Centro Federal de Educação Tecnológica de Minas Gerais - Departamento de Computação

Laboratório de Algoritmos e Estruturas de Dados II (2ECOM.025)

Lista de Exercícios 4

Professores: Amadeu Almeida e Thiago Rodrigues **Data de entrega:** 16 de Setembro de 2019

Instruções: leia com atenção **todas** as orientações abaixo antes de começar a lista.

- Os exercícios deverão ser feitos individualmente e valem 5 pontos.
- Submeta no SIGAA um arquivo compactado chamado **Lista4.zip**. Ele deverá conter:
  - As classes Java que implementam as soluções das tarefas.
  - Um relatório em **PDF** com os gráficos dos experimentos e suas respectivas explicações.
- O prazo de entrega é até às 16:40 do dia 13 de setembro.
- Códigos fonte em PDF e submissões em atraso serão desconsiderados.
- O enunciado desta lista possui 2 exercícios e 2 páginas.

Exercício 1: Classe Heap

Este exercício objetiva implementar uma classe que cria uma estrutura de dados do tipo **heap**, ordenando-a por meio do algoritmo **Heapsort**

1. Implemente os atributos da classe **Heap**:
   - private final int[] heap;
   - private final int[] heapOrdenado;
   - private int trocasDePosicoes;
2. Implemente os métodos a seguir:
   - public Heap(int [] h): Método construtor.
   - private void construcaoHeap(): Transforma um vetor qualquer em um **heap**.
     Dica: este método pode ser chamado pelo construtor.
   - private void manutencaoHeap(int noAtual, int tamanhoHeap): restaura as propriedades um **heap** sempre que necessário.
   - private int retiraMaximo(int tamanhoHeap): remove o registro de maior valor do **heap** e o reorganiza.
   - public void heapsort(): implementa o algoritmo de ordenação **Heapsort**.
   - public int[] getHeap()
   - public int[] getHeapOrdenado()
   - public int getTrocasDePosicoes()
3. Explique o objetivo de cada um dos métodos implementados no item 2 por meio de comentários no próprio código.

Exercício 2: Experimentos computacionais

As tarefas deste exercício visam testar a complexidade dos métodos de construção e ordenação da classe **Heap** para diferentes tamanhos de vetores.

1. Experimento 1:
   (a) Crie 10 **heaps** distintos contendo N elementos **ORDENADOS EM ORDEM CRESCENTE**, no qual N varia em intervalos de 10.000, entre os números 10.000 e 100.000 e calcule o número de trocas de posições ao longo da criação de cada **heap**.
   Dica: uma troca de posição acontece quando os valores de duas posições do **heap** são permutadas.
   (b) Aplique o algoritmo **Heapsort** para ordenar cada **heap** criado no item (a) e retorne a quantidade de trocas de posições no decorrer de cada ordenação.
2. Experimento 2:
   (a) Gere 10 **heaps** diferentes contendo N elementos **ORDENADOS EM ORDEM DECRESCENTE**, no qual N varia em intervalos de 10.000, entre os números 10.000 e 100.000 e calcule a quantidade de trocas de posições durante a formação de cada **heap**.
   (b) Ordene cada **heap** criado no item (a) utilizando o algoritmo **Heapsort** e compute o número de trocas de posições ao longo de cada uma das ordenações.
3. Experimento 3:
   (a) Crie 10 **heaps** diferentes contendo N elementos **ALEATÓRIOS DISTINTOS**, ou seja, que não se repetem, entre 1 e 500.000, no qual N varia em intervalos de 10.000, entre os números 10.000 e 100.000 e retorne quantas trocas de posições houveram durante a formação de cada **heap**.
   (b) Execute o algoritmo **Heapsort** para ordenar cada **heap** criado no item (a) e calcule a quantidade de trocas de posições durante cada ordenação.
4. Faça dois gráficos:
   - Um que compare, para cada valor de N, o número de trocas de posições durante as criações dos **heaps** dos experimentos 1, 2 e 3 desta lista. O eixo X deste gráfico contém a quantidade de elementos de cada **heap** (N) e o eixo Y é composto pelo número de trocas de posições. Depois, cite se os resultados dos três experimentos são similares ou diferentes e explique os motivos desta equivalência ou discrepância, associando-os com a complexidade da criação de um **heap**.
   - Um que apresente, para cada valor de N, o número de trocas de posições durante as ordenações dos experimentos 1, 2 e 3 desta lista. Neste gráfico, o eixo X refere-se a quantidade de elementos de cada **heap** (N) enquanto os números de trocas são retratados pelo eixo Y. Em seguida, aponte se os resultados dos experimentos são semelhantes ou discrepantes e explique as razões da semelhança ou da diferença, relacionando-as com a complexidade do **Heapsort**.