Documentação Implementação 3

Arthur de Sá Braz de Matos

Esta é a documentação da implementação 3 da disciplina de Teoria dos Grafos e Computabilidade. Os códigos foram desenvolvidos em C++ e tem como objetivo implementar os algoritmos de Dijkstra, MinMax e MaxMin para grafos conexos, direcionados e com pesos positivos, bem como possíveis aplicações no mundo real.

1 Dijkstra

1.1 Estruturas de Dados

O algoritmo utiliza as seguintes estruturas de dados:

- **Vertices e Visitados**: Duas listas:
  - `vertices`: Contém todos os vértices do grafo.
  - `visitados`: Armazena os vértices que já foram processados pelo algoritmo.
- **Distâncias**: Um vetor `distancias` guarda a menor distância encontrada até o momento de cada vértice em relação ao vértice de origem. Inicialmente, todas as distâncias são definidas como ∞, exceto a distância do vértice inicial, que é 0.
- **Matriz de Adjacência**: Uma matriz bidimensional `matriz` guarda os pesos das arestas que conectam os vértices. O valor na posição `matriz[i][j]` representa o peso da aresta do vértice i para o vértice j.

1.2 Funcionamento do Algoritmo

O método principal é `shortestPath(int u)`, que executa os seguintes passos:

1. **Inicialização**: A distância do vértice de origem (u) para si mesmo é definida como 0.
2. **Laço Principal**: Enquanto houver vértices não visitados, o algoritmo seleciona aquele com a menor distância acumulada utilizando a função `menorDistancia()`.
3. **Atualização de Distâncias**: Para o vértice com menor distância acumulada, o algoritmo atualiza as distâncias dos seus vértices vizinhos. A nova distância é calculada como:

   $$\texttt{distancias[v]} = \min\Big(\texttt{distancias[v]},\\ \texttt{distancias[vMenorDistancia]} + \texttt{matriz[vMenorDistancia][v]}\Big)$$

   Dessa forma, pegamos o menor valor entre o atual e o valor do vértice de saída somado com o tamanho da aresta que os conecta. Sendo `vMenorDistancia` o vértice com a menor distância acumulada, encontrado anteriormente.

4. **Marcação de Visitados**: Após atualizar as distâncias dos vizinhos, o vértice é marcado como visitado e o processo se repete até que todos os vértices sejam visitados.

5. **Retorno das Distâncias**: Ao final, o vetor `distancias` é retornado, contendo a menor distância de todos os vértices em relação ao vértice de origem.

1.3 Função `menorDistancia()`

A função `menorDistancia()` percorre os vértices não visitados e retorna aquele que possui a menor distância acumulada. Se todos os vértices foram visitados ou não há mais caminhos possíveis, o algoritmo termina.

1.4 Exemplo de Uso

O arquivo de entrada `graph1.graph` contém a definição dos vértices e suas arestas com os respectivos pesos. O usuário escolhe um vértice inicial e um vértice final. O programa então calcula e exibe a menor distância entre esses dois vértices utilizando o algoritmo de Dijkstra.

1.5 Resultado

Ao final da execução, o programa imprime a menor distância entre os vértices de origem e destino escolhidos, ou exibe uma mensagem de erro caso não exista um caminho possível. Além disso, o vetor de distâncias completo é exibido.

2 Algoritmo MinMax

O algoritmo MinMax é utilizado para encontrar o caminho que minimiza o peso máximo entre os vértices de um grafo ponderado. Esse algoritmo é uma variante do problema de menor caminho, mas em vez de buscar minimizar a soma dos pesos das arestas ao longo de um caminho, ele visa minimizar o valor máximo entre essas arestas.

Abaixo está a explicação detalhada do algoritmo:

2.1 Descrição

Dado um grafo ponderado representado por uma matriz de adjacência `matriz`, onde o valor `matriz`$[i][j]$ representa o peso da aresta que vai do vértice i para o vértice j, o algoritmo MinMax realiza os seguintes passos:

1. Inicialmente, todos os vértices têm seus pesos definidos como ∞, exceto o vértice de origem u, cujo peso é $-\infty$.

2. A cada iteração, escolhe-se o vértice com o menor peso ainda não visitado e marca-o como visitado.

3. Para cada vizinho v desse vértice, calcula-se o maior valor entre o peso atual do vértice u e o peso da aresta que liga u a v. O valor obtido é comparado com o peso do vértice v, e o menor valor entre eles é atualizado no vetor de pesos.

4. O processo se repete até que todos os vértices tenham sido visitados ou não haja mais vértices acessíveis.

O objetivo final do algoritmo é encontrar o menor valor máximo de um caminho entre o vértice inicial e os outros vértices do grafo. Esse valor é armazenado na variável `valorMinMax`, que é atualizado a cada iteração.

2.2 Equações

A equação utilizada no algoritmo é a seguinte:

$$\texttt{valorMinMax} = \min\left(\max(\texttt{pesos}[vMenorPeso], \texttt{matriz}[vMenorPeso][v]), \texttt{pesos}[v]\right)$$

Essa equação compara o peso do caminho entre os vértices e atualiza o peso do vértice vizinho, garantindo que o peso máximo ao longo do caminho seja minimizado.

3 Algoritmo MaxMin

O algoritmo MaxMin é utilizado para encontrar o caminho que maximiza o menor peso entre os vértices de um grafo ponderado. Diferentemente do algoritmo MinMax, onde buscamos minimizar o valor máximo das arestas, o MaxMin busca maximizar o valor mínimo encontrado ao longo de um caminho.

3.1 Descrição

Dado um grafo ponderado representado por uma matriz de adjacência `matriz`, onde o valor $\texttt{matriz}[i][j]$ representa o peso da aresta que vai do vértice i para o vértice j, o algoritmo MaxMin realiza os seguintes passos:

1. Inicialmente, todos os vértices têm seus pesos definidos como $-\infty$, exceto o vértice de origem u, cujo peso é ∞.

2. Para cada iteração, escolhe-se o vértice com o maior peso ainda não visitado, com base no número de arestas não percorridas. Caso esse vértice não tenha mais arestas adjacentes, ele é marcado como visitado.

3. Para cada vizinho v, calcula-se o menor valor entre o peso atual do vértice u e o peso da aresta que liga u a v. O valor obtido é comparado com o peso do vértice v, e o maior valor entre eles é atualizado no vetor de pesos.

4. O processo se repete até que todos os vértices tenham sido visitados ou não haja mais vértices acessíveis.

O objetivo do algoritmo é maximizar o valor mínimo de um caminho entre o vértice inicial e os outros vértices do grafo. Esse valor é armazenado na variável `valorMaxMin`, que é atualizado durante cada iteração.

3.2 Equações

A equação fundamental utilizada no algoritmo MaxMin é a seguinte:

$$\texttt{valorMaxMin} = \max\left(\min(\texttt{pesos}[vMaiorPeso], \texttt{matriz}[vMaiorPeso][v]), \texttt{pesos}[v]\right)$$

Essa equação garante que o peso de cada caminho seja atualizado considerando o mínimo entre os pesos da aresta atual e o maior valor mínimo calculado até o momento.

4 Aplicações reais

4.1 Dijkstra

1. Navegação GPS e Sistemas de Mapas: Softwares como Google Maps e Waze utilizam o Dijkstra para calcular a rota mais curta entre dois locais, levando em consideração a distância, tempo de viagem ou evitar pedágios.

2. Transporte Público: Sistemas de transporte como trens e ônibus usam o Dijkstra para otimizar rotas, garantindo que passageiros possam chegar ao destino final da forma mais rápida possível, considerando conexões, tempos de espera, etc.

4.2 MinMax

1. Planejamento Urbano: O MinMax pode ser aplicado em planejamento urbano. Por exemplo, ao construir hospitais, o objetivo pode ser minimizar a distância máxima que os cidadãos terão que percorrer para acessar esse serviço, garantindo que todos estejam relativamente próximos.

2. Design de Redes Elétricas: Ao projetar uma rede elétrica, é importante minimizar a carga máxima em qualquer ponto da rede para evitar falhas e sobrecargas. O algoritmo MinMax pode ser utilizado para distribuir a carga de maneira eficiente, minimizando o risco de um ponto crítico ser sobrecarregado.

4.3 MaxMin

1. Design de Redes de Energia: No design de redes elétricas, o algoritmo MaxMin pode ser utilizado para garantir que a menor carga transmitida em qualquer ponto da rede seja maximizada, assegurando que nenhum segmento da rede receba uma carga muito baixa, o que poderia causar instabilidades.

2. Gerenciamento de Riscos em Investimentos: O algoritmo MaxMin pode ser aplicado na diversificação de carteiras de investimento. A ideia é maximizar o mínimo retorno esperado, garantindo que o pior retorno seja o melhor possível, minimizando os riscos associados a perdas financeiras.