# Atividade Avaliativa 3

## Tema: Estrutura de Dados Heap

**Prof. Mateus M. Luna** - *mateus_m_luna@ufg.br*
**Prof. André L. Moura** - *andre_moura@ufg.br*

O objetivo dessa tarefa é demonstrar o uso da Estrutura de Dados Heap (também conhecida como Fila de Prioridades) para ordenação de um conjunto de dados composto. A atividade é individual.

Uma empresa deseja acompanhar o endereço eletrônico de seus funcionários que estão trabalhando em homeoffice. Para isso a empresa dispõem de arquivos de log que contém os campos **Primero nome** (*first_name*), **Sobrenome** (*last_name*), **email**, **gênero** (*gender*) e **endereço ip** do acesso (*ip_address*). Os campos estão separado por vírgula e a primeira linha contém o cabeçalho de identificação dos campos (similar à um arquivo CSV). Nesse processo de verificação a empresa deseja ordenar os funcionários **pelo primeiro nome em ordem alfabética**. Nesta versão do projeto, você implementará a ordenação por Heap. No final deste documento está detalhado um possível pseudocódigo para a ordenação, baseando-se na lógica demonstrada em sala de aula de *heapify* do vetor.

Além dos dados ordenados, o sistema deverá apresentar (sempre na tela) **o tempo gasto** para a ordenação, e **o número de comparações e trocas** feitas entre registros. Os arquivos podem conter no máximo 5000 elementos, mas a quantidade exata de elementos não está disponível previamente.

Exemplo do arquivo com 4 registros:

```
first_name,last_name,email,gender,ip_address
Izabel,Brinkworth,ibrinkworth0@xinhuanet.com,Female,124.157.20.133
Jobie,Rope,jrope1@shop-pro.jp,Female,209.132.107.148
Margy,Retallack,mretallack2@qq.com,Female,32.55.159.175
Yolanda,Bartelot,ybartelot3@admin.ch,Female,253.191.240.71
```

Para fins de validação do desempenho da solução proposta e de referência do desempenho de cada método serão utilizados os mesmos exemplos de arquivos com algumas quantidades de registros e condições de ordenação distinta dos dados da atividade anterior. Você pode aproveitar o seu código anterior de leitura de dados, já que o vetor deve conter uma *struct* com as informações todas. Você deve apresentar um relatório contendo, **para cada um dos conjuntos de dados disponibilizados as informações de tempo de execução, número de comparações realizadas e número de movimentações (trocas) feitas**. Nestas tabelas, deixe **separado o total de comparações e movimentação da etapa de conversão do vetor em uma heap**, para que se tenha noção do quanto foi dedicado para nesta parte da tarefa em particular .

O relatório deve ser um documento PDF, enviado para o SIGAA, junto com o código fonte dos algoritmos usados.

PSEUDOCÓDIGO DA ORDENAÇÃO POR HEAP:

```
define PAI ← ( (i - 1) / 2 )
define ESQ ← ( i*2 + 1 )
define DIR ← ( (i*2 + 1) + 1 )
```

```
define heapifica(vetor, i, tam)
      maior ← i

      se ESQ < tam && vetor[ESQ] > vetor[maior]
          maior = ESQ
      fimse
      se DIR < tam && vetor[DIR] > vetor[maior]
          maior = DIR
      fimse

      se maior != i
            troca(vetor[i], vetor[maior])
            heapifica(vetor, maior, tam)
      fimse

      retorna
fimdefine

define constroi_heap (vetor, tam)

      i ← tam-1

      para i ← PAI até i ≥ 0, decrescentemente faça
            heapifica(vetor, i, tam)
      fimpara

fimdefine

define heap_sort (vetor, tam)

      constroi_heap(vetor, tam)

      para i ← tam – 1 até i > 0 decrescentemente faça
            troca(vetor)[i], vetor[0])
            tam ← tam - 1
            heapifica(vetor, 0, tam)
      fimpara

fimdefine
```