BCC202 – Estruturas de Dados I (2018-02)

Departamento de Computação - Universidade Federal de Ouro Preto - MG

Aula Prática 7 – Ordenação: HeapSort

Data de entrega: 02/11/2018 às 22:00.
Atenção: *O que vale é o horário do RunCodes, e não do* seu*, ou do* meu*, relógio.*

Procedimento para a entrega:.

1. Para cada questão, implemente sua solução.
2. Para testar suas soluções, implemente um único método *main()*, que poderá conter, por exemplo, um *Menu* de interações e possibilidades do usuário especificar os dados de entrada.
3. Especifique o *Makefile* com as instruções necessárias para compilação e execução do seu código, sendo que o *Makefile* deve conter também o redirecionamento da entrada e da saída (e.g., **`./prog.exe < input.txt > output.txt`**.)
4. Compacte em um único diretório o seu código fonte juntamente com o *Makefile*, o arquivo de entrada e o arquivo de saída usados para testes (e.g., *input.txt, output.txt*).
5. Faça a entrega do arquivo compactado, obrigatoriamente em formato *.zip*, no *RunCodes*, na tarefa correspondente.

- **Não utilize caracteres acentuados ou especiais para nomes de pastas, arquivos e na especificação de comentários no código**.
- **Implemente em conformidade com boas práticas para reuso e modularização do código.**

- Bom trabalho!

Questão 01

Implemente o algoritmo de ordenação: `HeapSort` . O algoritmo receberá vetores de inteiros, conforme descrito a seguir.

1. Gere um vetor de entrada com um milhão de números aleatórios e menores ou iguais a um milhão. Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também o seu tempo de execução (e.g., em segundos).

2. Gere um vetor de entrada com um milhão de números em ordem crescente (i.e., 1,2,3, 4 até 1.000.000). Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também seu tempo de execução (e.g., em segundos).

3. Gere um vetor de entrada com um milhão de números em ordem decrescente (i.e., 1.000.000, 999.999, até 1). Ordene tais números utilizando cada um dos algoritmos de ordenação mencionados. Para cada um dos algoritmos, indique o número de comparações e movimentações realizadas e também seu tempo de execução (e.g., em segundos).

Exemplo de entrada e Saída

A entrada inicia com o número de vetores a serem ordenados. Cada vetor é iniciado pelo número de elementos contidos no vetor e é seguido dos elementos a serem ordenados. No caso, considere o número de vetores e disposição dos elementos conforme descrito anteriormente.

A saída apresenta a quantidade de comparações e movimentos realizados na execução de cada algoritmo e o tempo de execução do algoritmo em uma unidade de tempo definida, seguindo o padrão definido na "Prática 05".

Questão 02

Discuta, de forma comparativa, a complexidade de tempo e de espaço dos algoritmos vistos até o momento, considerando as diferentes configurações de entradas previamente definidas. Aponte os cenários de melhor e pior caso de cada algoritmo.

Questão 03

Mostre, passo a passo, a ordenação do vetor a seguir usando o HeapSort.
v = {0, 5, 3, 10 , 2, 20, 18, 8 }

Importante

Embora seus algoritmos devam ser testados para ordenar vetores com um milhão de elementos e as ordenações de tais vetores sejam utilizadas para discutir a complexidade desses algoritmos, para submissão no RUNCODES, considere vetores com apenas mil elementos. As soluções para as questões teóricas devem estar disponíveis num arquivo *.pdf*, que deve ser compactado juntamente com o código implementado na "Questão 1".