Ana Beatriz Silva Zerati

O problema consistia na ordenação de um vetor de structs com 200 mil posições. A estrutura tinha como propósito armazenar dois valores, “high” e “low”, que em conjunta análise formam um extenso número (maior do que o valor máximo contemplado pelo tipo int ou long int).

Para essa ordenação, foram utilizados dois algoritmos, insertion e quicksort, a fim de comparação. Ambos os algoritmos de ordenação determinam a ordem dos valores com base na comparação com valores do vetor. Desse modo, deve-se considerar na análise a existência de dois parâmetros de comparação, “high” e “low”, de tal forma que a comparação de dois valores de mesmo “high” é guiada pelos valores de “low”.

Sintetizando o funcionamento dos algoritmos, o insertion baseia-se em laços de repetição, começando da posição um do vetor (até a última) percorrendo-o dessa posição ao princípio e “reposicionando” valores maior do que o utilizado na comparação, inserindo-o em sua posição correta. Já o quicksort se baseia na recursão e na escolha de um pivô (última posição na implementação escolhida para o problema), número utilizado para comparação. Com base nessa comparação e em trocas com o elemento “bigger” (posteriormente trocado com o próprio pivô), acabamos com o vetor dividido em dois com o pivô no meio (sendo que metade do vetor possui valores menores que o pivô e a outra metade maiores). Esse processo é repetido recursivamente até que todo o vetor esteja ordenado (até que cheguemos em um “subvetor” com apenas uma posição).

Para a comparação dos métodos, foram gerados cinco arquivos de maneira aleatória, que, por sua vez, foram submetidos a ordenação. Todos os testes foram feitos em um dispositivo de seguintes configurações: processador i5 de 10° geração com 8 GB de RAM. A contagem do tempo foi feita a partir da função clock() da biblioteca “time.h”, que mede o tempo decorrido medido em CLOCKS_PER_SEC desde, chamada ao início do processo (após a leitura do arquivo) e após a ordenação. Os valores obtidos estão reunidos na tabela abaixo:

Insertion sort	quicksort
23.35 s	0.031 s
22.87 s	0.032 s
22.87 s	0.033 s
22.9 s	0.031 s
22.91 s	0.032 s

Assim, obtemos uma média de 22,98 s para o insertionsort e 0,032 s para o quicksort. Fica, então, evidente a diferença de agilidade de ambos os algoritmos quando se tratando de vetores com um número grande de elementos. Isso se comprova mais uma vez ao olharmos a complexidade de ambos, sendo o insertion $\phi(n)$ no melhor caso e $O(n^2)$ no pior, em oposição ao quick, com caso médio $\phi(n * \log(n))$ e pior caso $O(n^2)$.