



5-5-2024

Programación

Universidad Nacional De Ingeniería



Ronald Oporta 2024 1623U
CHRISTOPHER LARIOS
IM7-S

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

El método de ordenamiento por fusión es útil cuando necesitas ordenar una lista de elementos de manera eficiente, mantener la estabilidad del orden, y cuando prefieres un algoritmo fácil de entender e implementar que funcione bien en una variedad de situaciones. el método de ordenamiento por fusión puede ser útil en situaciones cotidianas donde necesitas organizar y clasificar información de manera eficiente para facilitar tu trabajo o estudio.

Elaboración de la Guía

Implementación en Java (Parte 1) - Método Mergesot

- MergeSort divide el arreglo en mitades recursivamente.

```
src > J App.java > App
1 public class App {
2
3     public static void mergeSort(int[] arr, int left, int right){
4         if (left < right) {
5             //Encuentra el punto medio del arreglo
6             int mid = (left + right) / 2;
7         }
```

- Llamadas recursivas para ordenar cada mitad

```
src > J App.java > App
1 public class App {
2
3     public static void mergeSort(int[] arr, int left, int right){
4         if (left < right) {
5             //Encuentra el punto medio del arreglo
6             int mid = (left + right) / 2;
7
8             //Ordena recursivamente la mitad izquierda
9             mergeSort(arr, left, mid);
10            //Ordena recursivamente la mitad derecha
11            mergeSort(arr, mid + 1, right);
12
13            // Combina las dos mitades ordenadas
14            merge(arr, left, mid, right);
15        }
16    }
17 }
```

Implementación en Java (Parte 2) - Método main

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

```
18 private static void merge(int[] arr, int left, int mid, int right){
19     //Tamaños de los subarreglos a fusionar
20     int sizeLeft=mid-left+1;
21     int sizeRight=right-mid;
22
23     //Arreglos temporales para almacenar los subarreglos
24     int[] tempLeft=new int[sizeLeft];
25     int[] tempRight=new int[sizeRight];
26
27     //Copia datos a los arreglos temporales
28     for (int i = 0; i < sizeLeft; i++) {
29         tempLeft[i]=arr[left+i];
30     }
31     for (int j = 0; j < sizeRight; j++) {
32         tempRight[j]=arr[mid+1+j];
33     }
34
35     //Fusiona los subarreglos temporales en el arreglo original
36     int i=0, j=0;
37     int k=left; //Indice inicial para el arreglo fusionado
38
39
40     while (i<sizeLeft && j<sizeRight) {
41         if(tempLeft[i]<=tempRight[j]){
42             arr[k]=tempLeft[i];
43             i++;
44         } else{
45             arr[k]=tempRight[j];
46             j++;
47         }
```

```
51 //Copia elementos restantes de tempLeft[] si los hay
52 while (i<sizeLeft){
53     arr[k]=tempLeft[i];
54     i++;
55     k++;
56 }
57
58 //Copia elementos restantes de tempRight[] si los hay
59 while (j<sizeRight){
60     arr[k]=tempRight[j];
61     j++;
62     k++;
63 }
64 }
65
66 Run | Debug
67 public static void main(String[] args) throws Exception {
68     int[] arr={38, 27, 43, 3, 9, 82, 10};
69     int n=arr.length;
70     mergeSort(arr, left:0, n-1); //Llamada al metodo de ordenamiento MergeSort
71
72
73     System.out.println(x:"Arreglo ordenado: ");
74     for (int num: arr){
75         System.out.print(num + " ");
76     }
77 }
```

Implementación en Java (Parte 3) - Método main

- Se crea un arreglo desordenado.

```
Run | Debug
66 public static void main(String[] args) throws Exception {
67     int[] arr={38, 27, 43, 3, 9, 82, 10};
68     int n=arr.length;
69
70     mergeSort(arr, left:0, n-1); //Llamada al metodo de ordenamiento MergeSort
71
72
73     System.out.println(x:"Arreglo ordenado: ");
74     for (int num: arr){
75         System.out.print(num + " ");
76     }
77 }
```

- Llamamos al método mergeSort para ordenar el arreglo

```
Run | Debug
66 public static void main(String[] args) throws Exception {
67     int[] arr={38, 27, 43, 3, 9, 82, 10};
68     int n=arr.length;
69
70     mergeSort(arr, left:0, n-1); //Llamada al metodo de ordenamiento MergeSort
71
72
73     System.out.println(x:"Arreglo ordenado: ");
74     for (int num: arr){
75         System.out.print(num + " ");
76     }
77 }
```

- Imprimimos el arreglo ordenado.

```
Run | Debug
66 public static void main(String[] args) throws Exception {
67     int[] arr={38, 27, 43, 3, 9, 82, 10};
68     int n=arr.length;
69
70     mergeSort(arr, left:0, n-1); //Llamada al metodo de ordenamiento MergeSort
71
72
73     System.out.println(x:"Arreglo ordenado: ");
74     for (int num: arr){
75         System.out.print(num + " ");
76     }
77 }
```

Evidencia del proyecto en Github pasos

```
ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort
git init
initialized empty Git repository in C:/Users/ronal/Desktop/ GuiadeOrdenamiento Mergesort/.git/

ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git config --global user.email "ronald.oporta395@std.uni.edu.ni"

ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git config --global user.name "Ronald0239"

ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git remote add origin https://github.com/Ronald0239/GuiadeOrdenamiento-Mergesort.git


ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git add .





ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git commit
int: Waiting for your editor to close the file...
master (root-commit) e516faa] 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




ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git branch -M
fatal: branch name required



ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (master)
git branch -M main

ronald@LAPTOP-OPORTA MINGW64 ~/Desktop/ GuiadeOrdenamiento Mergesort (main)
git push -u origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 2 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 2.93 KiB | 499.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Ronald0239/GuiadeOrdenamiento-Mergesort.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'
```



 **GuiadeOrdenamiento-Mergesort** Public





  Unwatch 1  Fork 0  Star 0

 main  1 Branch  0 Tags


  Code


About


 **Ronald0239** first commit e516faa · 14 minutes ago  1 Commits


 .vscode	first commit	14 minutes ago
 bin	first commit	14 minutes ago
 src	first commit	14 minutes ago
 README.md	first commit	14 minutes ago


Este trabajo es de la Univercidad Nacional De Ingenieria- Programacion mi nombre es Ronald Ramon Oporta Sequeira

 Readme

 Activity

 0 stars

 1 watching

 0 forks