Sistemas de Informação — DECSI - UFOP
CSI105 Algoritmos e Estruturas de Dados III — 2022/1
Professor: Dr. George Henrique Godim da Fonseca — Data: 26/09/22

Aluno: Matrícula: Valor: 50,0 Nota:

Atividades Práticas

1. Objetivos.

   - Desenvolver a habilidade de programação de algoritmos em grafos.
   - Avaliar o impacto da complexidade computacional no tempo de execução.
   - Aplicar os conhecimentos em algoritmos para resolver problemas reais.

2. Descrição.

   As atividades práticas ligadas à disciplinas de Algoritmos e Estruturas de Dados III são organizadas em três etapas: (i) Criação de uma biblioteca de algoritmos em grafos; (ii) Implementação e análise de algoritmos para o problema do caminho mínimo; e (iii) Aplicação da biblioteca para resolver um problema real com grafos. Cada uma dessas atividades será descrita em detalhes nas próximas seções.

   1. Biblioteca de Algoritmos em Grafos [20 pontos]

   A primeira atividade prática, que será usada como base para completar as demais, consiste do desenvolvimento de uma biblioteca contendo diversas funções para manipular e resolver problemas em grafos. Boa parte dessa biblioteca será construída nas aulas práticas com o auxílio do professor. Essa biblioteca deve ser composta por ao menos três classes:

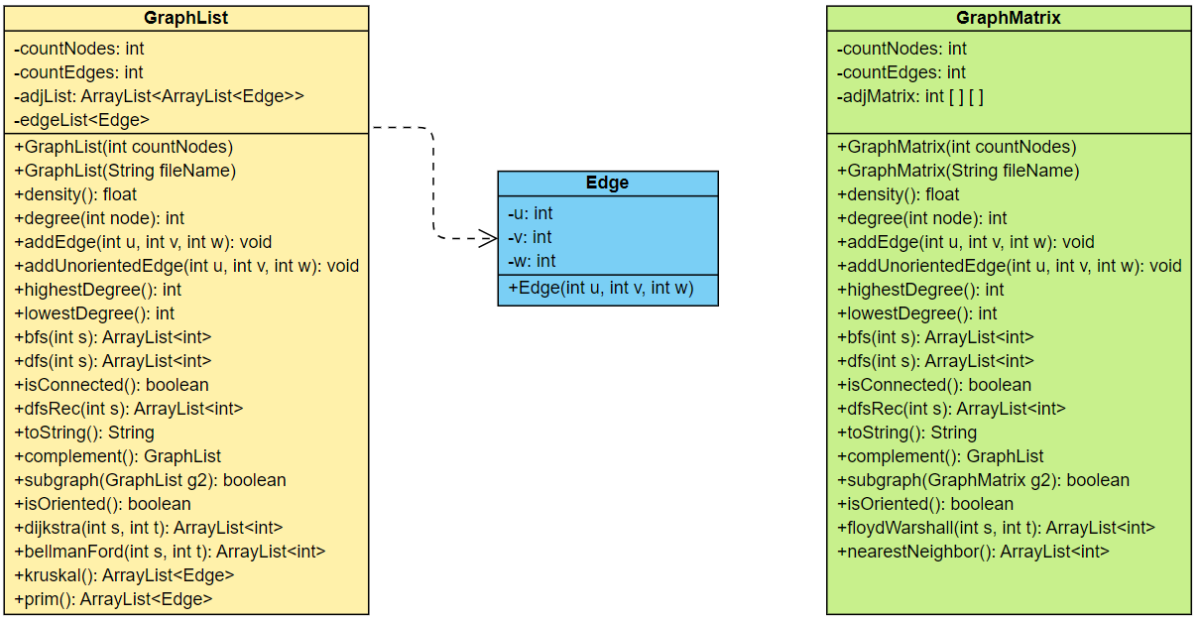
   GraphMatrix Contém implementações de algoritmos em grafos utilizando matriz de adjacências;

   GraphList Composta de implementações de algoritmos em grafos utilizando lista de adjacências;

   Edge Representação de uma aresta no formato origem (u), destino (v) e peso (w).

   O código base desenvolvido ao longo das aulas, contendo ainda a assinatura de cada uma dessas funções e uma breve descrição está disponível no GitHub[1]. Veja na figura a seguir o diagrama de classes sugerido para a biblioteca.

[1] `https://github.com/georgehgfonseca/GraphLib`



2. Análise de Algoritmos de Caminho Mínimo [10 pontos]

Durante as aulas foi ressaltado que, frequentemente, há diversos algoritmos para resolverem um mesmo problema computacional. Cada algoritmo é aplicado de acordo com as características em específico do problema a ser resolvido, ou se estamos mais interessados em obter melhor eficiência de tempo no pior caso, no caso médio, ou ainda se a complexidade de espaço é crítica.

Essa etapa consiste em implementar os algoritmos de *Dijkstra*, *Bellman-Ford* e *Floyd-Warshall* para encontrar o caminho mínimo entre dois nós s e t, que deverão ser informados pelo usuário. A versão aprimorada do algoritmo de *Bellman-Ford* também deve ser implementada. Os tempos de execução de cada um desses 4 algoritmos para os grafos abaixo, anexos ao enunciado, devem ser reportados em um relatório cujo modelo foi disponibilizado também em anexo.

Arquivo	Descrição
toy.txt	Exemplo dado em aula, que pode ser checado manualmente.
rg300_4730.txt	Grafo aleatório com 300 vértices e 4730 arestas.
rome99c.txt	Descreve um mapa das estradas de Roma em 1999.
facebook_combined.txt	Conexões entre contatos do Facebook (com 4039 usuários).
USA-road-dt.DC.txt	Descreve um mapa das estradas de Washington DC.
USA-road-dt.NY.txt	Descreve um mapa das estradas de New York.

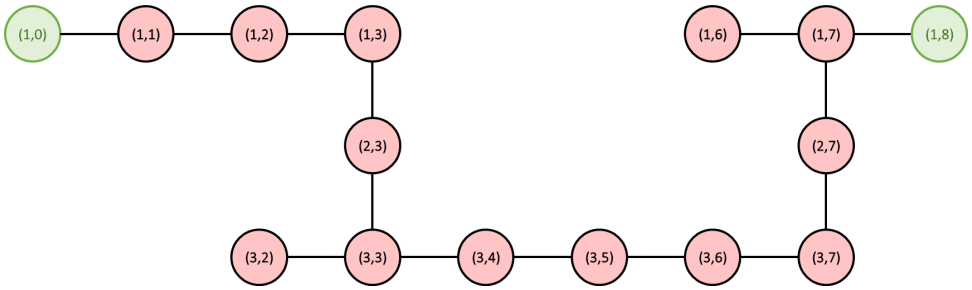
Sugiro utilizar a instância "toy" para testar a corretude dos algoritmos visto que é fácil verificar seu resultado manualmente.

3. Solução do Problema do Labirinto [20 pontos]

Grafos podem ser utilizados para representar e resolver computacionalmente uma vasta quantia de problemas. Nessa atividade resolveremos o problema do labirinto através de algoritmos em grafos. O labirinto será representado por uma matriz onde cada posição traz informação sobre determinada coordenada do labirinto. Essa matriz será informada ao programa através de um arquivo de texto, com apenas quatro possíveis caracteres: '`#`' representando paredes, ' ' para as passagens, '`S`' representando o ponto de início e '`E`', a saída do labirinto. Veja um exemplo de arquivo abaixo:

```
#########
S   ##  E
### ### #
#       #
#########
```

Esse labirinto pode ser resolvido ao encontrar qualquer caminho de `S` (nó (1,0)) a `E` (nó (1,8)), no grafo apresentado a seguir.



Nesse exemplo é fácil notar que o único caminho de `S` a `T` é:

```
(1,0) (1,1) (1,2) (1,3) (2,3) (3,3) (3,4) (3,5) (3,6) (3,7) (2,7) (1,7)
   (1,8)
```

Foram gerados 5 labirintos de teste com auxílio do site Maze Generator [2]. Os arquivos também estão em anexo a esse enunciado, além do relatório a ser preenchido, reportando os resultados obtidos.

[2] `https://www.dcode.fr/maze-generator`

3. Interação com o usuário

   A interação com o usuário deve ocorrer na função `main` do seu programa. Primeiramente, o programa deve perguntar ao usuário qual tarefa a resolver (1 - Caminho Mínimo; 2 - Labirinto; 3 - Sair). Caso a opção 1 seja escolhida, deve-se informar ainda o arquivo de entrada, o algoritmo a executar e os nós de origem e destino do caminho. Ao final da execução o programa deve exibir o tempo de execução que levou, o caminho mínimo de s a t e o custo desse caminho. Caso a opção 2 tenha sido escolhida, o programa deve solicitar ao usuário o arquivo de entrada e, após a execução, informar o caminho de `S` a `E`, bem como o tempo de execução. Segue um exemplo de interação com o programa:

```
Informe a tarefa:
  1 - Caminho Mínimo
  2 - Labirinto
  3 - Sair
<1>
Arquivo: <cm/toy.txt>
Origem: <0>
Destino: <3>
Processando...
Caminho: [0, 1, 2, 3]
Custo: 5
Tempo: 0.003s

Informe a tarefa:
1 - Caminho Mínimo
2 - Labirinto
3 - Sair
<2>
Arquivo: <maze/toy.txt>
Processando...
Caminho: (1,0) (1,1) (1,2) (1,3) (2,3) (3,3) (3,4) (3,5) (3,6) (3,7) (2,7)
    (1,7) (1,8)
Tempo: 0.005s

Informe a tarefa:
1 - Caminho Mínimo
2 - Labirinto
3 - Sair
<3>
```

4. Avaliação.

   O trabalho deverá ser feito individualmente ou em dupla. O relatório deve ser enviado via Moodle até as 23:59h do dia 26/10/22. As implementações serão avaliadas com relação à corretude e à eficiência (tempo de execução). Caso se tenha alguma dúvida com relação à autoria do trabalho o professor poderá solicitar uma apresentação presencial ao aluno (ou dupla).

Bom trabalho!