UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

LUIZ HENRIQUE SERAFIM DA SILVA

ALGORITMO DE DIJKSTRA

Presidente Prudente
2023

1.0 Introdução

O algoritmo de Dijkstra é um algoritmo clássico utilizado para encontrar o caminho mais curto entre um nó de origem e todos os outros nós em um grafo ponderado, ou seja, um grafo em que cada aresta possui um peso ou custo associado. Ele foi desenvolvido pelo cientista da computação holandês Edsger W. Dijkstra em 1956.

O objetivo do algoritmo de Dijkstra é determinar a menor distância acumulada de um nó de origem para todos os outros nós no grafo. Ele funciona de forma iterativa, expandindo os nós a partir da origem, sempre escolhendo o próximo nó que possui a menor distância acumulada conhecida. O algoritmo mantém um conjunto de nós visitados e um conjunto de nós não visitados.

2.0 Sequência de Passos

2.1 Inicialização

- Defina a distância da origem para ela mesma como 0 e a distância de todas as outras origens como infinito (ou um valor muito grande).
- Coloque todos os nós no conjunto de nós não visitados.

2.2 Escolha o nó mais próximo não visitado

- Procure o nó não visitado com a menor distância acumulada a partir da origem. Na primeira iteração, esse nó será a própria origem, pois sua distância é definida como 0.

2.3 Atualização das distâncias

- Para o nó escolhido, analise todas as arestas que saem dele e atualize as distâncias acumuladas dos nós adjacentes, se necessário.
- Se a distância acumulada do nó atual + o peso da aresta para um nó adjacente for menor do que a distância atual desse nó adjacente, atualize a distância acumulada.
- Repita esse processo para todos os nós adjacentes ainda não visitados.

2.4 Marcar o nó como visitado

- Após atualizar as distâncias de todos os nós adjacentes, marque o nó atual como visitado e remova-o do conjunto de nós não visitados.

2.5 Repetir

- Continue repetindo os passos 2, 3 e 4 até que todos os nós tenham sido visitados. Nesse ponto, todas as distâncias acumuladas representam as menores distâncias a partir da origem para cada nó.

Após a conclusão do algoritmo de Dijkstra, você terá a menor distância acumulada de todos os nós da origem e poderá construir o caminho mais curto para qualquer nó de destino percorrendo as arestas com as menores distâncias acumuladas de volta à origem. O algoritmo de Dijkstra é eficiente para grafos densos, com complexidade de tempo O(V^2) usando uma matriz de adjacência, onde V é o número de nós no grafo. Entretanto, com algumas otimizações, como o uso de uma fila de prioridades (min-heap) para armazenar os nós não visitados e selecionar o próximo nó de forma eficiente, o algoritmo pode ser melhorado para O(E + V log V), onde E é o número de arestas. Essa versão otimizada é geralmente preferida em grafos grandes e densos.

3.0 Explicação Código

1. Importação de bibliotecas:

- “import heapq”: É usado para implementar a fila de prioridades, que é uma estrutura de dados fundamental para o algoritmo de Dijkstra.
- “import os”: É usado para executar comandos do sistema operacional, neste caso, é utilizado para limpar a tela (os.system("cls")).

2. Classe Grafo:

- É definida uma classe chamada Grafo para representar o grafo ponderado.
- O construtor “__init__” inicializa o grafo com um dicionário vazio chamado “vertices”, que será usado para armazenar os vértices e suas respectivas arestas e pesos.

3. Métodos da classe Grafo:

- “adicionar_vertice”: Recebe um vértice como argumento e adiciona-o ao grafo com um dicionário vazio associado a ele. Esse dicionário vazio será usado para armazenar as arestas que partem desse vértice.
- “adicionar_aresta”: Recebe dois vértices e um peso como argumentos e adiciona uma aresta bidirecional (não direcionada) entre esses vértices com o peso especificado.
- “dijkstra”: Recebe um vértice de origem e um vértice de destino como argumentos e retorna a distância mínima entre eles e o caminho mais curto (lista de vértices) percorrido para alcançar o destino.

4. Lógica do Algoritmo de Dijkstra:

- O algoritmo utiliza a estratégia de uma fila de prioridades (implementada com um heap) para garantir que os vértices mais próximos da origem sejam explorados primeiro.
- Inicialmente, todas as distâncias são definidas como infinito, exceto a origem, que é definida como 0, pois a distância dela até ela mesma é zero.
- A fila de prioridades é inicializada com a origem e sua distância (0).

- Enquanto houver vértices na fila de prioridades, o algoritmo explora os vizinhos do vértice de menor distância (vértice mais próximo).
- Para cada vértice adjacente ao vértice atual, o algoritmo calcula a distância até esse vértice, passando pelo vértice atual. Se essa distância for menor do que a distância atualmente registrada para o vértice adjacente, a distância é atualizada, e o vértice adjacente é adicionado à fila de prioridades para ser explorado posteriormente.
- O processo continua até que o destino seja alcançado na exploração ou até que não haja mais vértices na fila de prioridades.

5. Criação do Grafo de Exemplo:

- Um objeto “rede” é criado a partir da classe “Grafo”.
- Os vértices A, B, C, D, E, F são adicionados ao grafo.

6. Adição das Arestas e Pesos (interação com o usuário):

- O usuário é solicitado a adicionar arestas e pesos para o grafo.
- As arestas são adicionadas usando o método “adicionar_aresta” do objeto “rede”.

7. Encontrando o Caminho Mais Curto:

- O usuário é solicitado a fornecer o vértice de origem e destino.
- O algoritmo de Dijkstra é invocado através do método “dijkstra” do objeto “rede”.
- O resultado é a distância mínima entre os vértices de origem e destino e o caminho mais curto para percorrer de um para o outro.

8. Imprimindo o Resultado:

- O resultado é impresso mostrando o caminho mais curto e a distância total percorrida.

4.0 Exemplo Teste

Execute o programa e na entrada dos dados insira as especificações pedidas, O exemplo de entrada utilizado para as arestas e os valores para o teste foi:

A
B
2
A
C
4
B
C
1
B

D
7
C
E
3
D
E
2
D
F
5
E
F
1
sair
A
F

Então ele coloca o menor caminho percorrido do nò “A” até o nó “F” e escreve o valor do caminho.