Estructuras de datos
Clase práctica 2

UNIVERSIDAD NACIONAL DE COLOMBIA SEDE MEDELLÍN

Contenido

- Ordenamiento de arreglos
- Búsqueda binaria
- Arreglos dinámicos

Material elaborado por: Julián Moreno

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

Ordenamiento de arreglos

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

```
import java.util.*;

public class Main{

    public static void main(String[] args){
        int i, a[] = new int[10];
        System.out.println("Arreglo desordenado");
        for (i=0; i<10; i++){
            a[i] = (int) Math.round(Math.random()*100);
            System.out.println(a[i]);
        }
        System.out.println("Arreglo ordenado");
        Arrays.sort(a);
        for (i=0; i<10; i++){
            System.out.println(a[i]);
        }
    }

}
```

Búsqueda en arreglos

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 en arreglos

```
import java.util.*;

public class Main{

    public static void main(String[] args){
        int i, b, p, a[] = new int[10];
        Scanner entrada = new Scanner(System.in);
        System.out.println("Arreglo");
        for (i=0; i<10; i++){
            a[i] = (int) Math.round(Math.random()*100);
            System.out.println(a[i]);
        }
        System.out.println("Valor a buscar [0,100]:");
        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

Ya vimos en clase teórica que la diferencia entre un arreglo "normal" y uno dinámico es que el segundo permite, luego de creado, agregar más elementos o eliminarlos. ¿Cómo podemos usarlos en Java?

Una forma de hacerlo es mediante la clase *Vector*, que ¿adivinen en que librería está? Entre los métodos que tiene, los siguientes serán de nuestro interés:

Arreglos dinámicos

`add(e)`	Ingresa el elemento *e* al final del arreglo
`add(i,e)`	Ingresa el elemento *e* en la posición *i* del arreglo
`clear()`	Borra todos los elementos
`get(i)`	Devuelve el elemento en la posición *i*
`isEmpty()`	Devuelve verdadero si el arreglo está vacío
`indexOf(e)`	Devuelve la posición de la primera ocurrencia del elemento *e* dentro del arreglo, o -1 si no está
`lastIndexOf(e)`	Devuelve la posición de la última ocurrencia del elemento *e* dentro del arreglo, o -1 si no está
`remove(i)`	Borra el elemento en la posición *i*
`set(i,e)`	Reemplaza el elemento en la posición *i* por *e*
`size()`	Devuelve la cantidad de elementos en el arreglo

Arreglos dinámicos

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

¿Significa esto que no podemos usar *Arrays.binarySearch()* en el caso de arreglos dinámicos?

La respuesta es que si lo podemos usar, el problema es que no podemos usar el *Array.sort()* pues este solo organiza arreglos estáticos. En otras palabras, podemos usar *Arrays.binarySearch()* pero si nosotros mismos garantizamos que el arreglo está ordenado ¿Cómo podemos hacer esto?

Fácil! Como la utilidad de un arreglo dinámico es precisamente agregar o eliminar elementos (de a uno), ¿Qué tal si siempre que agregamos un nuevo elemento lo "movemos" a la posición donde deba para que el arreglo quede ordenado?

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Vector numeros = new Vector();
        Scanner entrada = new Scanner(System.in);
        int i, aux, x, k;
        String opcion;

        do{
            System.out.println("Numero a ingresar en el arreglo : ");
            x = entrada.nextInt();
            numeros.add(x);
            for (i = numeros.size()-1; i > 0; i--){
                if ((int) numeros.get(i) < (int) numeros.get(i-1)){
                    aux = (int) numeros.get(i);
                    numeros.set(i, (int)numeros.get(i-1));
                    numeros.set(i-1, aux);
                }
                else
                    break;
            }
            System.out.println("Desea ingresar otro número? (s/n): ");
            opcion = entrada.next();
        } while (opcion.compareTo("s") == 0); //...
```

```
        //...
        System.out.println("Numeros ingresados");
        for (i = 0; i < numeros.size(); i++){
            System.out.println(numeros.get(i));
        }

        do{
            System.out.println("Numero a buscar: ");
            x = entrada.nextInt();
            k = Arrays.binarySearch(numeros.toArray(), x);
            if (k >= 0){
                System.out.println(x + " si esta en el arreglo");
            }
            else{
                System.out.println(x + " no esta en el arreglo");
            }
            System.out.println("Desea buscar otro número? (s/n): ");
            opcion = entrada.next();
        } while (opcion.compareTo("s") == 0);

    }

}
```

Ejercicios en CPP

Ahora si: ¿qué tal si arrancamos con el ejercicio "K-esimos elementos"?

Tareas

Realizar (analizar, diseñar, implementar y enviar) al menos 2 ejercicios del segundo módulo