Algoritmos de ordenación Arrays

Encarnación Sánchez Gallego

CONTENIDOS

- Introducción
- Algoritmos de ordenación:
  - Intercambio
  - Burbuja
  - Quicksort
  - Método sort

Introducción

Ordenar un array es muy importante, ya sea de números o de cadenas, puede haber casos en que nos interese que los datos esten ordenados en un array. Tenemos varias formas de hacerlo, tanto con números como con cadenas, los más importantes son:

- **Intercambio**
- **Burbuja**
- **Quicksort**
- **Método sort de la clase java.util.Arrays**

Intercambio

Consiste en comparar el primer valor con el resto de las posiciones posteriores, cambiando el valor de las posiciones en caso de que el segundo sea menor que el primero comparado, después la segunda posición con el resto de posiciones posteriores. Te enseñamos un ejemplo de como funciona.

- Método intercambio con números:

```
public static void intercambio(int lista[]){

        //Usamos un bucle anidado
        for(int i=0;i<(lista.length-1);i++){
            for(int j=i+1;j<lista.length;j++){
                if(lista[i]>lista[j]){
                    //Intercambiamos valores
                    int variableauxiliar=lista[i];
                    lista[i]=lista[j];
                    lista[j]=variableauxiliar;

                }
            }
        }
    }
```

- Método intercambio con cadenas:

```
public static void intercambioPalabras(String lista[]){

                    //Usamos un bucle anidado
                for(int i=0;i<(lista.length-1);i++){
                    for(int j=i+1;j<lista.length;j++){
                        if(lista[i].compareToIgnoreCase(lista[j])>0){
                            //Intercambiamos valores
                            String variableauxiliar=lista[i];
                            lista[i]=lista[j];
                            lista[j]=variableauxiliar;

                        }
                    }
                }
    }
```

Burbuja

Consiste en comparar el primero con el segundo, si el segundo es menor que el primero se intercambian los valores. Después el segundo con el tercero y así sucesivamente, cuando no haya ningún intercambio, el array estará ordenado. Lo peor de este método de ordenación, es que tiene una complejidad de **O(n2)** haciendo que cuanto más valores a ordenar mayor tiempo tardará en ordenar.

- Método burbuja para números:

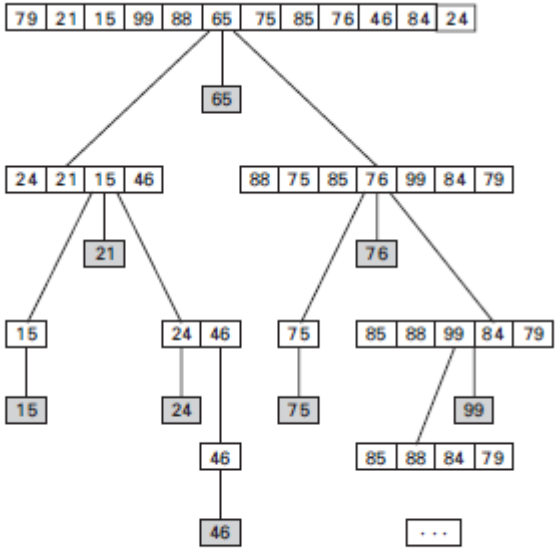
```
public static void burbuja (int lista[]){
        int cuentaintercambios=0;
        //Usamos un bucle anidado, saldra cuando este ordenado el array
        for (boolean ordenado=false;!ordenado;){
            for (int i=0;i<lista.length-1;i++){
                if (lista[i]>lista[i+1]){
                    //Intercambiamos valores
                    int variableauxiliar=lista[i];
                    lista[i]=lista[i+1];
                    lista[i+1]=variableauxiliar;
                    //indicamos que hay un cambio
                    cuentaintercambios++;
                }
            }
            //Si no hay intercambios, es que esta ordenado.
            if (cuentaintercambios==0){
                ordenado=true;
            }
            //Inicializamos la variable de nuevo para que empiece a contar de nuevo
            cuentaintercambios=0;
        }
    }
```

- Método burbuja para cadena:

```
public static void burbujaPalabras (String lista_palabras[]){
    boolean ordenado=false;
    int cuentaIntercambios=0;
    //Usamos un bucle anidado, saldra cuando este ordenado el array
    while(!ordenado){
        for(int i=0;i<lista_palabras.length-1;i++){
            if (lista_palabras[i].compareToIgnoreCase(lista_palabras[i+1])>0){
                //Intercambiamos valores
                String aux=lista_palabras[i];
                lista_palabras[i]=lista_palabras[i+1];
                lista_palabras[i+1]=aux;
                //indicamos que hay un cambio
                cuentaIntercambios++;
            }
        }
        //Si no hay intercambios, es que esta ordenado.
        if (cuentaIntercambios==0){
            ordenado=true;
        }
        //Inicializamos la variable de nuevo para que empiece a contar de nuevo
        cuentaIntercambios=0;
    }
}
```

Quicksort

Consiste en ordenar un array mediante un pivote, que es un punto intermedio en el array, es como si se ordenaran pequeños trozos del array, haciendo que a la izquierda esten los menores a ese pivote y en la derecha lo mayores a este, después se vuelve a calcular el pivote de trozos de listas. Usa recursividad. Le pasamos el array, su posición inicial y su posición final como parámetro. Tiene una complejidad de **O(n log2 n),** haciendo que mejore el rendimiento aun teniendo muchos valores que ordenar.



Izquierda: 24, 21, 15, 46
Pivote: 65
Derecha: 88, 75, 85, 76, 99, 84, 79

- Quicksort con números:

```
public static void quicksort (int lista1[], int izq, int der){
    int i=izq;
    int j=der;
    int pivote=lista1[(i+j)/2];
    do {
        while (lista1[i]<pivote){
            i++;
        }
        while (lista1[j]>pivote){
            j--;
        }
        if (i<=j){
            int aux=lista1[i];
            lista1[i]=lista1[j];
            lista1[j]=aux;
            i++;
            j--;
        }
    }while(i<=j);
    if (izq<j){
        quicksort(lista1, izq, j);
    }
    if (i<der){
        quicksort(lista1, i, der);
    }
}
```

- Quicksort con cadenas:

```
public static void quicksortP (String lista1[], int izq, int der){
        int i=izq;
        int j=der;
        int pivote=(i+j)/2;
        do {
            while (lista1[i].compareToIgnoreCase(lista1[pivote])<0){
                i++;
            }
            while (lista1[j].compareToIgnoreCase(lista1[pivote])>0){
                j--;
            }
            if (i<=j){
                String aux=lista1[i];
                lista1[i]=lista1[j];
                lista1[j]=aux;
                i++;
                j--;
            }
        }while(i<=j);
        if (izq<j){
            quicksortP(lista1, izq, j);
        }
        if (i<der){
            quicksortP(lista1, i, der);
        }
    }
```

Método sort

Método **sort** de **java.util.Arrays**: para ejecutarlo escribimos **Arrays.sort(array a ordenar);** simplemente insertamos como parámetro el array que queremos ordenar. Tiene varios métodos para distintos tipos.