**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++.

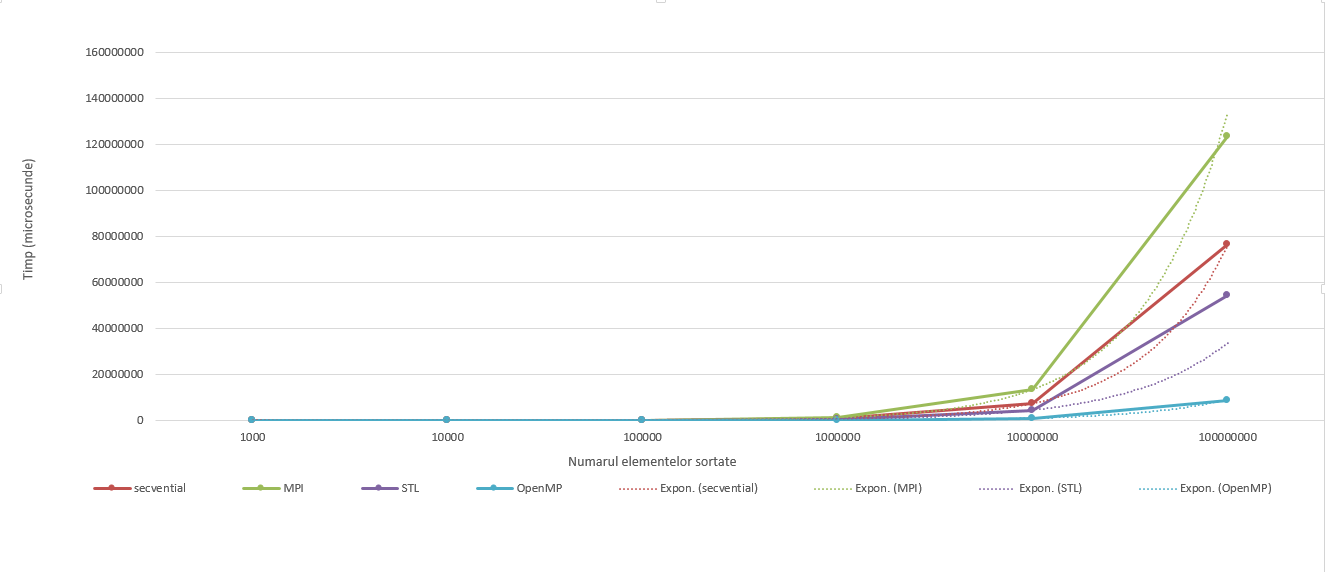
* Functionalitatea algoritmului

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)** | **Timpul sortarii sirului paralel STL**  **(microsecunde)** | **Timpul sortarii sirului paralel OpenMP**  **(microsecunde)** | **Intervalul numerelor din sir** |
| --- | --- | --- | --- | --- | --- | --- |
| Test1.txt | 10 | 24 | - | 23 | 1,5 | 0 - 99 |
| Test2.txt | 50 | 34 | - | 526 | 6,4 | 0 - 99 |
| Test3.txt | 70 | 47 | - | 643 | 10,7 | 0 - 99 |
| Test4.txt | 100 | 64 | - | 489 | 12,2 | 0 – 99 |
| Test5.txt | 10 | 10 | 23 | 30 | 1,7 | 0 - 99999999 |
| Test6.txt | 50 | 34 | 78 | 511 | 6,3 | 0 - 99999999 |
| Test7.txt | 100 | 65 | 143 | 480 | 12,7 | 0 - 99999999 |
| Test8.txt | 200 | 128 | 255 | 728 | 25,1 | 0 - 99999999 |
| Test9.txt | 500 | 333 | 687 | 1223 | 65,4 | 0 - 99999999 |
| Test10.txt | 700 | 464 | 881 | 1790 | 95,8 | 0 - 99999999 |
| Test11.txt | 1000 | 689 | 1210 | 2659 | 132,6 | 0 - 99999999 |
| Test12.txt | 5000 | 3393 | 6311 | 7270 | 917,8 | 0 - 99999999 |
| Test13.txt | 10000 | 6712 | 14653 | 11425 | 1659,2 | 0 - 99999999 |
| Test14.txt | 50000 | 35592 | 69946 | 24302 | 8138 | 0 - 99999999 |
| Test15.txt | 100000 | 70959 | 162545 | 45862 | 18734,7 | 0 - 99999999 |
| Test16.txt | 500000 | 372243 | 701062 | 214592 | 56202,5 | 0 - 99999999 |
| Test17.txt | 1000000 | 770403 | 1381161 | 493755 | 112404,3 | 0 - 99999999 |
| Test18.txt | 5000000 | 3619932 | 7244061 | 2376035 | 562020,9 | 0 - 99999999 |
| Test19.txt | 10000000 | 7204779 | 13502538 | 4195331 | 955435,53 | 0 - 99999999 |
| Test20.txt | 100000000 | 76514387 | 123738371 | 54430052 | 8598919,77 | 0 - 99999999 |

* Referinte ChatGPT

Intrebare:

Am un proiect despre mergesort de realizat

Raspuns:

Cu siguranță te pot ajuta cu proiectul tău despre Mergesort. Mergesort este un algoritm de sortare eficient, bazat pe principiul divide et impera. Așadar, în ce fel anume îți pot fi de folos? Ai nevoie de explicații teoretice, de ajutor cu codul sursă sau de ambele?

Intrebare:

Vreau sa imi explici daca este posibil sa il paralelizez

Raspuns:

Da, algoritmul Mergesort poate fi paralelizat pentru a beneficia de performanțe îmbunătățite în sistemele cu mai multe nuclee de procesare sau în medii distribuite. În esență, paralelizarea constă în împărțirea listei de sortat în subliste mai mici, care sunt apoi sortate independent folosind Mergesort, iar rezultatele parțiale sunt combinate pentru a obține lista finală sortată.

Iată o descriere succintă a unui posibil algoritm de paralelizare a Mergesort:

1. Se împarte lista inițială în mod recursiv în subliste mai mici până când fiecare sublistă conține un singur element.

2. Se lansează procese (sau thread-uri) separate pentru a sorta fiecare sublistă în paralel. Fiecare proces va utiliza Mergesort pentru a sorta sublista sa.

3. Se așteaptă ca toate procesele să se termine și să furnizeze listele lor parțial sortate.

4. Se combină listele parțial sortate într-o singură listă folosind operația de interclasare (merge) a algoritmului Mergesort.

5. Rezultatul final este lista sortată complet.

Intrebare:

Ce pare ai de a folosi OpenMP, Mpi si STL pentru a paraleliza algoritmul?

Raspuns:

Folosirea OpenMP, MPI și STL (Standard Template Library) este o abordare excelentă pentru a paraleliza algoritmul Mergesort. Aceste tehnologii oferă suport pentru paralelizarea task-urilor și manipularea listelor într-un mod eficient și convenabil. Iată cum poți utiliza fiecare dintre acestea pentru a paraleliza Mergesort:

1. OpenMP: OpenMP este o bibliotecă care permite programarea paralelă pe arhitecturi cu memorie partajată, cum ar fi procesoarele cu mai multe nuclee. Pentru a utiliza OpenMP în implementarea ta a Mergesort, urmează acești pași:

- Identifică secțiunile critice ale codului în care pot fi executate în paralel operațiile de sortare și interclasare.

- Folosește pragmele OpenMP pentru a specifica regiunile paralele și a seta numărul de fire de execuție (thread-uri) dorit.

- Asigură-te că sincronizezi corect firele de execuție și că eviți condițiile de cursă (race conditions) și alte probleme legate de concurență.

2. MPI: MPI (Message Passing Interface) este o bibliotecă utilizată pentru programarea paralelă în sisteme distribuite. Dacă dorești să paralelizezi Mergesort pe mai multe noduri de calcul, poți urma acești pași:

- Distribuie datele inițiale între nodurile de calcul.

- Folosește funcțiile MPI pentru a realiza comunicarea și sincronizarea între noduri.

- Pe fiecare nod, implementează algoritmul Mergesort în mod similar cu varianta secvențială, dar pe datele alocate nodului respectiv.

- Folosește operațiile MPI adecvate pentru a combina și intercala listele parțial sortate într-o singură listă finală.

3. STL: STL este o bibliotecă standard în C++ care oferă algoritmi și structuri de date eficiente. În cazul Mergesort, poți utiliza STL pentru a simplifica implementarea și manipularea listelor:

- Utilizează containterul `std::vector` pentru a reprezenta lista de sortat.

- Folosește algoritmul `std::sort` pentru a sorta subliste individuale sau `std::merge` pentru a combina listele parțial sortate.

- Asigură-te că utilizezi iterații și funcții STL corespunzătoare pentru a manipula datele în mod eficient.