# Trabalho de Aquecimento: Top K Elementos

Michel Pires, Centro Federal de Educação Tecnológica de Minas Gerais August 2, 2023

**Observações:**

1. O trabalho deve ser realizado individualmente, logo, qualquer indício de cópia será desconsiderada para avaliação.

2. O trabalho entregue deve obrigatoriamente executar em sistema operacional Linux com gcc/g++ na versão 11 ou superior, sendo de responsabilidade do aluno validar possíveis erros de compilação e execução. O processo de compilação deve seguir os padrões já pré estabelecidos na disciplina.

3. Linguagem de programação aceitas: C / C++.

---

Um exemplo clássico de problema que pode ser solucionado utilizando-se hash e heap é o chamado top k itens. Neste problema, é preciso encontrar os k itens mais valiosos de uma coleção de dados. Logo, utiliza-se o hash para contar a frequência de todos os itens, enquanto o heap se aplica na manutenção de uma lista dos k itens de maior valor. Sabendo-se disso, elabore uma solução em C/C++ que dado uma entrada:

- Crie uma tabela de dispersão (hash) para contar a frequência de cada elemento tokenizado da coleção de dados de entrada.
- Crie uma árvore de prioridades (heap) de tamanho k e insira os primeiros k elementos do hash nela.
    1. Para cada elemento restante na hash, compare a contagem com o menor valor do heap.
    2. Se a contagem for maior do que o menor valor da heap, remova o menor valor, insira o novo elemento e refaça a estrutura.
    3. Caso contrário, ignore o elemento e vá para o próximo.
- No final, a heap conterá os k elementos com maiores valores (frequências) da coleção de dados. Então, imprima-os em ordem crescente.

Esse algoritmo é uma combinação eficiente do uso de hash para contar a frequência dos elementos e heap para manter a lista dos k elementos com maiores valores. Sua complexidade, caso implementado adequadamente, é de $\mathcal{O}(nlogk)$, onde $n$ é o tamanho da coleção de dados e $k$ o número de itens mais relevantes.

# 1 Considerações

- Este programa deverá ler uma coleção de arquivos contento textos sem nenhuma formatação ("arquivo ASCII") onde cada sentença termina por um sinal de pontuação (".", "?", "!"").
- Cada parágrafo é separado por, pelo menos, uma linha em branco.
- Considere como palavra uma sequência de letras delimitada por espaço em branco, ”coluna da esquerda”, ”coluna da direita” e símbolos de pontuação.
- Faz parte do projeto do seu programa fornecer uma saída legível.
- Os arquivos a serem utilizados para teste serão disponibilizados junto a essa descrição e devem ser utilizados para quaisquer validações, não sendo necessário a adoção de mais entradas para esse processo.
- A documentação detalhando todo o processo de desenvolvimento deve ser produzida como README.mb e disponibilizada no git junto com o projeto.

# 2 Documentação

A documentação a ser produzida deve conter, pelo menos, as seguintes partes:

- Um detalhamento mínimo que explique as fases de especificação, projeto e implementação. Nessa etapa, deve-se incluir uma ampla discussão sobre as estruturas utilizadas e o motivo de sua escolha para manter o desempenho da solução.
- Para os arquivos utilizados para teste, uma descrição da saída esperada.
- Uma parte do README.md contendo todas as instruções necessárias para a execução de seu trabalho para arquivos de entrada diferentes dos adotados durante os testes.
- Um git contendo todo o projeto, bem como, toda descrição pertinente sobre sua execução, projeto e implementação. Neste documento (README.mb), espera-se observar uma discussão sobre as estruturas adotadas e o motivo de tal utilização. Nesse último ponto, tente realizar um paralelo com outras estruturas para demonstrar de fato que as estruturas adotadas são as melhores possíveis para o problema em questão.

**Data de Entrega**: 25/08/2023
**Valor**: 10 pontos