# Trabalho Prático 3
# Algoritmos 1

**Gabriel de Araújo Marques (2019054544)**

**Universidade Federal De Minas Gerais (UFMG)**
**Belo Horizonte - MG - Brasil**

## 1 - Introdução

Foi requisitado que se desenvolvesse uma solução que calculasse o menor custo para se chegar a um destino através do metrô da *Big Apple*, de forma que a cada escala feita em um determinado intervalo de tempo há a aplicação de descontos progressivos, sendo possível fazer paradas em cada escala para reiniciar os descontos aplicados.

## 2 - Implementação

O programa foi desenvolvido em C++ em um ambiente Linux e compilado utilizando o G++.

### 2.1 - Modelagem Computacional

Para resolver o problema, foi utilizada uma matriz triangular superior contendo os custos para se chegar a cada escala, sendo cada coluna da matriz preenchida com o custo para se chegar à respectiva escala. Para se saber o menor custo para chegar a uma escala, pegamos o menor valor da coluna com seus custos.

### 2.2 - Estrutura de Dados

Para representar cada escala, foi criada uma classe Stopover, que contém atributos para representar o tempo da viagem e o preço cheio do bilhete e métodos para acessar o tempo e calcular o valor do bilhete dado um desconto percentual. Na função principal do programa, são definidos vetores para representar os descontos

e as escalas. Além disso, é definida uma matriz para ser a matriz triangular superior de custos.

## 2.3 - Algoritmos

Para se resolver o problema, foi utilizada uma solução bottom-up, calculando primeiro o custo mínimo para cada escala para então encontrar o custo mínimo final. Itera-se sobre o vetor de escalas, montando uma linha da matriz triangular superior para cada escala. Cada linha i representa os custos para se percorrer cada escala, fazendo uma parada na escala i, partindo do menor custo para se chegar até a escala i, que é dado pelo valor mínimo da coluna anterior a i. A linha 3, por exemplo, representa os custos para se percorrer cada escala fazendo uma parada antes da escala 3, dado que se chegou à escala 3 com o menor custo possível. Ao final da iteração, tem-se a matriz com todas as linhas devidamente preenchidas, então pegamos o valor mínimo da última coluna, que representa o menor custo.

```
getMenorCusto(escalas, descontos, maxEscalas, maxIntervalo, totalEscalas) {
  custos := matrix[totalEscalas][totalEscalas]

  for (i in 0:totalEscalas) {
    intervaloAtual := 0
    escalasVisitadas := 0
    descontoAtual := 0.0
    precoAcumulado := min(custos[0:totalEscalas][i-1])

    for (j in i:totalEscalas) {
      descontoAtual += descontos[escalasVisitadas]
      custos[i][j] = escalas[i].preco - escalas[i].preco*descontoAtual/100

      escalasVisitadas++
      intervaloAtual += escalas[i].tempo

      if intervaloAtual > maxIntervalo OR escalasVisitadas >= maxEscalas {
        descontoAtual := 0
        intervaloAtual := 0
        escalasVisitadas := 1
      }
    }
  }

  return min(custos[0:totalEscalas][totalEscalas - 1])
}
```

## 3 - Análise de Complexidade Assintótica de Tempo

Para calcular o custo mínimo para se percorrer n escalas, é necessário construir a matriz de custos de dimensões n x n. Por se tratar de uma matriz triangular superior, cada linha construída tem um tamanho menor que a anterior em uma unidade. Enquanto a linha 1 tem n itens diferentes de 0, a linha 2 tem n-1 e a linha 3 tem n-2 e assim até a linha n que possui 1 item diferente de zero. Dessa forma, temos a complexidade dada por $n^2 - \sum_{i=0}^{n-1} i$. No infinito, esse valor tende a $n^2$, portanto temos uma complexidade assintótica de tempo de $\Theta(n^2)$.

## 4 - Conclusão

O trabalho apresentou um problema muito interessante, se tratando de um problema que não podia ser resolvido com um algoritmo guloso. Foi possível criar um algoritmo com complexidade quadrática para se resolver o problema, atendendo aos requisitos de tempo do trabalho (tempo máximo de execução de 5 segundos). Para tentar resolver o problema, em um primeiro momento, tentei utilizar uma solução de força bruta, porém antes mesmo de finalizar a implementação eu percebi como a solução não teria um tempo de execução razoável. Então, revi com mais cuidado as aulas sobre programação dinâmica, o que me levou a tentar uma solução por memoização de valores, porém não consegui concretizar a ideia. Foi então que consegui chegar à solução bottom-up, através da construção da matriz de custos que resultou na solução final do trabalho.

## 5 - Referências

Slides e aulas da professora Olga