Relatório do Projeto 5

Guilherme Romanholo Bofo

22 de janeiro de 2023

Em um primeiro momento, os códigos com os métodos de ordenação foram feitos com a biblioteca time.h, a fim de analisar o período de tempo apenas na execução do sort, para uma maior precisão.

O primeiro algoritmo analisado foi o Quick Sort. Primeiramente ele foi analisado utilizando a otimização do compilador com a flag -O2, e depois ele foi compilado sem nenhuma otimização. Seguem os resultados obtidos:

Compilado com -O2	Tempo (s)
Teste 1	0.015625
Teste 2	0.015667
Teste 3	0.015671
Teste 4	0.015648
Teste 5	0.015656

Tabela 1: Quick sort.

Compilado sem -O2	Tempo (s)
Teste 1	0.062363
Teste 2	0.062348
Teste 3	0.062381
Teste 4	0.062408
Teste 5	0.062362

Tabela 2: Quick sort.

Além disso, o próximo algoritmo analisado foi o Intertion Sort, o qual também foi testado com a flag de otimização e sem ela. Seguem os resultados obtidos:

Compilado com -O2	Tempo (s)
Teste 1	18.638688
Teste 2	14.949720
Teste 3	18.608763
Teste 4	18.651370
Teste 5	15.930668

Tabela 3: Insertion sort.

Compilado sem -O2	Tempo (s)
Teste 1	69.473902
Teste 2	69.452861
Teste 3	69.173325
Teste 4	69.384726
Teste 5	69.230080

Tabela 4: Insertion sort.

Dessa forma, observando os tempos obtidos conseguimos estimar a diferença de tempo entre os dois algoritmos de ordenação:

- Média Quick Sort: 0,0623724 s
- Média Quick Sort(O2): 0,0156534 s
- Média Insertion Sort: 69,3429788 s
- Média Insertion Sort(O2): 17,3558418 s

Assim, observamos que compilado com ou sem otimização, o Quick Sort é aproximadamente $1111,757$ vezes mais rápido que o Insertion sort em ambos os casos.

Por fim, podemos observar também que o tempo de execução do Insertion Sort é reduzido consideravelmente quando compilamos ele com a flag de otimização, sendo quase 4 vezes mais rápido quando otimizado.