

|  |
| --- |
| Relatório do TP2 de AED  2023 |
|  |
| 26 junho  ISMAT – Algoritmia e Estrutura de Dados  Da autoria de: Cayan Prola e Diogo Fernandes Professor: Tiago Candeias |



# Introdução

|  |
| --- |
| Foram nos dados dois data sets diferentes, um contendo a correlação entre as submissões de trabalhos de alunos e outro contendo o tempo da entrega de cada trabalho. Nosso objetivo foi de criar um programa em Java capaz de calcular a porcentagem de quais grupos podem ter copiado de outros efetuando a comparação dessa correlação entre os grupos, e utilizando um limite específico para obter apenas os valores que tem maior porcentagem entre si.  Nos foi dado também os argumentos necessários para correr o programa, que são o seguinte:  E o output esperado de: . Desenvolvimento |

Inicialmente, dividimos o trabalho em partes. Precisamos primeiro ler os dados a partir dos data sets, que são dois arquivos “csv” chamados de “correl.csv” e “time.csv”, contendo as correlações entre as entregas e o tempo da entrega respectivamente. Para isso, dividimos os dados em origem, destino e seu peso, podendo assim efetuar as comparações necessárias. Utilizamos um algoritmo para verificar a posição atual, se estamos na primeira linha, salvamos os “headers”, caso não, salvamos os valores de cada “header” iterando sobre todas as linhas. Conseguindo ter os valores salvos, precisamos efetuar suas comparações.

Em seguida criamos a classe Grafo, que executa muitas funções. A primeira é criar um mapa capaz de salvar os valores que necessitamos. Este mapa guarda os vértices e os nós do grafo.

Nós utilizamos sempre as variáveis “origem”, “destino” e “valor” para termos a origem do vértice, o seu destino e qual o peso entre os dois.

Em seguida utilizamos o método “add” para adicionar todos os valores que precisamos ao grafo e utilizamos uma verificação para caso a origem não tenha sido adicionada, criamos um “HashMap” interno e o colocamos no mapa, usando a “origem” como chave. Em seguida, recuperamos esse mapa interno correspondente ao vértice “origem” e colocamos o vértice “destino” e seu peso associado nele, que é a variável “valor”.

Após isso, temos o método “get”. Este método verifica se o mapa contém o vértice “origem” e o vértice “destino” e retorna o peso associados a ambos. E caso nulo, retorna 0.

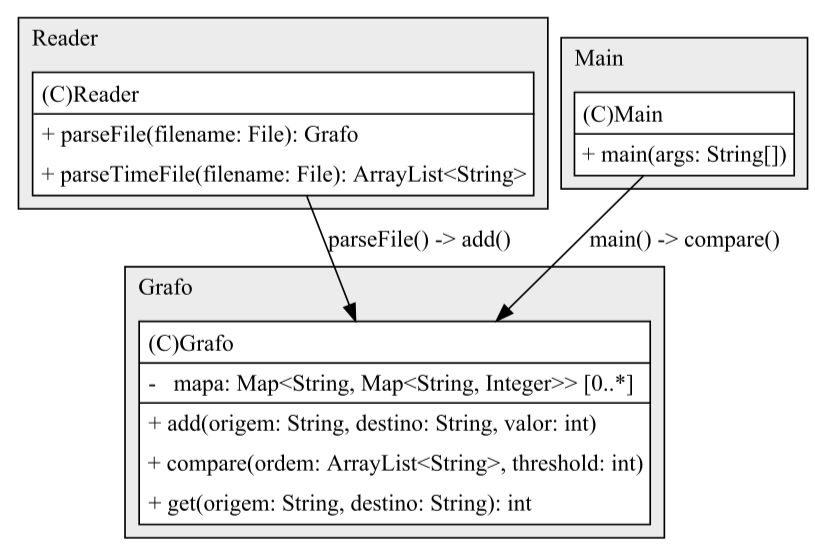
E por último utilizamos o método “compare” que compara o peso das arestas em relação ao limite nos dado anteriormente. Salvamos esse limite em uma variável através dos argumentos do programa e o passamos como parâmetro do método junto com a variável “ordem” que contem a ordem dos vértices a serem comparados a partir do ficheiro “time.csv”, tendo assim a comparação baseada na ordem de entrega. Para obtermos o resultado, usamos um método “foreach” para comparar cada valor, usamos um “if” para verificar se o valor entre a comparação é maior do que o limite passado, caso seja, damos print naquela chave. Por último calculamos a porcentagem somando o peso de cada vértice e dividindo pelo número de números.

## Nosso resultado:

A picture containing text, screenshot, font, line

Description automatically generated

## Diagrama de classes:



Classe main:

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Classe Reader:

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Classe Grafo: A picture containing text, screenshot, font

Description automatically generatedUma imagem com texto, captura de ecrã

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

# Conclusão

Conseguimos pôr em prática os conhecimentos obtidos na cadeira de algoritmia e estrutura de dados, como grafos e o uso de estratégias de concepçao de algoritmos para dividirmos melhor o nosso problema e resolvê-lo de uma maneira mais pratica e rápida.