Atividade 3

Profiling - gprof

Integrantes :
Felipe Seiji Momma Valente 12543700
Fernando Gonçalves Campos 12542352
Thiago Shimada 12691032

Descrição da Atividade

Para esta atividade, o grupo optou pela opção 2, escolhendo os algoritmos de ordenação QuickSort, HeapSort e MergeSort. Cada um deles foi executado cem vezes por execução e teve um total de 10 execuções, com a cache sendo limpa entre as execuções de cada Sort, utilizando sempre o mesmo vetor para evitar casos atípicos que pudessem distorcer os resultados. A partir dessas execuções, os dados foram coletados por meio da ferramenta gprof e, em seguida, submetidos a uma análise detalhada.

Dados Obtidos

Total :

- Tempo de execução : 3,594s (+- 0,099s)

Clean Cache :

- Tempo de execução : 1,35s (+- 0,0457)

_Init:

- Tempo de execução : 0,089s (+- 0,0314)

QuickSort :

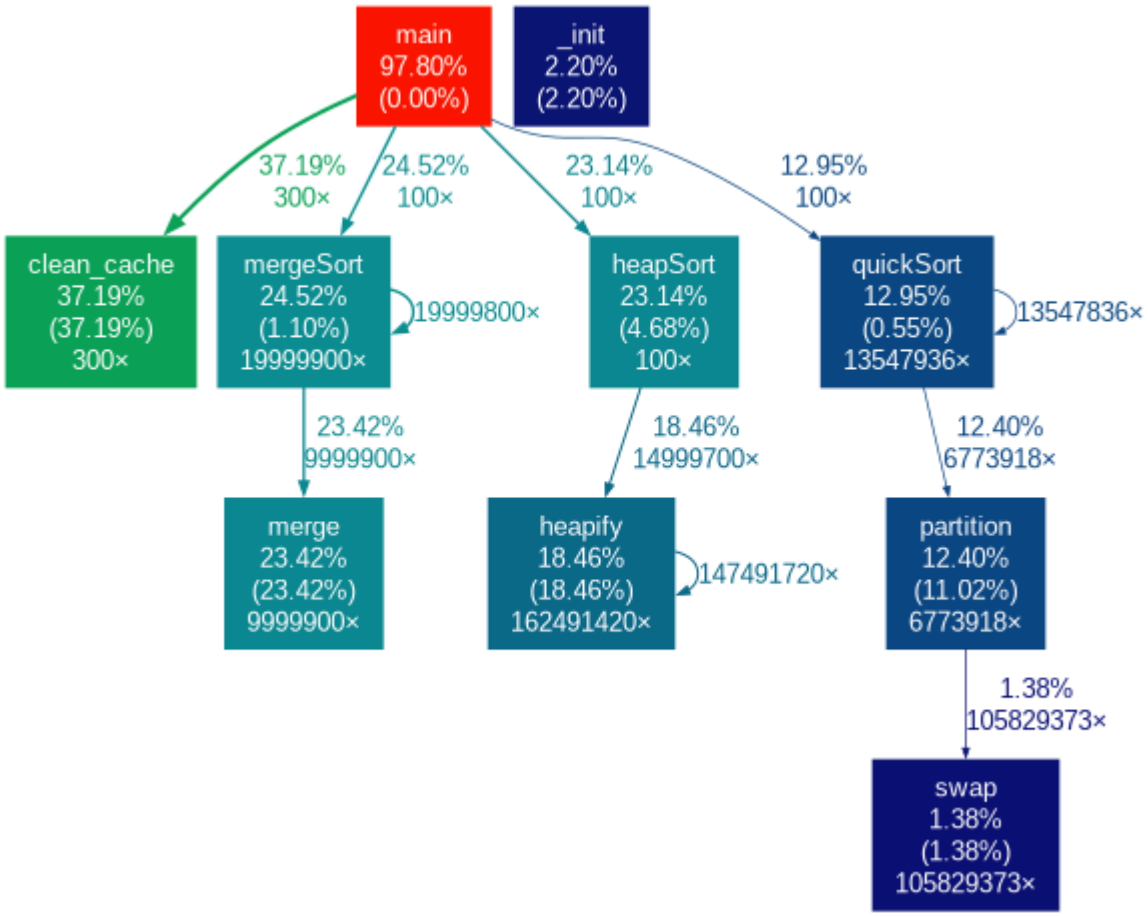
- Tempo de execução : 0,491s (+- 0,0461s)

Heapsort :

- Tempo de execução : 0,819s (+- 0,0885s)

MergeSort :

- Tempo de execução : 0,9s (+- 0,034s)



O grafo gerado pelo gprof mostra a função main como ponto inicial do programa, responsável por chamar os algoritmos de ordenação e também a função auxiliar de limpeza de cache. Observa-se que a função clean_cache consome cerca de 37% do tempo total de execução, sendo executada uma vez antes de cada algoritmo para garantir que os resultados não fossem influenciados pelo armazenamento temporário da CPU.

Em relação aos algoritmos de ordenação, o mergeSort foi responsável por aproximadamente 25% do tempo, com destaque para a função merge, que concentrou quase todo o custo do algoritmo, o que é esperado, já que o processo de intercalação é o ponto central de sua complexidade. O heapSort apresentou tempo semelhante, em torno de 23%, sendo a função heapify o gargalo principal, visto que é chamada repetidamente para manter a propriedade de heap. Já o quickSort teve um consumo menor, cerca de 13% do tempo, sendo a função partition a mais custosa, acompanhada por chamadas adicionais à função swap.

Conclusão

A análise dos resultados evidencia diferenças significativas no desempenho dos algoritmos de ordenação. O QuickSort obteve o menor tempo de execução médio (0,491s). O HeapSort (0,819s) e o MergeSort (0,9s) apresentaram tempos mais altos(mesmo todos

sendo de mesma magnitude), mas vale destacar o MergeSort pela consistência de execução.

Além disso, operações auxiliares, como a limpeza de cache (1,35s) e a inicialização (_Init, 0,089s), representam uma fração considerável da execução. Assim, pode-se concluir que, no cenário analisado, o QuickSort se sobressai, tendo quase metade do tempo de execução dos outros, mostrando que mesmo que todos esses algoritmos eram para ter mesmo tempo de execução teórico, na prática a diferença ainda pode ser grande.