acquisizione dati **
array letto:
2 4 6 8 10
Press any key to continue . . .
```

Massimo elemento in un array

- **Scrivere un metodo int massimo(int[ ] a) che calcola e restituisce il valore del massimo elemento di un array di interi non vuoto a**
  - **Massimo(new int[ ] { 3, 1, 2 }) deve restituire 3**
  - **Massimo(new int[ ] { 3, 7, 5 }) deve restituire 7**
- **Scrivere un metodo int posizioneMassimo(int[ ] a) che calcola la posizione dell'elemento di valore massimo di a**
  - **posizioneMassimo(new int[ ] { 3, 1, 2 }) deve restituire 0**
  - **posizioneMassimo(new int[ ] { 3, 1, 3 }) può restituire 0 o 2**

Considerazioni

- Se calcoliamo solo l'elemento massimo non sappiamo in quale posizione si trova,
- Se calcoliamo la posizione dell'elemento massimo, sappiamo accedere al valore relativo
- Calcolare la posizione in cui si trova l'elemento massimo è un problema più generale e premette di risolvere entrambi gli esercizi

MassimoArray: verifica di correttezza . . .

```
/* array con 1 solo elemento */
v = new int[] {1}; stampa(v);
System.out.println("Il massimo e' in posizione
  d'indice 0 -- ris. =" + massimo(v));

/* array con l'ultimo elemento massimo */
v = new int[] {1,2,3,4,5}; stampa(v);
System.out.println("Il massimo e' in posizione
  d'indice 4 -- ris. =" + massimo(v));

/* array con il primo elemento massimo */
v = new int[] {5,4,3,2,1}; stampa(v);
System.out.println("Il massimo e' in posizione
  d'indice 0 -- ris. =" + massimo(v));
```

. . . MassimoArray: verifica di correttezza

```
/* array con valori tutti uguali */

v = new int[] {5,5,5}; stampa(v);
System.out.println("Il massimo e' in posizione
  d'indice 0 -- ris. =" + massimo(v));

/* array con un generico elemento massimo */

v = new int[] {5,4,1,6,2,3}; stampa(v);
System.out.println("Il massimo e' in posizione
  d'indice 3 -- ris. =" + massimo(v));
```

MassimoArray: analisi del problema

- **Che tipo di problema è?**
- **Algoritmo:**
  - **Input: array v**
  - **Output: posizione del massimo trovato**
  - **Pre: array non vuoto (almeno un elemento)**
  - **Memorizzo il valore e l'indice della prima componente in due variabili max e indmax;**
  - **Esamino tutti i rimanenti elementi dell'array con un ciclo for: sia i l'indice del generico elemento visitato: se v[i]>max allora aggiorno max e indmax rispettivamente con v[i] e i**

MaxEl: il metodo…

```
public static int massimo(int[]  v){
    //pre: array non vuoto - almeno 1 elemento
    int max, indmax;
    int i;
          max = v[0];
          indmax = 0;
          for (i=1; i<v.length; i++)
                if (v[i] > max){
                      max = v[i];
                      indmax = i;
                }
    return indmax;
}
```

Esempio di esecuzione

```
array letto:
1
Il massimo e' in posizione d'indice 0 -- ris. =0
array letto:
1 2 3 4 5
Il massimo e' in posizione d'indice 4 -- ris. =4
array letto:
5 4 3 2 1
Il massimo e' in posizione d'indice 0 -- ris. =0
array letto:
5 5 5
Il massimo e' in posizione d'indice 0 -- ris. =0
array letto:
5 4 1 6 2 3
Il massimo e' in posizione d'indice 3 -- ris. =3
Press any key to continue . . .
```

Verifica Sequenza crescente

- **Scrivere un metodo boolean crescente(int[ ] a) che verifica se a è ordinato in modo crescente**
  - un array è ordinato in modo crescente se per ogni indice k, l'elemento di indice k è maggiore di tutti gli elementi di indice minore di k
  - è sufficiente verificare se ogni elemento è minore dell'elemento che lo segue immediatamente
  - intuitivamente, bisogna confrontare ciascun elemento di indice i dell'array con l'elemento che lo segue immediatamente, quello di indice i+1
  - attenzione – l'ultimo elemento di un array non ha un elemento che lo segue immediatamente

Sequenza crescente: verifica di correttezza

```
public static void testCrescente(){
   int[] v;

   /* array vuoto */
   v = new int[] {};
   stampa(v);
   System.out.println("La sequenza e' crescente TRUE ="
       + crescente(v));

   /* array di due elementi crescente*/
   v = new int[] {1,2};
   stampa(v);
   System.out.println("La sequenza e' crescente TRUE  ="
       + crescente(v));

                              . . .
```

Sequenza crescente: verifica di correttezza

```
                    . . .
/* array di due elementi non crescente */
  v = new int[] {2,1};
  stampa(v);
  System.out.println("La sequenza NON e'
     crescente FALSE =" + crescente(v));

/* array con l'ultimo elemento non crescente
  */
  v = new int[] {1,2,3,4,1};
  stampa(v);
  System.out.println("La sequenza NON
     e'crescente FALSE =" + crescente(v));

                    . . .
```

… Sequenza crescente: verifica di correttezza

```
                                . . .

/* array con il secondo elemento non crescente */
   v = new int[] {1,-2,3,4,5};
   stampa(v);
   System.out.println("La sequenza NON e'crescente
       FALSE =" + crescente(v));

 /* array  crescente */
    v = new int[] {1,2,3,4,5};
    stampa(v);
    System.out.println("La sequenza  e'crescente
       TRUE =" + crescente(v));
}
```

Sequenza crescente: analisi del problema

- **Che tipo di problema è? Verifica universale**
- **Algoritmo:**
  - **Input: array v**
  - **Output: boolean (true/false) (crescente/non crescente)**
  - **Pre: array non nullo**
  - **verifica universale:**

devo verificare che v[i]<v[i+1]

per tutti gli i da 0 fino al penultimo (v.length-1)

Sequenza crescente: il metodo

```
public static boolean crescente(int[]  v){
    //pre: array non vuoto
    boolean crescente;
    int i;
    /* verifica universale: devo verificare che v[i]<v[i+1]
       per tutti gli i da 0 fino al penultimo */
          crescente=true;
          i=0;
          while (i<v.length - 1 && crescente){
                if (v[i]>=v[i+1])
                      crescente = false;
                i=i+1;
                }
    return crescente;
}
```

Sequenza crescente: esempio d'esecuzione

```
array letto:

La sequenza e' crescente TRUE =true
array letto:
3
La sequenza e' crescente TRUE =true
array letto:
1 2
La sequenza e' crescente TRUE  =true
array letto:
2 1
La sequenza NON e' crescente FALSE =false
array letto:
1 2 3 4 1
La sequenza NON e'crescente FALSE =false
array letto:
1 -2 3 4 5
La sequenza NON e'crescente FALSE =false
array letto:
1 2 3 4 5
La sequenza  e'crescente TRUE =true
Press any key to continue . . .
```

Verifica esistenza coppia elementi uguali

- **Scrivere un metodo boolean coppiaUguali(int[ ] a) che verifica se a contiene almeno una coppia di elementi uguali**
  - **bisogna confrontare ciascun elemento X dell'array con ogni altro elemento Y dell'array**
    - **dove X e Y sono elementi diversi, nel senso che hanno indici diversi**
  - **bisogna confrontare ciascun elemento di indice i dell'array con ogni altro elemento di indice j dell'array - dove i e j hanno valore diverso**
  - **quindi è necessario utilizzare due variabili indice per accedere agli elementi di uno stesso array**

Coppia Uguali: verifica di correttezza ...

```
public static void testCoppiaUguali(){
      int[] v;

      System.out.println("*** TEST Coppia Uguali ***");
      /* array con 2 elementi uguali */
      v = new int[] {1,1};
      stampa(v);
      System.out.println("true  " + coppiaUguali(v));

      /* array con due elementi distinti */
      v = new int[] {2,3};
      stampa(v);
      System.out.println("false  " + coppiaUguali(v));

                              . . .
```

Coppia Uguali: verifica di correttezza ...

```
                                    . . .
/* array con 3 elementi tutti uguali */
      v = new int[] {3,3,3};
      stampa(v);
      System.out.println("true  " + coppiaUguali(v));

/* array con 3 elementi in cui il primo e
                           l'ultimo sono uguali */
      v = new int[] {3,2,3};
      stampa(v);
      System.out.println("true  " + coppiaUguali(v));

/* array con diversi elementi tutti distinti */
      v = new int[] {3,1,2,5,4,0};
      stampa(v);
      System.out.println("false  " + coppiaUguali(v));
```

… Coppia Uguali: verifica di correttezza

```
                                        . . .
/* array con diversi elementi in cui gli ultimi due
              sono uguali */
        v = new int[] {3,2,1,5,6,3};
        stampa(v);
        System.out.println("true  " + coppiaUguali(v));

/* array con diversi elementi in cui i primi due
              sono uguali */
        v = new int[] {4,4,6,2,3,1};
        stampa(v);
        System.out.println("true  " + coppiaUguali(v));

/* array con diversi elementi in cui due sono uguali */
        v = new int[] {5,4,2,0,23,1,2,7,8};
        stampa(v);
        System.out.println("true  " + coppiaUguali(v));
}
```

Coppia Uguali: analisi del problema

- **Che tipo di problema è? Verifica esistenziale**
- **Algoritmo:**
  - **Input: array v**
  - **Output: boolean coppiaTrovata (true/false) esiste/non esiste**
  - **pre: array non vuoto - almeno 2 elementi**

    oppure: nel caso sia vuoto o composto da 1 o 0 elementi il metodo restituisce false

  - **Esamino la sequenza: inizialmente coppiaTrovata=false, mi fermo appena ho trovato due elementi uguali o ho terminato di controllare tutti gli elementi**
  - **devo confrontare ogni elemento con tutti gli altri tranne se stesso**
  - **esamino ogni elemento dell'array finche' non trovo una coppia di uguali o ho terminato di controllare tutti gli elementi**

Coppia Uguali: raffinamento dell'algoritmo

```
coppiaTrovata=false
i=0;
```

- mi fermo appena ho trovato due elementi uguali o ho terminato di controllare tutti gli elementi

```
while (i<v.length && !coppiaTrovata){
```

- devo confrontare ogni elemento in posizione i-esima con tutti gli altri tranne se stesso
- i: indice dell'elemento fissato,
- j: indice dell'elemento con cui lo confronto

```
  j=0;
```

- mi fermo appena ho trovato due elementi uguali o ho terminato di controllare tutti gli elementi

```
  while (j<v.length && !coppiaTrovata){
```

- se gli indici sono diversi, ma i rispettivi elementi sono uguali ho trovato una coppia

```
        if (i != j && v[i]==v[j])
              coppiaTrovata=true;
        j=j+1;}
  i=i+1;}
```

Coppia Uguali: il metodo . . .

```
public static boolean coppiaUguali(int[]  v){
  //pre: array non vuoto - almeno 2 elementi

  /* oppure: nel caso sia vuoto o composto da 1 o 0
  elementi il metodo restituisce false */

  int i,j;
  boolean coppiaTrovata;
  /* verifica esistenziale: inizialmente
      coppiaTrovata=false,
      mi fermo appena ho trovato due elementi uguali */
  coppiaTrovata = false;
  /* devo confrontare ogni elemento con tutti gli altri
      tranne se stesso */
  /* esamino ogni elemento dell'array finche' non trovo
      una coppia di uguali o ho terminato di
      controllare tutti gli elementi */
                          . . .
```

. . . Coppia Uguali: il metodo

```
                                . . .
i=0;
while (i<v.length && !coppiaTrovata){

/* generico confronto dell'elemento in posizione
d'indice i, con tutti gli altri */

    j=0;
    while (j<v.length && !coppiaTrovata){
          if (i != j && v[i]==v[j])
                coppiaTrovata=true;
                j=j+1;}

    i=i+1;}

return coppiaTrovata;}
```

Coppia uguali: esempio d'esecuzione

```
*** TEST Coppia Uguali ***
array letto:
1 1
true  true
array letto:
2 3
false  false
array letto:
3 3 3
true  true
array letto:
3 2 3
true  true
```

```
array letto:
3 1 2 5 4 0
false  false
array letto:
3 2 1 5 6 3
true  true
array letto:
4 4 6 2 3 1
true  true
array letto:
5 4 2 0 23 1 2 7 8
true  true
Press any key to continue . . .
```

Inverso di una sequenza

- Scrivere un metodo int[ ] inverso(int[ ] a) che crea e restituisce un array che contiene gli stessi elementi di a, ma disposti in ordine inverso
- Osservazioni:
  - Il metodo non deve modificare a, ma creare un nuovo array: v
  - Input: a
  - Output: v
  - Algoritmo: il primo el. di a va memorizzato come ultimo in v, il secondo come penultimo, … l'ultimo come primo

Inverso di una sequenza: verifica di correttezza …

```
public static void testInversoArray(){
     int[] v;

     System.out.println("*** TEST Inverso Array ***");

     /* array vuoto */
     v = new int[] {};
     stampa(v);
     System.out.print("inverso "); stampa(inverso(v));

     /* array con 1 elemento */
     v = new int[] {1};
     stampa(v);
     System.out.print("inverso "); stampa(inverso(v));

     /* array con 2 elementi */
     v = new int[] {1,2};
     stampa(v);
     System.out.print("inverso "); stampa(inverso(v));

     /* array con tre elementi*/
     v = new int[] {1,2,3};
     stampa(v);
     System.out.print("inverso "); stampa(inverso(v));

     /* array con un generico numero di elementi */
     v = new int[] {1,2,3,4,5,6,7,8,9};
     stampa(v);
     System.out.print("inverso "); stampa(inverso(v));
}
```

Inverso di una sequenza: il metodo

```
public static int[] inverso(int[]  v){
    //pre: array non nullo
    int[] inv; // inv e' il nuovo array
    int i;
    int l; // per memorizzare la lunghezza dell'array

    l = v.length;
    inv = new int[l];

    for(i=0; i<l; i++)
          inv[i] = v[l-i-1];

    return inv;
}
```

Inverso: esempio d'esecuzione

```
*** TEST Inverso Array ***
array letto:

inverso array letto:

array letto:
1
inverso array letto:
1
array letto:
1 2
inverso array letto:
2 1
array letto:
1 2 3
inverso array letto:
3 2 1
array letto:
1 2 3 4 5 6 7 8 9
inverso array letto:
9 8 7 6 5 4 3 2 1
Press any key to continue . . .
```

Inverso: modifica

- Modificare il programma in modo che l'array, invece di essere creato, viene modificato.

Elemento più frequente

- **Scrivere un metodo che, dato un array a, calcola e restituisce l'elemento più frequente di a**
  - **calcolare prima l'array occ delle frequenze degli elementi di a**
  - **l'elemento da restituire è quello che in a occupa la posizione del massimo di occ**

Elemento più frequente: verifica di correttezza

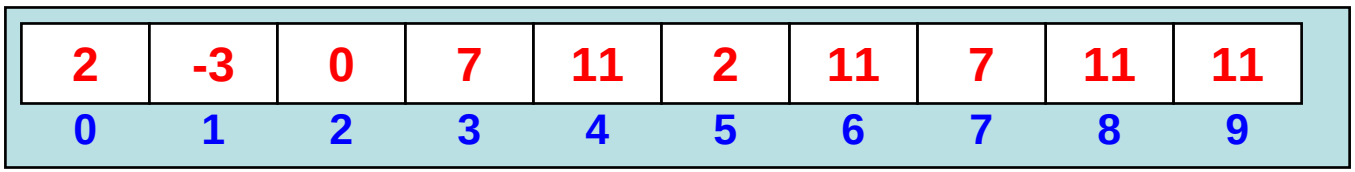
```
public static void testElementoPiuFrequente(){
   int[] v;

   System.out.println("*** TEST Elemento Piu' Frequente ***");

   /* array con 1 elemento */
   v = new int[] {1};
   stampa(v);
   System.out.println("elemento piu' frequente 1 = "+ piuFrequente(v));
   System.out.println();

   /* array con 2 elementi distinti */
   v = new int[] {1,2};
   stampa(v);
   System.out.println("elemento piu' frequente 1 = "+ piuFrequente(v));
   System.out.println();

   /* array con due elementi uguali */
   v = new int[] {2,2};
   stampa(v);
   System.out.println("elemento piu' frequente 2 = "+ piuFrequente(v));
   System.out.println();
```

Elemento più frequente: verifica di correttezza

```
/* array con tre elementi di cui due uguali */
v = new int[] {1,2,1};
stampa(v);
System.out.println("elemento piu' frequente 1 = " + piuFrequente(v));
System.out.println();

/* array con diversi elementi */
v = new int[] {1,2,3,1,2,3,1,2,3,3};
stampa(v);
System.out.println("elemento piu' frequente 3 = " + piuFrequente(v));
System.out.println();

/* array con diversi elementi */
v = new int[] {1,2,3,1,2,3,1,2,3,3,5,4,2,5,4,3,1};
stampa(v);
System.out.println("elemento piu' frequente 3 = " + piuFrequente(v));
System.out.println();

/* array con diversi elementi */
v = new int[] {10,2,10,1,10,10,1,2,10,10,5,10,10,5,10,10,1};
stampa(v);
System.out.println("elemento piu' frequente 10 = " + piuFrequente(v));
System.out.println(); }
```

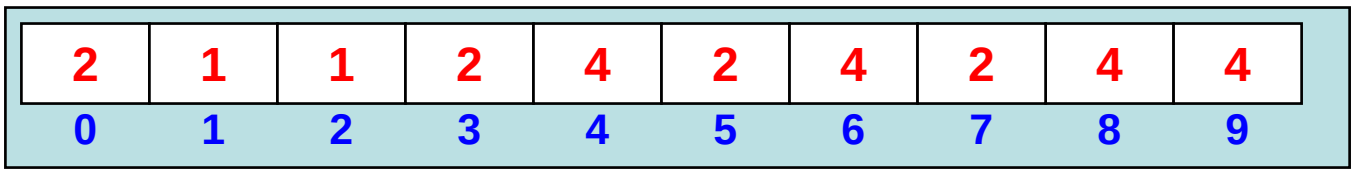
Elemento più frequente: analisi del problema

- Calcolare l'array occ delle frequenze di tutti gli elementi di a
- Esempio:
  - a



2	-3	0	7	11	2	11	7	11	11
0	1	2	3	4	5	6	7	8	9

  - occ



2	1	1	2	4	2	4	2	4	4
0	1	2	3	4	5	6	7	8	9

  - Calcolo la frequenza di ogni elemento di a, non tenendo conto di possibili elementi uguali (di cui calcolo nuovamente la frequenza)

Elemento più frequente: calcolo della frequenza di un elemento in un array

```
// metodo che calcola la frequenza di un elemento
// all'interno dell'array v

public static int freq(int[] v,int a){
    int cont;
    int i;
    cont=0;
    for(i=0; i<v.length;i++){
          if(v[i]==a) cont++;
    }

    return cont;
}
```

Elemento più frequente: creazione dell'array delle occorrenze

- Inizializzo il nuovo array con la stessa lunghezza di **v**

```
occ=new int[v.length];
```

- Per ogni elemento di **v** calcolo la frequenza e la memorizzo nel corrispondente indice di **occ**

```
for(i=0;i<v.length;i++){
    occ[i]=freq(v,v[i]);
}
```

- Calcolo il massimo di **occ**, ricordando a quale elemnto in **v** corrisponde quella frequenza

Elemento più frequente: calcolo della frequenza massima

max: frequenza massima
vmax: elemento di v a cui corrisponde la freq. massima
pmax: indice di occ di frequenza massima
Inizializzazioni:

```
max=0;
vmax=0;
pmax=0;
```

Confronto ogni elemento di **occ** con **max**

```
for(i=0;i<occ.length;i++){
```

Se la frequenza di **occ[i]** è maggiore del **max** allora aggiorno il **max**, **vmax** e **pmax**

```
if(occ[i]>max){
        max=occ[i];
        vmax=v[i];
        pmax=i;
}
```

Elemento più frequente: il metodo . . .

```
/* Metodo che calcola e restituisce
        l'elemento più frequente in v */

public static int piuFrequente(int[] v){
   int[] occ;
   int i;
   int max,pmax,vmax;

   occ=new int[v.length];

// calcola le frequenze degli elementi di v

   for(i=0;i<v.length;i++){
        occ[i]=freq(v,v[i]);
   }
```

. . . Elemento più frequente: il metodo

```
    max=0;
    vmax=0;
    pmax=0;
    for(i=0;i<occ.length;i++){
         if(occ[i]>max){
              max=occ[i];
              vmax=v[i];
              pmax=i;
         }
    }
    return vmax; // return v[pmax];
}
```

Elemento più frequente: esempio d'esecuzione

```
*** TEST Elemento Piu' Frequente ***
array letto:
1
elemento piu' frequente 1 = 1

array letto:
1 2
elemento piu' frequente 1 = 1

array letto:
2 2
elemento piu' frequente 2 = 2

array letto:
1 2 1
elemento piu' frequente 1 = 1

array letto:
1 2 3 1 2 3 1 2 3 3
elemento piu' frequente 3 = 3

array letto:
1 2 3 1 2 3 1 2 3 3 5 4 2 5 4 3 1
elemento piu' frequente 3 = 3

array letto:
10 2 10 1 10 10 1 2 10 10 5 10 10 5 10 10 1
elemento piu' frequente 10 = 10

Press any key to continue . . .
```

Elemento più frequente: nuovo algoritmo

- Perché bisogna calcolare la frequenza delle occorrenze di un elemento se è già stato esaminato?

- a

2	-3	0	7	11	2	11	7	11	11
0	1	2	3	4	5	6	7	8	9

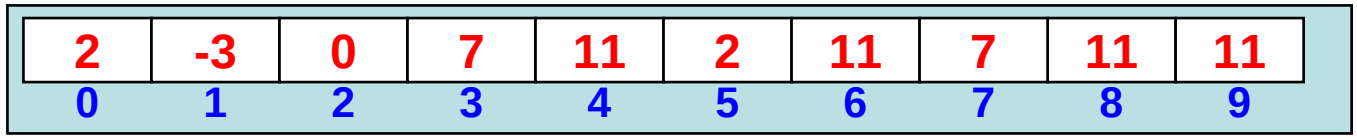
- occ

2	1	1	2	4	2	4	2	4	4
0	1	2	3	4	5	6	7	8	9

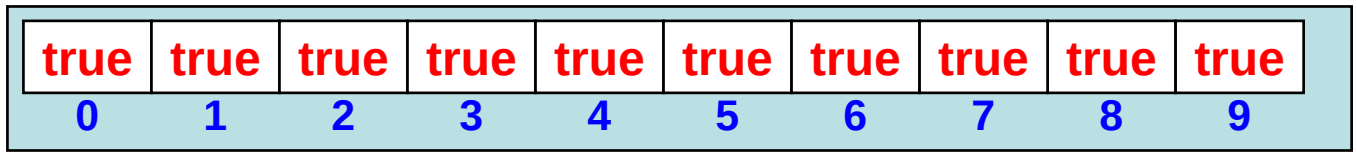
- Si devono ricordare gli elementi di cui si sono già calcolate le occorrenze, come?
- Con un array di booleani in cui ogni elemento è true se l'occorrenza è ancora da calcolare, false altrimenti

… Elemento più frequente: nuovo algoritmo

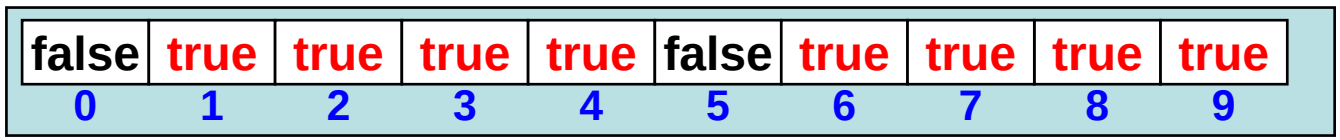
- **Esempio: sia esamina il nome dell'array di booleani**
  - **a**



  - **esamina all'inizio**



  - **Dopo che è stato analizzato il primo elemento di a**



  - **Nel seguito la componente d'indice 5 si potrà saltare perché già esaminata**
  - **Il calcolo della frequenza massima viene eseguito man mano che si calcola una nuova frequenza**

Elemento più frequente: nuovo metodo…

```
public static int piuFrequente(int[]  v){
    //pre: array non nullo
    int[] inv;
    boolean[] esamina;
    int i,k,l;
    int occ, maxOcc,app,elMaxFreq;

/* per ogni elemento calcolo il numero di
          occorrenze nell'array */
/* a parita' di elemento ci sara' la stessa
          occorrenza */
/* posso migliorare l'esame dell'array con un array
    di booleani che mi dice
    se l'elemento e' da considerare o no:
    inizialmente tutte le componenti sono
    inizializzate a true (da esaminare) false
    (da non considerare perche' gia' esaminato) */
                        . . .
```

Elemento più frequente: nuovo metodo…

```
. . .

l = v.length;
esamina = new boolean[l];

/* inizializzo l'array di booleani */
for(i=0; i<l; i++)
     esamina[i] = true;

/* inizializzo maxocc ed elMaxFreq*/
maxOcc = 0;
elMaxFreq = v[0];

. . .
```

…Elemento più frequente: nuovo metodo

```
    /* considero l'elemento j-esimo: man mano che
calcolo le occorrenze metto a false gli elementi
uguali le cui occorrenze sono sommate in occ*/
    for (i=0; i<l; i++){
       if (esamina[i]){
          occ = 1;
          app = v[i];
          for (k=i+1; k<l; k++)
                if (esamina[k] && v[k]==app){
                      occ++;
                      esamina[k]=false;
                }
          if (occ> maxOcc){
                maxOcc = occ;
                elMaxFreq = app;
          }
       }
    }
    return elMaxFreq; }
```