Corso di Laurea Ingegneria Informatica
Fondamenti di Informatica 2

Dispensa E04

Esercizi sui problemi di ordinamento

C. Limongelli

Marzo 2008

Contenuti

- Selectioin sort ricorsivo
- Ordinamento lessicografico di stringhe
- Ordinamento di un array di oggetti

Selection sort ricorsivo

- Ordinare una sequenza di interi in ordine non decrescente con la versione ricorsiva del selection sort
- Precondizione:
  - Array di interi non nullo

Algoritmo…

- **Passo base:**
  - La sequenza costituita da un solo elemento è ordinata
- **Passo ricorsivo:**
  - Suppongo che le prime i-1 componenti dell'array siano gia' state ordinate
  - Ordino la porzione di array a partire dalla componente d'indice i:
    - Scambio il contenuto della componente d'indice i con il massimo calcolato sul sotto-array
    - Chiamo ricorsivamente il metodo selectionSort sul sotto-array a partire dalla componente d'indice i+1

…Algoritmo…

- **Sia n la lunghezza dell'array v: n = v.length**
- **Passo base:**
  - La sequenza costituita da un solo elemento è ordinata
  - Quindi non bisogna fare nulla
  - Il passo base diventa la parte else (mancante) dell'istruzione condizionale

```
if (!passo base)
   passo ricorsivo
else
   non fare niente
```

…Algoritmo

Se la porzione di array contiene più di un elemento

allora

Scambio il contenuto della componente d'indice i dell'array v con il minimo calcolato sul sotto-array creato con le componenti di v da i+1 a n

scambia(v,i,min(v,i,n-1))

Ordino la porzione di array a partire dalla componente d'indice i+1:

selectionSortRic(scambia(v,i,min(v,i,n-1)), i+1);

Il metodo selectionSortRic

```
public static void selectionSortRic(int[] v, int i) {
    int n;
    n = v.length;
    if (i < n-1)
       selectionSortRic(scambia(v,i,min(v,i,n-1)), i+1);
    }
```

- Come viene richiamato la prima volta?

```
selectionSortRic(a, 0);
```

- Attenzione: ora min e scambia ritornano rispettivamente un intero e un array di interi

Il metodo principale: alcuni test

```
class SelectionSortRicorsivo{

   public static void main(String[] args){
        int[] a;

        a = new int[] { 2, 1};
        System.out.println("array letto: ");
        stampa(a);
        selectionSortRic(a, 0);
        System.out.println("array ordinato: ");
        stampa(a);
        System.out.println("********************");
        a = new int[] { 1, 5, 0, 8, 5 };
        System.out.println("array letto: ");
        stampa(a);
        selectionSortRic(a, 0);
        System.out.println("array ordinato: ");
        stampa(a);
}
```

Ordinamento lessicografico di stringhe

- **Dato in input un array di stringhe ordinare i suoi elementi secondo l'ordine lessicografico**
- ***Dato in input:***
  - *Input acquisito da tastiera*
  - *Input acquisito da file*
  - *Input generato con un metodo pseudo-casuale*
  - *Input generato ad hoc per effettuare i test*
  - *…*
- ***Supponiamo di generare uno o più array ad hoc per utilizzarli nei test di correttezza del metodo***

Ordinamento lessicografico…

- **Se due stringhe S1 e S2 non rappresentano la stessa sequenza di caratteri allora:**
  - o hanno uno o piu' caratteri in certe posizioni che non corrispondono, ad esempio:
    "cassa" e "casse"
    "cassa" e 'cappa"
  - o hanno lunghezze diverse, ad esempio: "cassa" e "cassaforte"
- **Per determinare l'ordine lessicografico, nel primo caso, prendiamo la prima coppia di caratteri che differiscono e li confrontiamo (usiamo la corrispondente codifica Unicode), mentre nel secondo caso confrontiamo le lunghezze**
  - cassa < forte
  - cassa < casse
  - cassa > cappa
  - cassa < cassaforte

Ordinamento lessicografico…

- **Esempi:**
  - cassa < forte
  - cassa = cassa
  - cassa < casse
  - cassa > cappa
  - cassa < cassaforte
- **In generale:**
  - Si confrontano i primi caratteri delle due stringhe
    - Se sono uguali si passa ad esaminare i due caratteri successivi
    - Altrimenti si ha un criterio per decidere il carattere minore nell'ordinamento lessicografico

…Ordinamento lessicografico…

- **Siano s1 e s2 due stringhe da confrontare:**
  - Se s1="" e s2 = "" allora s1 = s2
  - Se s1="" e s2 ≠ "" allora s1 < s2
  - Se s1≠ "" e s2 = "" allora s1 > s2
  - Altrimenti, si confrontano i primi due caratteri delle stringhe c1 e c2:
    - Se c1 > c2 allora s1 > s2
    - Se c1 < c2 allora s1 < s2
    - Se c1 = c2 allora si esamina l'ordine lessicografico delle sottostringhe ottenute da s1 e s2 avendo tolto il primo carattere da entrambe

Codifica dell'ordinamento lessicografico

```
/* true se s1 e maggiore (lessicograficamente) di 2 */
public static boolean maggiore(String s1, String s2){
 boolean res;
 if(s1.length()==0 && s2.length()==0) res = true;
 else if(s1.length()==0 && s2.length()>0) res = false;
 else if(s1.length()>0 && s2.length()==0) res = true;
 else {/* caso generale:stringhe entrambe non nulle */
   if (s1.charAt(0) == s2.charAt(0))
     res = maggiore(s1.substring(1),s2.substring(1));
   else
     res = s1.charAt(0) > s2.charAt(0);
 }
 return res;
}
```

Alternativa: il metodo delle api di Java

- **public int compareTo(String anotherString)**

  Compares two strings lexicographically. The comparison is based on the Unicode value of each character in the strings. The character sequence represented by this String object is compared lexicographically to the character sequence represented by the argument string.

- **The result is:**

  a negative integer if this String object lexicographically precedes the argument string.

  a positive integer if this String object lexicographically follows the argument string.

  zero if the strings are equal;

  - s1.compareTo(s2) > 0 => s1>s2
  - s1.compareTo(s2) < 0 => s1<s2
  - s1.compareTo(s2) = 0 => s1=s2

Ordinamento lessicografico: bubbleSort

```
public static void bubbleSort(String[] a) {
   boolean finito = false, fattoscambio;
   int i = 0;              /* fatte 0 passate */
   int n = a.length;
   while (!finito) {
       i=i+1;
       fattoscambio = false;
       for (int j=0; j < n-i; j++)
       /* si modifica il criterio per il confronto
          tra due elementi a[j] > a[j+1] */
         if (maggiore(a[j],a[j+1])) {
           scambia(a, j, j+1);
           fattoscambio = true;
         }
       if ((!fattoscambio) || (i==n-1))
          finito = true;
   }
}
```

Il metodo main e il metodo di test

```
public static void main(String[] args){
   testStringhe();
}


public static void testStringhe(){
   String[] r;
   String s;

   r = crea();
   stampa(r);
   bubbleSort(r);
   System.out.println("*** Nomi ordinati ***");
   stampa(r);
}
```

- Crea() puo' essere definito in qualunque modo

Il metodo crea()

```
public static String[] crea(){
   String[] res;

   res = new String[]{"Carlo", "Maria", "Giovanni",
                      "Paola", "Alessandro",
                      "Teresa", "Alessio",
                      "Edoardo", "Mario"};
   return res;
}
```

Esempio d'esecuzione

```
Carlo
Maria
Giovanni
Paola
Alessandro
Teresa
Alessio
Edoardo
Mario
*** Nomi ordinati  ***
Alessandro
Alessio
Carlo
Edoardo
Giovanni
Maria
Mario
Paola
Teresa
Press any key to continue . . .
```

Esercizio

- **Scrivere un metodo che, ricevendo come parametro un array di oggetti Rettangolo, ordina l'array in modo non decrescente**
  - **rispetto alla base dei rettangoli**
  - **a parità di base, i rettangoli vanno ordinati per altezza non decrescente**

La classe rettangolo…

- Variabili d'istanza e costruttore

```
class Rettangolo{

   private int base;
   private int altezza;

   public Rettangolo(int b, int h) {
       base = b;
       altezza = h;
   }
```

…La classe rettangolo

- I metodi d'istanza e il metodo toString()

```
    public int getBase(){
        return base;
    }

    public int getAltezza(){
        return altezza;
    }


    public int area() {
        return base*altezza;
    }

    public int perimetro() {
        return 2*base + 2*altezza;
    }

    public String toString(){
        return "base:    " + base + "\n"+
               "altezza: " + altezza + "\n"+
               "area:    " + base*altezza + "\n";
    }
}
```

Creare un array di rettangoli casuali

- crea un array di n rettangoli generati casualmente con dimensioni tra 0 e 100

```
public static Rettangolo[] crea(int n){
    Rettangolo[] res;
    int i;
    double b,h;

    res=new Rettangolo[n];
    for (i=0; i<n; i++)
          res[i]=new Rettangolo(
                        (int)(Math.random()*100),
                        (int)(Math.random()*100)
                        );
    return res;
}
```

Il metodo di test

```
public static void testRettangoli(){
    Rettangolo[] r,p;
    Rettangolo res;

    r = crea(5);
    stampa(r);
    ordinaBase(r);
    System.out.println(" Rettangoli ordinati
                        rispetto alla base ");
    stampa(r);
}
```

Ordinamento dei rettangoli

- Rispetto alla base
- Usando il selectionSort ricorsivo

```
public static void ordinaBase(Rettangolo[] r){
    selectionSortRic(r,0,r.length);
}
```

- Il metodo definito precedentemente non puo' essere usato perchè è definito per gli interi e non per gli oggetti di tipo Rettangolo
- Bisogna modificare il selection sort

Selection sort per i rettangoli

```
public static void selectionSortRic
                    (Rettangolo[] v, int i, int n) {
    if (i < n-1)
      selectionSortRic(scambia(v,i,min(v,i,n-1)),
                       i+1, n);
}
```

- I metodi scambia e min definiti precedentemente non possono essere usati perchè sono definiti per interi e non per oggetti di tipo Rettangolo

Il metodo scambia

```
private static Rettangolo[] scambia(Rettangolo[] dati,
   int i, int j) {

  Rettangolo temp;  /*var di supporto per lo scambio*/

  /* scambia dati[i] con dati[j] */
  temp = dati[i];
  dati[i] = dati[j];
  dati[j] = temp;

  return dati;
}
```

Il metodo min

```
private static int min(Rettangolo[] a, int inf, int sup){
 int i,indmin;

 indmin = inf;
 for (i = inf+1; i <= sup; i++)
   if (a[i].getBase() < a[indmin].getBase())
     indmin = i;
 return indmin;
}
```

Esempio d'esecuzione

```
base:    23
altezza: 51
area:    1173

base:    62
altezza: 23
area:    1426

base:    45
altezza: 72
area:    3240

base:    68
altezza: 82
area:    5576

base:    89
altezza: 36
area:    3204
```

```
Rettangoli ordinati rispetto alla base
base:    23
altezza: 51
area:    1173

base:    45
altezza: 72
area:    3240

base:    62
altezza: 23
area:    1426

base:    68
altezza: 82
area:    5576

base:    89
altezza: 36
area:    3204

Press any key to continue . . .
```

Esercizi

- Scrivere un metodo **void mischia(int[] a)** che modifica, in modo casuale, l'ordine degli elementi all'interno di **a**