

-Arrays-

Fundamentals to programming I

Yeimy Estephany Huanca Sancho ¹ Omid Ernesto Chahuaris Choque ²

¹System Engineering School
System Engineering and Informatic Department
Production and Services Faculty
San Agustín National University of Arequipa

2020-04-08



UNSA

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

Content

- 1 Arrays
- 2 Character string
- 3 Sorting algorithms
- 4 Search algorithms
- 5 Requirements
- 6 Code
- 7 References

Arrays-Definition

- Also called arrangements.
- Data structure containing a collection of data of the same type.

Arrays-Properties

- They are used as containers to store related data.
- All the data included in the array is of the same type.
- The size of the Array is established when it is created.
- To access the Array data, the position it occupies within the set must be used.

Arrays-Declaration

- Square brackets are used to indicate that it is an Array and not a simple variable.

one-dimensional array

```
type name[ ];
```

or

```
type [ ] name;
```

two-dimensional array

```
type name[ ] [ ];
```

or

```
type [ ] [ ] name;
```

Arrays-Creation

- They are created with the "new" operator.

one-dimensional

```
int [ ] ages = new int [ 15 ];
```

two-dimensional

```
int [ ] [ ] temperatures = new int [10] [16];
```

Arrays-Use

- Indexes are used to access the elements of an array.

one-dimensional

```
float[] notas = new float[3];
```

```
notas[0] = 2
```

```
notas[1] = 5
```

```
notas[2] = 6
```

two-dimensional

```
int [][] matrix = new int[4][5];
```

```
matrix[0][0] = 15;  
matrix[0][1] = 21;  
matrix[0][2] = 18;  
matrix[0][3] = 9;  
matrix[0][4] = 15;
```

```
matrix[1][0] = 10;  
matrix[1][1] = 52;  
matrix[1][2] = 17;  
matrix[1][3] = 19;  
matrix[1][4] = 7;
```

```
matrix[2][0] = 19;  
matrix[2][1] = 2;  
matrix[2][2] = 19;  
matrix[2][3] = 17;  
matrix[2][4] = 7;
```

```
matrix[3][0] = 92;  
matrix[3][1] = 13;  
matrix[3][2] = 13;  
matrix[3][3] = 32;  
matrix[3][4] = 41;
```

Arrays-Initialization in the declaration

- You can assign an initial value to the elements of an Array in the declaration itself.

```
int vector[] = {1, 2, 3, 5, 7};  
int matriz[][] = { {1,2,3}, {4,5,6} };
```


Arrays-Manipulation Examples

- Operations are done component by component.

one-dimensional

```
static float media (float datos[])
{
    int    i;
    int    n = datos.length;
    float suma = 0;

    for (i=0; i<n; i++)
        suma = suma + datos[i];

    return suma/n;
}
```

two-dimensional

```
public static void mostrarMatriz (double matriz[][])
{
    int i,j;
    int filas = matriz.length;
    int columnas = matriz[0].length;

    // Recorrido de las filas de la matriz
    for (i=0; i<filas; i++) {

        // Recorrido de las celdas de una fila
        for (j=0; j<columnas; j++) {

            System.out.println ( "matriz["+i+"]["+j+"]="
                                + matriz[i][j] );

        }
    }
}
```

Character string

- The `java.lang.String` class is used to represent character strings in Java. It includes different methods that help us with character string operations.

Character string-substring

- The "substring" method allows us to get a substring.

```
String java="Java";  
String s = java.substring(0,3);  
System.out.println(s);           // Jav
```

Character string-charAt

- The "charAt (n)" method returns the character that is in position n of the string.

```
String java="Java";  
char c = java.charAt(2);  
System.out.println(c);           // v
```

Character string-indexOf

- The "indexOf(s)" method returns the position of a substring within a string.

```
String java="Java";  
int p = java.indexOf("av");  
System.out.println(p);           // 1
```

Character string-replace

- The "replace (old, new)" method replaces substrings.

```
String java="Java";  
java.replace("ava","ini");  
System.out.println(java);           // Jini
```

Character string-equals

- The "equals (s)" method is used to whether two strings are equals.

```
if (s.equals("Hola")) {  
    ...  
}
```

Character string-startsWith

- The method "starts with (s)" tells us if a string starts with a certain prefix.

```
if (s.startsWith("get")) {  
    ...  
}
```


Character string-endsWith

- The "endsWith (s)" method tells us if a string ends with a certain suffix.

```
if (s.endsWith(".html")) {  
    ...  
}
```

Character string-length

- The "length" method tells us the length of the string

```
String saludo = "Hola";
```

```
int longitud = saludo.length();
```

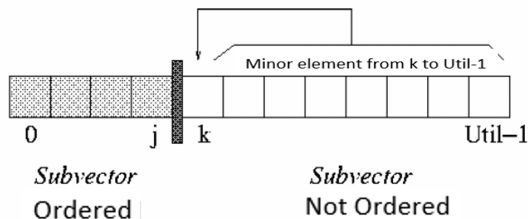
Sorting algorithms

- sort by selection.
- insertion sort.
- Direct exchange ordering (Bubblesort).
- QuickSort

Sorting algorithms

- Selection sort

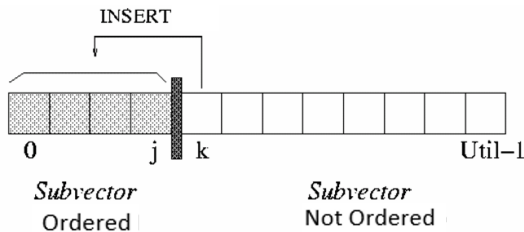
- Procedure: In each iteration, the smallest element of the unordered subset or subvector is selected and exchanged with the first element of this.
- Greater efficiency
- Complexity: n squared
- Method: selection



Sorting algorithms

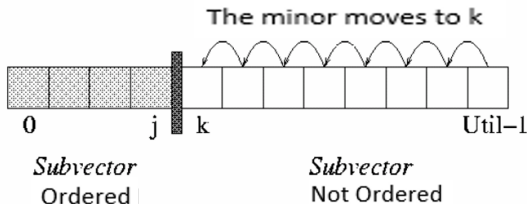
- Insertion sort

- Procedure: In each iteration, an element from the unordered subvector is inserted at the correct position within the ordered subvector.
- Greater efficiency
- Complexity: $n^2/2$
- Method: insert
- Example: a deck of cards



Sorting algorithms

- Direct exchange ordering (Bubblesort)
 - Lower efficiency
 - Complexity: n squared
 - Method: exchange
 - Is more simple



Sorting algorithms

- QuickSort

- Procedure: Reposition the other elements of the list on each side of the pivot, so that on one side are all the lesser than him, and on the other the greater
- Greater efficiency
- Complexity: $n \log n$ or n squared in the worst case
- Method: partition

Sorting algorithms

- Linear search or Sequential search
 - Process :It is compared sequentially, one by one.

```

1
2 public class BusquedaLineal {
3     //Linear search for an element in a vector
4     // - Returns the position of "data" in the vector
5     // - If "data" is not in the vector, returns -1
6     public static void main (String[]args) {
7         double []a={9.0,5.0,3.0,4.0,8.0,2.0};
8         Search(a,4.0);
9     }
10
11     static void Search (double vector[], double dato)
12     {
13         int i;
14         int N = vector.length;
15         int pos = -1;
16
17         for (i=0; (i<N) && (pos==-1); i++)
18
19             if (vector[i]==dato)
20                 pos = i;
21             System.out.println(pos);
22
23     }
24 }
25

```


Sorting algorithms

- Binary search
 - Pre-condition: Must be ordered
 - Process :
 - The searched data is compared with the element in the center of the vector
 - If they match, we have found the desired data.
 - If the data is greater than the central element of the vector, we have to find the data in the second half of the vector.
 - If the data is less than the central element of the vector, we have to find the data in the first half of the vector.

Sorting algorithms

- Binary search

```
// recursive implementation
// Usage: binSearch (vector, 0, vector.length-1, data)
static int binSearch
(double v[], int izq, int der, double buscado){
    int centro = (izq+der)/2;
    if (izq>der)
        return -1;
    else if (buscado==v[centro])
        return centro;
    else if (buscado<v[centro])
        return binSearch(v, izq, centro-1, buscado);
    else
        return binSearch(v, centro+1, der, buscado);
}

// Iterative implementation
// Use: binSearch (vector, data)
static int binSearch (double v[], double buscado){
    int izq = 0;
    int der = v.length-1;
    int centro = (izq+der)/2;

    while ((izq<=der) && (v[centro]!=buscado)) {

        if (buscado<v[centro])
            der = centro - 1;
        else
            izq = centro + 1;
        centro = (izq+der)/2;
    }
}
```

Requirements

- JDK ≥ 1.8
- MS Windows, MacOS, GNU/Linux: Oracle
 - jdk-8u261-windows-i586.exe
 - jdk-8u261-windows-x64.exe
 - jdk-8u261-macosx-x64.dmg
 - jdk-8u261-linux-x64.tar.gz (Optional)
- GNU/Linux: Ubuntu

Code

- Selection sort

```
1
2 public class insercion {
3     public static void main (String[] args) {
4         int []a={3,8,6,4};
5         OrdInsercion(a);
6         for (int i = 1; i <= a.length; i++) {
7             System.out.println(a[i-1]+" ");
8         }
9     }
10
11     private static void OrdInsercion(int[] A) {
12         int aux, i;
13         for (int j = 1; j < A.length; j++) {
14             aux = A[j];
15             i = j-1;
16             while (i>-1 && A[i]>aux) {
17                 A[i+1]= A[i];
18                 i = i-1;
19             }
20             A[i+1] = aux ;
21         }
22     }
23     // TODO Auto-generated method stub
24 }
```

Code

- Selection sort

```

2 public class Seleccion {
3
4 public static void main(String[] args){
5     int b[]={5,9,3,4,55,26,74};
6     OrdSeleccion(b);
7     for (int i = 1; i <= b.length; i++) {
8         System.out.println(b[i-1]+" ");
9     }
10
11 }
12 private static void OrdSeleccion(int[] a) {
13     // TODO Auto-generated method stub
14     for (int i = 0; i < a.length - 1; i++)
15     {
16         int min = i;
17         for (int j = i + 1; j < a.length; j++)
18         {
19             if (a[j] < a[min])
20             {
21                 min = j;
22             }
23         }
24         if (i != min)
25         {
26             int aux= a[i];
27             a[i] = a[min];
28             a[min] = aux;
29         }
30     }
31 }

```

Code

- Insertion sort

```
1
2 public class insercion {
3     public static void main (String[] args) {
4         int []a={3,8,6,4};
5         OrdInsercion(a);
6         for (int i = 1; i <= a.length; i++) {
7             System.out.println(a[i-1]+" ");
8         }
9     }
10
11     private static void OrdInsercion(int[] A) {
12         int aux, i;
13         for (int j = 1; j < A.length; j++) {
14             aux = A[j];
15             i = j-1;
16             while (i>-1 && A[i]>aux) {
17                 A[i+1]= A[i];
18                 i = i-1;
19             }
20             A[i+1] = aux ;
21         }
22         // TODO Auto-generated method stub
23
24
```

- Direct exchange ordering (Bubblesort)

```
1
2 public class burbuja {
3     public static void main (String []args) {
4         int []n = {4,10,2,9};
5         int []o = new int [4];
6         o=OrdBurbuja(n);
7         System.out.println(o[0]+","+o[1]+","+o[2]+","+o[3]);
8     }
9
10    public static int [] OrdBurbuja(int[]n) {
11        int aux;
12        int t = n.length;
13        for (int i = 1; i < t; i++) {
14            for (int k = t-1; k>= i;k--) {
15                if(n[k] < n[k-1]) {
16                    aux = n[k];
17                    n[k] = n[k-1];
18                    n[k-1]= aux ;
19                }
20            }
21        }
22        return n ;
23    }
24 }
25 }
```

Code

- QuickSort

```
11- static void QS(int[] vector, int primero, int ultimo){
12     int i=primero, j=ultimo;
13     int pivote=vector[(primero + ultimo) / 2];
14     //divide y venceras , ve por partes
15     int aux;
16
17     do
18     {
19         while(vector[i]<pivote) i++;
20         while(vector[j]>pivote) j--;
21
22         if (i<=j)
23         {
24             aux=vector[j];
25             vector[j]=vector[i];
26             vector[i]=aux;
27             i++;
28             j--;
29         }
30     }
31     while (i<=j);
32
33     if(primero<j)
34     {
35         QS(vector,primero, j);
36     }
37
38     if(ultimo>i)
39     {
40         QS(vector,i, ultimo);
41     }
```


References - Web pages

- <https://elvex.ugr.es/decsai/java/>
- https://es.wikipedia.org/wiki/Algoritmo_de_ordenamiento
- <https://es.wikipedia.org/wiki/Quicksort>
- <https://gl-eqn-programacion-ii.blogspot.com/2010/06/metodos-de-ordenamiento.html>
- <https://sites.google.com/a/espe.edu.ec/programacion-ii/home/a1-arreglos/algoritmos-de-ordenacion-y-busqueda>
- <https://slideplayer.es/slide/1383043/>
- https://www.youtube.com/watch?v=__tUncS0AsNE

References - Books

- Java programming basics of Marco Aedo

Thanks for your attention!...