Algoritmos para Grafos

Trabalho 2

Professor: Tadeu Zubaran

.

1 Definição

Este trabalho consiste em projetar e implementar o **Algoritmo de Bellman-Ford**. O trabalho deve ser feito ou em C++(recomendado) ou em C.

Você deve ter as seguintes opções:

1. Mostrar cada iteração do algoritmo, cada iteração do for da linha 4 do algoritmo 1. Mostre o estado dos vetores d e um caminho mínimo para cada vértice.
2. Mostrar o tempo de execução do algoritmo.

Algorithm 1 Algoritmo de Bellman-Ford

```
1: procedure BellmanFord(G = (V, E, w), s)
2:     InicializaFonteUnica(G, s)
3:     // Relaxando as arestas
4:     for i = 1 : n − 1 do
5:         for e = (u, v) ∈ E do
6:             Relaxa(u, v, w)
7:     //Testando se evitamos ciclo negativo
8:     for e = (u, v) ∈ E do
9:         if d[v] > d[u] + w[(u, v)] then return false
       return true
```

2 Testes

2.1 Dígrafo Fixo

Mostre a as iterações do algoritmo (opção 1), iniciando no vértice $s = 0$, para o **dígrafo** com a seguinte matriz de adjacência:

```
0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Ordenando as arestas pelo primeiro índice do par ordenado e usando o segundo como critério desempate, os custos das arestas são:

$$5, -4, -3, 2, 8, 9, 2, 6, 3, 9, 4, 7, 1, 2, -1$$

2.2 Dígrafo Aleatório

Crie dígrafos aleatórios, onde cada aresta tem 20% de chance de existir. Para cada número n no intervalo $[1, 1000]$ crie exatamente um grafo com número de vértices igual a n, totalizando 1000 grafos. Esse dígrafo não deve ter endo-arcos (loops) ou arcos paralelos no mesmo sentido (em sentido contrário eles podem existir). Os pesos nas arestas são inteiros aleatórios no intervalo $[-5, 50]$

Execute o algoritmo guardando o número de acessos á estrutura de dados. Faça um gráfico no qual o

- eixo vertical mostra número de acessos à memória, e o
- eixo horizontal mostra o número de vértices.

Você pode usar o software da sua preferência para fazer esse gráfico.

3 Avaliação

Não copie o código. Faça seu próprio código! Plágio receberá nota 0.

Mostre o seu código em aula executando para todos o casos de teste, e explique em detalhe o seu projeto e seu código.

Critérios de avaliação:

- Clareza e corretude da **explicação** do código.
- Clareza e corretude do código.
- Explicação das decisões de projeto.
- Código implementado com boas práticas de programação.

- Explicações dos gráficos gerados.
- Domínio da conexão da implementação com a teoria.

4 Avaliação

Não copie o código. Faça seu próprio código! Plágio receberá nota 0. Critérios de avaliação:

- Clareza e corretude da **explicação** do código.
- Clareza e corretude do código.
- Explicação das decisões de projeto.
- Código implementado com boas práticas de programação.
- Explicações dos gráficos gerados.
- Domínio da conexão da implementação com a teoria.

5 Entrega

A entrega consiste de duas coisas. Uma apresentação para o professor em aula e upload do código no google class.

O upload do código deve ser feito no class. **Faça o upload apenas do seu *.cpp (ou *.c) e de seus *.h (ou *.hpp)**. Caso você faça seu próprio makefile faça o upload dele também. **Não compacte**, você pode fazer upload de mais de um arquivo em cada tarefa.