Implementação 3

Vitor Lúcio

October 10, 2024

1 Introdução

Esta documentação explica o funcionamento e as especificações de três algoritmos implementados em C++. Todos eles lidam somente com grafos direcionados e ponderados somente com valores positivos. Os algoritmos abordados são:

- Algoritmo de Dijkstra
- Algoritmo de Dijkstra (MinMax)
- Algoritmo de Dijkstra (MaxMin)

2 Algoritmo de Dijkstra

O algoritmo de Dijkstra é utilizado para encontrar o menor caminho entre um nó inicial e todos os outros nós de um grafo.

2.1 Especificações

- O código começa lendo o número de vértices, arestas e o destino desejado.
- Depois lê as relações entre os vertices. (vertice x, aresta pra y, peso da aresta)
- O grafo é representado por um vetor de listas de pares, onde cada par contém um vértice adjacente e o peso da aresta que o conecta.
- Utiliza-se uma fila de prioridade (min-heap) para processar os vértices em ordem crescente de distância.

2.2 Exemplo

Entrada	**Saída**
4 5 3 0 1 3 0 2 1 1 2 3 1 3 1 2 3 3	Caminho: 0 2 3 Distâncias: 0 3 1 4

2.3 Explicação

O código segue os seguintes passos:

1. Inicializa o grafo e os vetores de distâncias e antecessores.
2. Insere o vértice inicial na fila de prioridade com distância 0.
3. Em cada iteração, retira o vértice com menor distância e atualiza as distâncias dos vértices adjacentes .
4. Repete o processo até que o destino seja alcançado ou a fila esteja vazia.
5. Por fim, constrói o caminho mínimo de volta utilizando o vetor de antecessores.

3 Algoritmo MaxMin

O algoritmo `maxMin.cpp` é uma variação de Dijkstra que procura calcular o máximo valor mínimo em um conjunto de dados. A lógica básica segue a estrutura de busca em grafos, com foco em garantir a propagação da menor capacidade da aresta, priorizando a maior capacidade dos caminhos possíveis.

3.1 Especificações

- Utiliza um vetor de distâncias para armazenar os valores máximos e mínimos calculados em cada iteração.
- O grafo é representado de forma semelhante ao do algoritmo de Dijkstra.

3.2 Exemplo

Entrada	**Saída**
4 5 3 0 1 3 0 2 1 1 2 3 1 3 1 2 3 3	Caminho: 0 1 2 3 Distâncias: ∞ 3 3 3

3.3 Explicação

O código segue os seguintes passos:

1. Inicializa o grafo e os vetores de distâncias e antecessores.
2. Insere o vértice inicial na fila de prioridade com distância ∞.
3. Em cada iteração, propago o vértice com menor peso, de forma que a menor capacidade se mantenha, dando prioridade para o caminho com maior capacidade.
4. Repete o processo até que o destino seja alcançado ou a fila esteja vazia.
5. Por fim, constrói o caminho mínimo de volta utilizando o vetor de antecessores.

4 Algoritmo MinMax

No arquivo `minMax.cpp`, temos a implementação de um algoritmo que visa calcular o menor valor máximo. A lógica básica segue a estrutura de busca em grafos, com foco em garantir a propagação da maior capacidade da aresta, priorizando a menor capacidade dos caminhos possíveis. Esse tipo de algoritmo é muito útil em problemas de otimização onde se deseja minimizar a maior perda possível.

4.1 Especificações

- Utiliza técnicas similares ao algoritmo MaxMin, mas com um foco inverso.
- O vetor de distâncias mantém o menor valor máximo possível em cada iteração.

4.2 Exemplo

Entrada	**Saída**
4 5 3 0 1 3 0 2 1 1 2 3 1 3 1 2 3 3	Caminho: 0 1 3 Distâncias: ∞ 3 1 3

4.3 Explicação

O código segue os seguintes passos:

1. Inicializa o grafo e os vetores de distâncias e antecessores.
2. Insere o vértice inicial na fila de prioridade com distância -∞.
3. Em cada iteração, propago o vértice com maior peso, de forma que a maior capacidade se mantenha, dando prioridade para o caminho com menor capacidade.
4. Repete o processo até que o destino seja alcançado ou a fila esteja vazia.
5. Por fim, constrói o caminho mínimo de volta utilizando o vetor de antecessores.