

UNIVERSIDAD NACIONAL DE INGENIERÍA



Introducción a la Programación

Método de ordenamiento: Mergesort

Nombre: Steven Alejandro Álvarez Avendaño

N.º Carnet: 2024-1675U

Docente: Ing. Cristopher Larios

Grupo: 1M7-S

Fecha: domingo 05 de mayo de 2024

Explique con sus propias palabras como puede ocupar este método de ordenamiento Mergesort.

Supongamos que tenemos una lista de nombres de personas junto con sus edades, y queremos ordenar esta lista alfabéticamente por el nombre de la persona. Sin embargo, si dos personas tienen el mismo nombre, entonces queremos ordenarlas por edad de manera ascendente. Entonces este sería un buen momento para usar Mergesort.

Para ordenar esta lista, se puede implementar Mergesort de tal manera que primero se comparen los nombres de las personas. Si los nombres son iguales, entonces compara las edades, el cual divide la lista en mitades, y recursivamente ordena cada mitad, luego combina las mitades ordenadas asegurándose que los elementos se ordenen correctamente. Todo este proceso dará una lista ordenada alfabéticamente por nombres y si hay nombres iguales, pues los ordenara por edad.

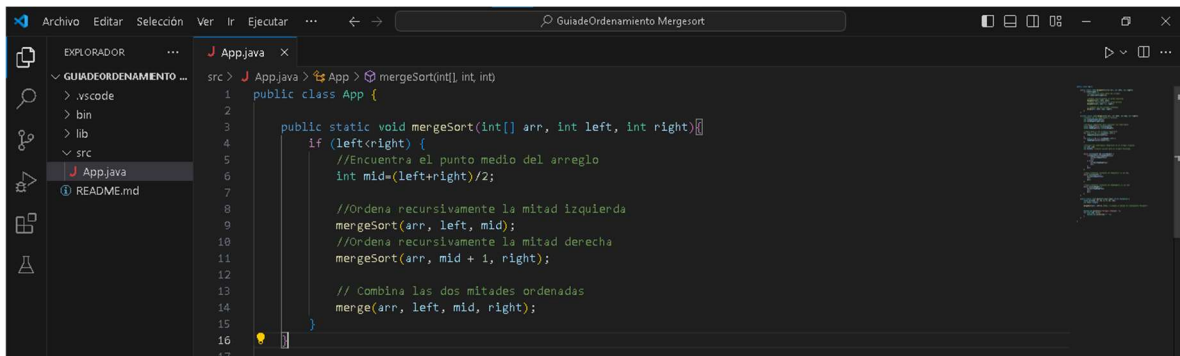
Elaboración de la Guía

Implementación en Java (Parte 1) - Método Mergesort

- MergeSort divide el arreglo en mitades recursivamente.



- Llamadas recursivas para ordenar cada mitad



Implementación en Java (Parte 2) - Método main

- El método Merge combina dos subarreglos ordenados (L y R) en un solo arreglo (arr).

```
private static void merge(int[] arr, int left, int mid, int right) {
    //Dividir de los subarreglos a fusionar
    int sizeOfLeft = left;
    int sizeOfRight = mid;

    //Arreglos temporales para almacenar los subarreglos
    int[] tempOfLeft = new int[sizeOfLeft];
    int[] tempOfRight = new int[sizeOfRight];

    //Copiar datos a los arreglos temporales
    for (int i = 0; i < sizeOfLeft; i++) {
        tempOfLeft[i] = arr[left + i];
    }
    for (int j = 0; j < sizeOfRight; j++) {
        tempOfRight[j] = arr[mid + 1 + j];
    }

    //Fusionar los subarreglos temporales en el arreglo original
    int low = left;
    int high = right;
    int k = left; //Indice inicial para el arreglo fusionado

    while (sizeOfLeft > 0 && sizeOfRight > 0) {
        if (tempOfLeft[0] < tempOfRight[0]) {
            arr[k] = tempOfLeft[0];
            sizeOfLeft--;
        } else {
            arr[k] = tempOfRight[0];
            sizeOfRight--;
        }
        k++;
    }

    //Copiar elementos restantes de tempOfLeft al arreglo
    while (sizeOfLeft > 0) {
        arr[k] = tempOfLeft[0];
        sizeOfLeft--;
        k++;
    }

    //Copiar elementos restantes de tempOfRight al arreglo
    while (sizeOfRight > 0) {
        arr[k] = tempOfRight[0];
        sizeOfRight--;
        k++;
    }
}
```

Implementación en Java (Parte 3) - Método main

- Se crea un arreglo desordenado.

```
public class App {
    private static void merge(int[] arr, int left, int mid, int right) {
        //...
    }

    public static void main(String[] args) throws Exception {
        int[] arr = {38, 27, 43, 3, 9, 82, 10};
        int n = arr.length;

        mergeSort(arr, left:0, n-1); //Llamada al metodo de ordenamiento MergeSort

        System.out.println("Arreglo ordenado: ");
        for (int num: arr) {
            System.out.print(num + " ");
        }
    }
}
```

- Llamamos al método MergeSort para ordenar el arreglo.
- Imprimimos el arreglo ordenado.

Ejecución del Programa

```
PS C:\Users\steve\Documents\Universidad\UNI\Programacion\Tareas\GuiadeOrdenamiento_Mergesort> 'C:\Program Files\Microsoft\jdk-11.0.16.1-hotspot\bin\java.exe' '-cp' 'C:\Users\steve\Documents\Universidad\UNI\Programacion\Tareas\GuiadeOrdenamiento_Mergesort\bin' 'App'
Arreglo ordenado:
3 9 10 27 38 43 82
PS C:\Users\steve\Documents\Universidad\UNI\Programacion\Tareas\GuiadeOrdenamiento_Mergesort>
```

Creación y Subida al Repositorio en GitHub

- Maquina local para guardar en el repositorio

```
MINGW64~/Users/steve/Documents/Universidad UNI/Programacion/Tareas/GuideOrdenamiento MergeSort
$ git init
Initialized empty Git repository in C:/Users/steve/Documents/Universidad UNI/Programacion/Tareas/GuideOrdenamiento MergeSort/.git/
$ git config --global user.email "stevenalv22@gmail.com"
$ git config --global user.name "Stevenalv22"
$ git remote add origin https://github.com/Stevenalv22/GuideOrdenamiento-MergeSort---2024-1675U.git
$ git add .
$ git commit
[master (root-commit) 3ea53f1] first commit
4 files changed, 104 insertions(+)
create mode 100644 vscode/settings.json
create mode 100644 README.md
create mode 100644 bin/App.class
create mode 100644 src/App.java
$ git branch -M main
$ git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 2.52 KiB | 746.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pushed-reused 0 (from 0)
To https://github.com/Stevenalv22/GuideOrdenamiento-MergeSort---2024-1675U.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
```

- Subida del proyecto en GitHub

