Lab. Programmazione (CdL Informatica) & Informatica (CdL Matematica) a.a. 2022-23

Monica Nesi

Università degli Studi dell'Aquila

23 Novembre 2022

Ancora metodi statici ricorsivi: almeno k elementi con una data proprietà

Scriviamo soluzioni ricorsive per altri problemi che abbiamo già risolto in modo iterativo.

Scrivere un metodo ricorsivo che, dati un array di interi `a` ed un intero `k`, restituisce `true` se in `a` compaiono *almeno* `k` numeri strettamente positivi, `false` altrimenti.
Si assuma `k>0`.

Metodo iniziale:

```
public static boolean almenokPosR(int[] a,
int k) {
  return almenokPosR(a,k,0,0);
}
```

Almeno k elementi con una data proprietà (cont.)

Metodo ricorsivo

```
public static boolean almenokPosR(int[] a,
int k, int i, int cont) {
  if (cont >= k)
    return true;
  if (i == a.length)
    return false;
  if (a[i] > 0)
    cont++;
  return almenokPosR(a,k,i+1,cont);
}
```

Le condizioni soddisfatte quando si esegue il comando condizionale `if (a[i] > 0)` sono: `0 <= i < a.length` e `0 <= cont < k`, ovvero le condizioni nella guardia booleana del ciclo nella soluzione iterativa corrispondente.

Verificare l'occorrenza di un elemento in ogni riga di un array bidimensionale

Scrivere un metodo ricorsivo che, dati un array bidimensionale di interi `a` ed un intero `x`, restituisce `true` se `x` occorre in *ogni riga* di `a`, `false` altrimenti.

Ricordiamo che, come nella soluzione iterativa, si richiede di:

- passare alla riga successiva *non appena* si trova un'occorrenza di `x` nella riga in esame;

- terminare il metodo restituendo `false` *non appena* si trova una riga che non contiene alcuna occorrenza di `x` (i.e., tale riga non soddisfa la condizione richiesta).

N.B. Con la ricorsione non vi è alcuna necessità di una variabile booleana (e.g. `trovato`) per passare alla riga successiva *non appena* si verifica che la riga in esame soddisfa la condizione richiesta.

Verificare l'occorrenza di un elemento in ogni riga di un array bidimensionale (cont.)

```
public static boolean occorreBiOgniRigaR
(int[][] a, int x) {
  return occorreBiOgniRigaR(a,x,0,0);
}

public static boolean occorreBiOgniRigaR
(int[][] a, int x, int i, int j) {
  if (i == a.length)
    return true;
  if (j == a[i].length)
    return false;
  if (a[i][j] == x)
    return occorreBiOgniRigaR(a,x,i+1,0);
  return occorreBiOgniRigaR(a,x,i,j+1);
}
```

Almeno k elementi con una data proprietà in ogni riga

Scrivere un metodo ricorsivo che, dati un array bidimensionale di interi `a` ed un intero `k`, restituisce `true` se in *ogni riga* `a[i]` di `a` esistono *almeno* `k` numeri strettamente positivi, `false` altrimenti. Si assuma `k>0`.

Vediamo una soluzione che non invoca il metodo `almenokPosR` sulle righe di `a`.

Metodo iniziale:

```
public static boolean almenokPosBiR(int[][] a,
int k) {
  return almenokPosBiR(a,k,0,0,0);
}
```

Almeno k elementi con una data proprietà in ogni riga (cont.)

Metodo ricorsivo

```
public static boolean almenokPosBiR(int[][] a,
int k, int i, int j, int cont) {
  if (i == a.length)
    return true;
  if (cont >= k)
    return almenokPosBiR(a,k,i+1,0,0);
  if (j == a[i].length)
    return false;
  if (a[i][j] > 0)
    cont++;
  return almenokPosBiR(a,k,i,j+1,cont);
}
```

Metodi statici ricorsivi che restituiscono array

Nei metodi statici iterativi che restituiscono un array monodim. o bidim., abbiamo visto che l'array da restituire deve essere

- dichiarato e creato all'inizio del corpo del metodo,
- ai suoi elementi devono essere assegnati dei valori secondo la specifica del problema,
- infine va restituito il riferimento a tale array.

Ciò vale anche per le soluzioni ricorsive.
Nel nostro approccio alle soluzioni ricorsive abbiamo *due metodi*, il primo non ricorsivo ed il secondo ricorsivo.

N.B. L'array da restituire deve essere *dichiarato e creato nel primo metodo* e poi *passato al metodo ricorsivo* insieme agli altri parametri.

Versione ricorsiva del metodo `lunghezze`

Scrivere un metodo ricorsivo che, dato un array monodimensionale di stringhe a, restituisce un array monodimensionale di interi b tale che l'elemento b[i] è la lunghezza della stringa a[i].

```
public static int[] lunghezzeR (String[] a) {
  int[] b = new int[a.length];
  return lunghezzeR(a,b,0);
}

public static int[] lunghezzeR (String[] a,
int[] b, int i) {
  if (i == a.length)
    return b;
  b[i] = a[i].length();
  return lunghezzeR(a,b,i+1);
}
```

Versione ricorsiva del reverse di un array

Scrivere un metodo ricorsivo che, dato un array monodimensionale di interi a, restituisce un *nuovo* array monodimensionale di interi con gli elementi di a rovesciati.

```
public static int[] reverseR (int[] a) {
  int[] b = new int[a.length];
  return reverseR(a,b,0);
}

public static int[] reverseR (int[] a,
int[] b, int i) {
  if (i == a.length)
    return b;
  b[i] = a[a.length-1-i];
  return reverseR(a,b,i+1);
}
```

Generare i prefissi di una stringa in modo ricorsivo

Scrivere un metodo ricorsivo che, data una stringa s, restituisce un array monodimensionale di stringhe contenente tutti i *prefissi* di s in *ordine crescente* (inclusa s).

```
public static String[] prefissiR(String s) {
  String[] a = new String[s.length()];
  return prefissiR(s,a,0);
}

public static String[] prefissiR(String s,
String[] a, int i) {
  if (i == s.length())
    return a;
  a[i] = s.substring(0,i+1);
  return prefissiR(s,a,i+1);
}
```

Metodi ricorsivi che restituiscono un array bidimensionale

Come nel caso monodimensionale, l'array bidimensionale da restituire deve essere *dichiarato e creato nel primo metodo* e poi *passato al metodo ricorsivo* insieme agli altri parametri.

Una volta noto l'indice `i` delle singole righe dell'array bidim., ogni riga è creata nel metodo ricorsivo prima di scorrerla (e.g., quando l'indice con cui si scorre la riga vale 0).

Scrivere un metodo ricorsivo che, dato un array monodimensionale di stringhe a, restituisce un array bidimensionale di caratteri c tale che la riga `c[i]` contiene i caratteri della stringa `a[i]` (da sx a dx).

Metodo iniziale:

```
public static char[][] caratteriR(String[] a) {
  char[][] c = new char[a.length][];
  return caratteriR(a,c,0,0);
}
```

Metodi ricorsivi che restituiscono un array bidimensionale (cont.)

Metodo ricorsivo

```
public static char[][] caratteriR(String[] a,
char[][] c, int i, int j) {
  if (i == c.length)
    return c;
  if (j == 0)
    c[i] = new char[a[i].length()];
  if (j == c[i].length)
    return caratteriR(a,c,i+1,0);
  c[i][j] = a[i].charAt(j);
  return caratteriR(a,c,i,j+1);
}
```

Le lunghezze di un array bidim. di stringhe

Scrivere un metodo ricorsivo che, dato un array bidimensionale di stringhe a, restituisce un array bidimensionale di interi b contenente le lunghezze degli elementi di a nelle posizioni corrispondenti.

Vediamo una soluzione che non invoca il metodo `lunghezzeR` sulle righe di a.

Metodo iniziale:

```
public static int[][] lunghezzeBiR
(String[][] a) {
  int[][] b = new int[a.length][];
  return lunghezzeBiR(a,b,0,0);
}
```

Le lunghezze di un array bidim. di stringhe (cont.)

Metodo ricorsivo

```
public static int[][] lunghezzeBiR
(String[][] a, int[][] b, int i, int j) {
  if (i == a.length)
    return b;
  if (j == 0)
    b[i] = new int[a[i].length];
  if (j == a[i].length)
    return lunghezzeBiR(a,b,i+1,0);
  b[i][j] = a[i][j].length();
  return lunghezzeBiR(a,b,i,j+1);
}
```

Secondo Parziale 30/11/2018

Esercizio 2.
Scrivere un metodo che, dati un array bidimensionale di interi a, un array monodimensionale di interi b ed un intero k, restituisce *true* se in *ogni* riga $a[i]$ di a esistono *almeno* k numeri minori o uguali a $b[i]$, altrimenti il metodo restituisce *false*.
Esempio: se $a =$ `{{7,3,-2,2,9},{5,0,8,3},{15,11,-2}}`, $b =$ `{6,3,11}` e $k = 2$, il metodo restituisce *true*.
N.B. i) Occorre restituire *false* non appena si trova una riga $a[i]$ che non soddisfa la condizione richiesta. ii) Occorre passare alla riga successiva non appena si verifica che una riga $a[i]$ soddisfa la condizione richiesta. iii) Si assuma che gli array a e b abbiano la stessa lunghezza e $k > 0$.

Esercizio 3.
Scrivere una versione ricorsiva del metodo dato per risolvere l'Es.2.

Secondo Parziale 30/11/2018 (cont.)

Esercizio 4.
Scrivere un metodo che, dato un array monodimensionale di stringhe a, restituisce un array bidimensionale di caratteri c tale che la riga $c[i]$ contiene i caratteri della stringa $a[i]$ (letta da sx a dx).
Esempio: se $a =$ `{"dcba","k","","9qr","v3"}`, il metodo restituisce l'array $c =$
`{{'d','c','b','a'},{'k'},{},{'9','q','r'},{'v','3'}}`.

Esercizio 5.
Scrivere una versione ricorsiva del metodo dato per risolvere l'Es.4.

Nota: questi sono i metodi già fatti `caratteri` e `caratteriR`.