Teoria dos Grafos - Trabalho de Disciplina – Parte 2

Amanda Lucio, Lucas Maximo

1 Local do arquivo

O repositório para o código está no link `https://github.com/AmandaACLucio/GraphLibrary`.Para acessar a readme do repositório é possível utilizar o link anterior ou acessar `https://github.com/AmandaACLucio/GraphLibrary#readme`

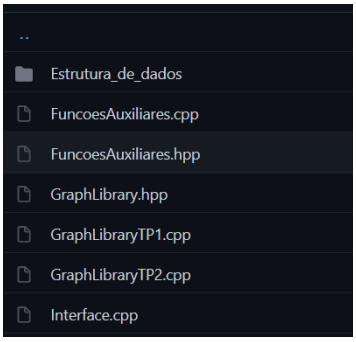
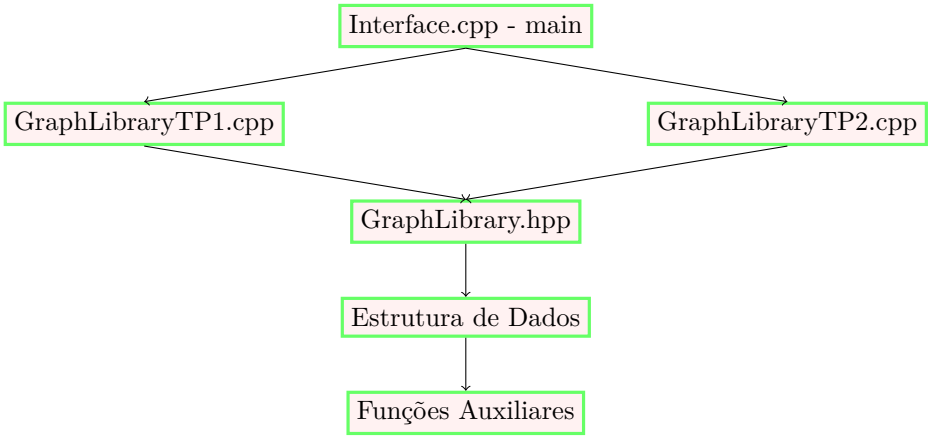


Figura 1: Print do github

2 Estrutura do Código

Os arquivos de códigos podem ser estruturadas da seguinte forma por hierarquia e chamada.



No arquivo funções auxiliares tem o uso de Estrutura de dados para construções de algumas funções

3 Explicando o código

- **Estruturas de Dados**: Foi criado as três estruturas de dados, lista de adjacência, matriz de adjacência e vetor de adjacência, Evitando a utilização de bibliotecas de estruturas de dados já prontas, com objetivo de se obter mais controle sobre o retorno das funções.
- **Grafos**: Originalmente a classe grafo contém três estruturas de dados (lista de adjacência, matriz de adjacência e vetor de adjacência) que carregam na entrada o grafo para as mesmas, permitindo que o usuário possa escolhar qual estrutura utilizar. Para contagem das arestas é verificado se ocorreu ou não adição do nó e seus os nós da mesma não são iguais. A classe inclui as função de

entrada, saída, BFS, DFS, ComponentesConexas (gera txt com todas as componentes), Diâmetro, Distância e análise se duas variáveis são de uma mesma componente. As alterações nessa classe permitiu inicialmente a inserção ordenada, de forma que não seja mais necessário utilizar sort para aplicação de algoritmos de melhor caminho. Além disso, foram inseridas variáveis booleanas peso e direcao, permitindo que o usuário decida qual o formato do grafo inserido.

- **Funções Auxiliares**: As funções de auxiliares servem de suporte para construção da classe grafos. A partir delas podemos preencher e mostrar um vetor. Além disso, é possível fazer a separação de strings de um texto, permitindo que os números das linhas sejam separados em dois. Outras funções incluem mapear um inteiro para um booleano, permitindo que o calculo de componentes conexas seja efetuado através da análise se um vertíce já teve sua posição em um vetor mapeada como true ou não.

4 Adaptações no código

- **Peso**: Para adaptadação da estrutura, foram utilizados adição de atributo nos nodes de vertice e lista de adjacência, e uma segunda matriz do tipo float, que possibilitou que grafos sem peso não tivesse seu desempenho de memória modificado.
- **Otimização da entrada**: A classe grafo contém três estruturas de dados (lista de adjacência, matriz de adjacência e vetor de adjacência) que carregam na entrada o grafo para as mesmas, permitindo que o usuário possa escolhar qual estrutura utilizar. Em um vetor populado (função que preenche todas as posições necessárias com 0) armazenamos os nós, e em outros se os mesmos existem ou não, já que os nós são preenchidos automaticamente na entrada, tratando a possibilidade de um nó saltando e adicionando-os ordenado. Para contagem das arestas é verificado se ocorreu ou não adição do nó e seus os nós da mesma não são iguais. A classe inclui a função de entrada, saída, BFS, DFS, ComponentesConexas (gera txt com todas as componentes), Diâmetro, Distância(com peso e sem peso) e análise se duas variáveis são de uma mesma componente, Dijkstra, MST e Excentricidade.
- **Novas funções auxiliares**: Foi escrita uma função capaz de escrever um grafo a partir de um vetor pai e uma variável custoTotal. A mesma sofreu sobrecarga para se ajusta a um grafo com peso e sem peso

5 Determinação da distância e o caminho mínimo

Tabela 1: Calculo da distância com inicio no vértice 1

Grafo	Fim 10	Fim 20	Fim 30	Fim 40	Fim 50
1.txt	0.97	1.2	0.69	1.06	1.31
2.txt	1.7	1.68	2.39	1.86	2.29
3.txt	1.91	1.97	2.78	2.41	2.07
4.txt	2.57	2.58	2.6	2.58	2.2
5.txt	14.23	18.4	16.55	17.94	13.55

inf vertices pertencentes a componentes diferentes
- refere-se a valor não existente ou não computado

6 Excentricidade

Tabela 2: Determinacao da Excentricidade

Grafo	10	20	30	40	50
1.txt	2	2	2	2	2
2.txt	3	2	3	3	3
3.txt	4	4	5	4	4
4.txt	5	4	5	5	5
5.txt	25	28	25	27	25

Tabela 3: Tempo de execução da excentricidade (s)

Grafo	10	20	30	40	50
1.txt	0.52	0.05	0.052	0.051	0.051
2.txt	0.695	0.651	0.619	0.634	0.62
3.txt	8.906	8.896	8.959	8.957	8.913
4.txt	282.361	277.015	273.908	273.555	277.296
5.txt	4903.1	5565.83	6158.21	5349.36	4727.61

- refere-se a valor não existente ou não computado

7 Tempo médio para calculo da excentricidade

Tabela 4: Determinação do Tempo médio para calcular a excentricidade com k = 50

Grafo	Tempo de execução médio (s)
1.txt	0.05028
2.txt	0.65538
3.txt	9.02826
4.txt	272.456
5.txt	4640.15

8 Árvore Geradora - MST

Tabela 5: Determinacao do custo mínimo e tempo de execução

Grafo	Custo Mínimo	Tempo de execução (ms)
1.txt	222.54	32
2.txt	2182.82	477
3.txt	22107.1	4944
4.txt	222138	66848
5.txt	5.641580e+06	278339

- refere-se a valor não existente ou não computado

9 Rede de colaboração

Distância e o caminho mínimo entre Edsger W. Dijkstra (2722) (o pesquisador) e: Alan M. Turing(11365), J. B. Kruskal (471365), Jon M.Kleinberg (5709), Eva Tardos (11386), Daniel R. Figueiredo (343930).

Tabela 6: Determinacao de distância de Edsger W. Dijkstra (2722)

Grafo	Custo Mínimo
Alan M. Turing (11365)	3.5596
J. B. Kruskal (471365)	2.84031
Jon M.Kleinberg (5709)	3.0976
Éva Tardos (11386)	4.20556
Daniel R. Figueiredo (343930)	2.96503