**MERGESORT – Proiect**

* Cerinte / tema proiectului

Mergesort este un algoritm de sortare care funcționează prin împărțirea unei matrice în subdiviziuni mai mici, sortând fiecare subbary și apoi îmbinând subgrupurile sortate înapoi împreună pentru a forma matricea sortată finală.

În termeni simpli, putem spune că procesul de sortare prin îmbinare este de a împărți matricea în două jumătăți, de a sorta fiecare jumătate și apoi de a îmbina din nou jumătățile sortate. Acest proces se repetă până când întreaga matrice este sortată.

Limbajul de programare in care a fost realizat codul este C++.

* Functionalitatea algoritmului

Functionalitatea algoritmului poate fi demonstrata graphic, pentru a putea fi inteles cu usurinta pe link-ul urmator:

<https://opendsa-server.cs.vt.edu/embed/mergesortAV>

* Informatii despre masina pe care a rulat codul
* **Specificatii ale dispozitivului:**
* **Procesor:** Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz
* **RAM:** 8,00 GB
* **Tip sistem:** Sistem de operare pe 64 de biți, procesor de tip x64
* **Specificatii Windows:**
* **Editie:** Windows 10 Pro
* **Versiune:** 22H2
* **Versiune SO:** 19045.2604
* **Experienta:** Windows Feature Experience Pack 120.2212.4190.0
* **Specificatii diverse:**
* **Tool:** Microsoft Visual Studio 2019
* Rezultate experimentale

Complexitatea temporală a sortării prin îmbinare este θ(Nlog(N)) în toate cele 3 cazuri (cel mai rău, mediu și cel mai bun), deoarece sortarea prin îmbinare împarte întotdeauna matricea în două jumătăți și necesită timp liniar pentru a îmbina două jumătăți.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test** | **Numarul de elemente** | **Timpul sortarii sirului secvential**  **(microsecunde)** | **Timpul sortarii sirului paralel MPI**  **(microsecunde)** | **Intervalul numerelor din sir** |
| Test1.txt | 10 | 24 | - | 0 - 99 |
| Test2.txt | 50 | 34 | - | 0 - 99 |
| Test3.txt | 70 | 47 | - | 0 - 99 |
| Test4.txt | 100 | 64 | - | 0 – 99 |
| Test5.txt | 10 | 10 | 23 | 0 - 99999999 |
| Test6.txt | 50 | 34 | 78 | 0 - 99999999 |
| Test7.txt | 100 | 65 | 143 | 0 - 99999999 |
| Test8.txt | 200 | 128 | 255 | 0 - 99999999 |
| Test9.txt | 500 | 333 | 687 | 0 - 99999999 |
| Test10.txt | 700 | 464 | 881 | 0 - 99999999 |
| Test11.txt | 1000 | 689 | 1210 | 0 - 99999999 |
| Test12.txt | 5000 | 3393 | 6311 | 0 - 99999999 |
| Test13.txt | 10000 | 6712 | 14653 | 0 - 99999999 |
| Test14.txt | 50000 | 35592 | 69946 | 0 - 99999999 |
| Test15.txt | 100000 | 70959 | 162545 | 0 - 99999999 |
| Test16.txt | 500000 | 372243 | 701062 | 0 - 99999999 |
| Test17.txt | 1000000 | 770403 | 1381161 | 0 - 99999999 |
| Test18.txt | 5000000 | 3619932 | 7244061 | 0 - 99999999 |
| Test19.txt | 10000000 | 7204779 | 13502538 | 0 - 99999999 |
| Test20.txt | 100000000 | 76514387 | 123738371 | 0 - 99999999 |

