Estructuras de datos
Clase práctica 3

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MEDELLÍN

Contenido

- Ordenamiento y búsqueda en arreglos

Material elaborado por: Julián Moreno

Facultad de Minas, Departamento de Ciencias de la Computación y la Decisión

Ordenamiento de arreglos estáticos

Como vimos en clase teórica, el ordenamiento de un arreglo se puede hacer en *O(n.log(n))* pero, ¿será que toca programar el *mergeSort*, o podemos usar algún “atajo”?

Afortunadamente para nosotros, Java nos facilita este trabajo mediante el método *Arrays.sort()* que se encuentra en la librería *java.util.**

La sintaxis es sumamente simple:

```
Arrays.sort(x); //Ordena de forma ascendente el arreglo x
```

Con la ventaja adicional que *x* puede ser de cualquier tipo (int, long, float, etc.).

Veamos un ejemplo …

Ordenamiento de arreglos estáticos

```
import java.util.*;

public class Main{

    public static void main(String[] args){
        int i;
        int a[] = {3, 1, 4, 5, 2};
        System.out.println("Arreglo desordenado");
        for (i=0; i<a.length; i++)
            System.out.println(a[i]);
        System.out.println("Arreglo ordenado");
        Arrays.sort(a);
        for (i=0; i<a.length; i++)
            System.out.println(a[i]);
    }

}
```

Búsqueda binaria en arreglos estáticos

Como vimos en clase teórica, una búsqueda lineal dentro de un arreglo se puede hacer en *O(n),* mientras que una búsqueda binaria (si el arreglo está ordenado), puede hacerse en *O(log(n))*, ¿será que Java vuelve y nos facilita la vida?

La respuesta es sí, mediante el método *Arrays.binarySearch()* que adivinemos … se encuentra en la librería *java.util.**

La sintaxis es sumamente simple:

```
Arrays.binarySearch(x, e);
//Busca el elemento e dentro del arreglo x y devuelve su posición, o un número negativo si no se encuentra
```

Nuevamente *x* puede ser de cualquier tipo, eso si, del mismo que *e*. Veamos un ejemplo …

Búsqueda binaria en arreglos estáticos

```
import java.util.*;

public class Main{

    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);
        int i, b, p;
        int a[] = {3, 1, 4, 5, 2};
        System.out.println("Arreglo desordenado");
        for (i=0; i<a.length; i++)
            System.out.println(a[i]);
        System.out.print("Valor a buscar: ");
        b = entrada.nextInt();
        Arrays.sort(a);
        p = Arrays.binarySearch(a, b);
        if (p >= 0)
            System.out.println("Se encuentra!");
        else
            System.out.println("No se encuentra!");
    }

}
```

Arreglos dinámicos

Como vimos anteriormente *indexOf(e)* devuelve, mediante búsqueda lineal, la posición de la primera ocurrencia del elemento *e* en el arreglo, o -1 si no se encuentra.

Pero ¿Podemos usar *sort()* y *binarySearch()* en el caso de arreglos dinámicos?

La respuesta es que si, usando *Collections.sort(x)* y *Collections.binarySearch(x)* respectivamente

Resumen

Operación	Arreglo estático	Arreglo dinámico
Declaración	`int a[ ] = new int[N]`	`Vector <Integer> a = new Vector<Integer>()` *
Indexado	`a[i]`	`a.get(i)`
Asignación / reemplazo	`a[i] = e`	`a.set(i, e)`
Búsqueda lineal	`Hay que programarlo`	`a.indexOf(e)` `a.lastIndexOf(e)`
Búsqueda binaria	`Arrays.binarySearch(x,e)`	`Collections.binarySearch(x,e)`
Ordenamiento	`Arrays.sort(x)`	`Collections.sort(x)`

* Recuerden que también puede ser Byte, Short, Long, Float, Double (comenzando con mayúscula)