**Trabalho 2 - Teoria dos Grafos**
**Grupo: Lucas Rolim e Anderson Barbosa**

Neste segundo trabalho, foi adicionada à biblioteca a possibilidade de manipular grafos com pesos e para trabalhar com ela foram criados novos métodos, são eles: *dijkstra*, que calcula a distância e o caminho mínimo entre um vértice e todos os outros usando o algoritmo de Dijkstra, *prim*, que gera uma árvore geradora mínima do grafo utilizando o algoritmo de Prim e *mean_dist*, que calcula a distância média do grafo.

# Decisões de implementação

## Armazenamento dos pesos das arestas

Para grafos com pesos, foi utilizado o contenedor map da biblioteca STL, nativa do C++. Para representar as aresta, foi utilizada a estrutura pair que possui dois atributos, T1 e T2, que, neste caso, representam os vértices. Como para este trabalho a biblioteca deve manipular apenas grafos não-direcionados, uma dada aresta (u,v) é representada pelo pair da seguinte forma: T1 = max(u,v) e T2 = min(u,v). Desta forma, o pair é utilizado como chave para o map e o valor desta chave é o peso da respectiva aresta.

## Distância e caminho mínimo

Para calcular a distância e o caminho mínimo entre um vértice e todos os outros foi utilizado o algoritmo de Dijkstra. As distâncias foram armazenadas no vetor min_dist e os caminhos foram determinados a partir de um vetor parent, sendo que ambos os vetores são atributos da classe Graph. Na fase inicial do desenvolvimento foi estudada a possibilidade de criar um vetor de vetor de int que conteria todos os caminhos mínimos mas, após implementar e testar o algoritmo, foi constatado que essa seria uma opção extremamente custosa em relação à memória e que a utilização de um vetor de pais seria muito mais vantajosa.

## Árvore geradora mínima

Para obter a árvore geradora mínima foi utilizado o algoritmo de Prim, pois sua implementação é consideravelmente mais simples do que a de outros algoritmos, como o algoritmo de Kruskal. A biblioteca gera um arquivo chamado “mst.txt” contendo a MST no mesmo formato utilizado para representar grafos e seu custo total é escrito na última linha do arquivo.

## Métodos search

A classe Graph possui 2 métodos chamados search, pois utilizamos a técnica de polimorfismo. O primeiro recebe como parâmetros 2 inteiros que representam, respectivamente, o vértice de origem e o vértice até o qual se quer chegar, sendo este opcional. Este método usa o método dijkstra caso o grafo possua pesos e usa o método bfs caso contrário. O segundo método search recebe como parâmetros duas sequências de caracteres que representam os nomes de duas pessoas entre as quais se quer encontrar o caminho mínimo na rede de colaboração entre pesquisadores. Este método utiliza o arquivo

que mapeia os números dos vértices na rede de colaboração para nomes de pessoas e, desta forma, exibe os nomes que formam o caminho mínimo entre dois determinados pesquisadores.

### Classe myHeap

Para construir o algoritmos de Dijkstra e de Prim com complexidade O((n+m)log(n)) é necessário utilizar um *heap* para armazenar os vértices e remover sempre aquele com o menor custo/distância. A linguagem C++, utilizada neste projeto, possui um estrutura de *heap*, porém esta não permite que seus elementos sejam atualizados e por isso optou-se pela implementação de um *heap* próprio, o myHeap, que conta com a possibilidade de atualizar o custo/distância de um vértice.

## Estudo de Caso

| Grafo_ 1 | Grafo_2 |
|---|---|
| **Distância entre os vértices 10 e 1:** 31<br>**Caminho entre os vértices 10 e 1:** 10 268 484 798 133 710 709 881 1000 1<br>**Tempo de Execução:** 0.006406<br>**Distância entre os vértices 20 e 1:** 38<br>**Caminho entre os vértices 20 e 1:** 20 21 416 141 604 4 3 2 1<br>**Tempo de Execução:** 0.007258<br>**Distância entre os vértices 30 e 1:** 48<br>**Caminho entre os vértices 30 e 1:** 30 29 28 134 74 709 881 1000 1<br>**Tempo de Execução:** 0.007081<br>**Distância entre os vértices 40 e 1:** 25<br>**Caminho entre os vértices 40 e 1:** 40 74 709 881 1000 1<br>**Tempo de Execução:** 0.006978<br>**Distância entre os vértices 50 e 1:** 30<br>**Caminho entre os vértices 50 e 1:** 50 51 919 768 881 1000 1<br>**Tempo de Execução:** 0.022768 | **Distância entre os vértices 10 e 1:** 9<br>**Caminho entre os vértices 10 e 1:** 10 8767 9110 8275 5675 1<br>**Tempo de Execução:** 0.506351<br>**Distância entre os vértices 20 e 1:** 10<br>**Caminho entre os vértices 20 e 1:** 20 21 979 4992 670 3985 1<br>**Tempo de Execução:** 0.495682<br>**Distância entre os vértices 30 e 1:** 9<br>**Caminho entre os vértices 30 e 1:** 30 5021 5020 9823 2538 3985 1<br>**Tempo de Execução:** 0.503755<br>**Distância entre os vértices 40 e 1:** 8<br>**Caminho entre os vértices 40 e 1:** 40 6804 6758 1423 2173 2538 3985 1<br>**Tempo de Execução:** 0.50285<br>**Distância entre os vértices 50 e 1:** 7<br>**Caminho entre os vértices 50 e 1:** 50 7672 857 3985 1<br>**Tempo de Execução:** 0.506095 |

| Grafo_3 | Grafo_4 |
|---|---|
| **Distância entre os vértices 10 e 1:** 5<br>**Caminho entre os vértices 10 e 1:** 10 99499 93705 52193 1<br>**Tempo de Execução:** 10.1023<br>**Distância entre os vértices 20 e 1:** 10<br>**Caminho entre os vértices 20 e 1:** 20 21 95822 10027 95946 27287 9824 43169 43170 1 | **Distância entre os vértices 10 e 1:** 58<br>**Caminho entre os vértices 10 e 1:** 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 6.06804<br>**Distância entre os vértices 20 e 1:** 129<br>**Caminho entre os vértices 20 e 1:** 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 |

| | |
|---|---|
| **Tempo de Execução:** 10.122<br>**Distância entre os vértices 30 e 1:** 16<br>**Caminho entre os vértices 30 e 1:** 30 31 60660 44171 95621 82358 15899 1<br>**Tempo de Execução:** 10.2022<br>**Distância entre os vértices 40 e 1:** 8<br>**Caminho entre os vértices 40 e 1:** 40 27630 2722 2435 15899 1<br>**Tempo de Execução:** 10.1218<br>**Distância entre os vértices 50 e 1:** 7<br>**Caminho entre os vértices 50 e 1:** 50 21939 93705 52193 1<br>**Tempo de Execução:** 10.15 | **Tempo de Execução:** 6.29246<br>**Distância entre os vértices 30 e 1:** 102<br>**Caminho entre os vértices 30 e 1:** 30 31 19296 627814 627813 627812 723121 723120 103161 103162 103163 126309 164224 387113 556565 15847 101894 814196 814195 396781 396782 5 4 3 2 1<br>**Tempo de Execução:** 6.24136<br>**Distância entre os vértices 40 e 1:** 174<br>**Caminho entre os vértices 40 e 1:** 40 39 38 37 36 35 34 33 32 31 19296 627814 627813 627812 723121 723120 103161 103162 103163 126309 164224 387113 556565 15847 101894 814196 814195 396781 396782 5 4 3 2 1<br>**Tempo de Execução:** 6.58068<br>**Distância entre os vértices 50 e 1:** 147<br>**Caminho entre os vértices 50 e 1:** 50 51 52 53 54 55 390888 286933 906093 285753 285752 285751 285750 285749 285748 285747 929180 596549 811148 811147 811146 5 4 3 2 1<br>**Tempo de Execução:** 6.67916 |

| Grafo_5 | Rede_de_Colaboração |
|---|---|
| **Distância entre os vértices 10 e 1:** 93<br>**Caminho entre os vértices 10 e 1:** 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 125.332<br>**Distância entre os vértices 20 e 1:** 150<br>**Caminho entre os vértices 20 e 1:** 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 123.771<br>**Distância entre os vértices 30 e 1:** 241<br>**Caminho entre os vértices 30 e 1:** 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 123.998<br>**Distância entre os vértices 40 e 1:** 332<br>**Caminho entre os vértices 40 e 1:** 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 123.77<br>**Distância entre os vértices 50 e 1:** 400<br>**Caminho entre os vértices 50 e 1:** 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1<br>**Tempo de Execução:** 123.55 | **Distância entre os vértices 10 e 1:** 2.80655<br>**Caminho entre os vértices 10 e 1:** 10 2574 83518 258146 7783 3244 1<br>**Tempo de Execução:** 21.0493<br>**Distância entre os vértices 20 e 1:** 2.37936<br>**Caminho entre os vértices 20 e 1:** 20 3984 353044 5161 9971 2524 4586 12822 537589 3244 1<br>**Tempo de Execução:** 21.0412<br>**Distância entre os vértices 30 e 1:** 2.68396<br>**Caminho entre os vértices 30 e 1:** 30 4106 220549 688962 209457 7314 5306 3244 1<br>**Tempo de Execução:** 21.0312<br>**Distância entre os vértices 40 e 1:** 3.1792<br>**Caminho entre os vértices 40 e 1:** 40 5265 272393 3244 1<br>**Tempo de Execução:** 20.987<br>**Distância entre os vértices 50 e 1:** 2.42058<br>**Caminho entre os vértices 50 e 1:** 50 556113 650795 12573 13155 13071 6885 426314 3244 1<br>**Tempo de Execução:** 21.2487 |

**2.**

| | Peso MST | T(MST) |
|---|---|---|
| Grafo 1 | 3856 | 0.010881 |
| Grafo 2 | 31663 | 0.150025 |
| Grafo 3 | 295994 | 1.53566 |
| Grafo 4 | 2.92568e+06 | 1.46804 |
| Grafo 5 | 7.99973e+07 | 40.8184 |

**3.**

| | d_med | T(d_med) |
|---|---|---|
| Grafo 1 | 18.4561 | 6.46756 |
| Grafo 2 | 6.81081 | 1667.33 |
| Grafo 3 | | |
| Grafo 4 | | |
| Grafo 5 | | |

**1.**
**Turing:** 2.14748e+09
**Kruskal:** 3.48037
**Kleinberg:** 2.70699
**Tardos:** 2.75351
**Daniel:** 2.94283

**Tempo de Execução para o cálculo de todas as distâncias juntas:** 23.908

**2. Obtenha uma árvore geradora mínima, e responda às seguintes perguntas:**
**(a) Determine os três vértices de maior grau na MST.**
**(b) Determine os vizinhos de Edsger W. Dijkstra e de Daniel R. Figueiredo na MST.**

**(a)**

Edsger W. Dijkstra,John R. Rice,Dan C. Marinescu,Chuang Lin,Bo Li,Y. Thomas Hou,Zhi-Li Zhang,Donald F. Towsley,Daniel R. Figueiredo

**(b)**

Wei Li 173; Wei Wang 146; Wei Zhang 142.