



Algoritmo de Djikstra

(Redes de Computadores)

Lucas Vinícius de Carvalho Ikeda

Sumário

1. Introdução	3
2. Descrição do algoritmo	3
3. Funcionamento	4
4. Conclusão	6
5. Referências	7

1. Introdução

O algoritmo de Dijkstra é uma das técnicas fundamentais em teoria dos grafos e resolução de problemas de caminho mais curto. Ele foi proposto pelo renomado cientista da computação Edsger W. Dijkstra em 1956. O algoritmo é amplamente utilizado em diversas aplicações práticas, tais como roteamento em redes de computadores, planejamento de rotas em sistemas de transporte, otimização de logística e muitos outros cenários que envolvem encontrar o caminho mais eficiente entre dois pontos em um grafo ponderado.

O principal objetivo do algoritmo de Dijkstra é determinar o caminho mais curto entre um nó de origem e todos os outros nós em um grafo ponderado, seja ele direcionado ou não direcionado. Neste contexto, o termo "caminho mais curto" refere-se ao trajeto que resulta em menor soma dos pesos das arestas, onde os pesos representam o custo, distância ou qualquer outra métrica associada à aresta.

O algoritmo de Dijkstra possui uma abordagem gulosa (greedy), o que significa que ele sempre escolhe a opção localmente ótima em cada etapa para encontrar a solução global mais eficiente. Entretanto, é importante destacar que essa estratégia de escolha localmente ótima somente é válida quando todos os pesos das arestas do grafo são não negativos. Caso contrário, o algoritmo pode não produzir resultados corretos.

O algoritmo de Dijkstra é especialmente útil em situações em que é necessário calcular rotas mais curtas em tempo real ou quando o grafo não sofre alterações frequentes, pois ele garante a descoberta das distâncias mínimas de forma eficiente.

2. Descrição do algoritmo

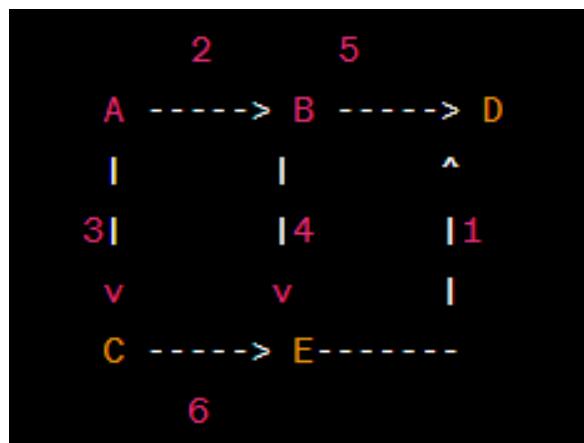
O algoritmo de Dijkstra funciona da seguinte forma:

1. **Inicialização:** Atribuímos distâncias infinitas para todos os nós, exceto para o nó de origem, que terá distância zero. Criamos um conjunto de nós não visitados que inicialmente inclui todos os nós do grafo.

2. **Escolha do nó de menor distância:** Selecionamos o nó de origem e marcamos como visitado. Em cada iteração, selecionamos o nó com a menor distância conhecida entre os nós não visitados.
3. **Relaxamento das arestas:** Para o nó selecionado, examinamos todos os seus vizinhos não visitados e atualizamos suas distâncias conhecidas caso encontremos um caminho mais curto. Para isso, calculamos a distância atual do nó de origem até o nó vizinho através do nó selecionado e a comparamos com a distância atualmente conhecida do nó vizinho. Se a distância atual for menor, atualizamos a distância do nó vizinho e definimos o nó selecionado como o nó pai do nó vizinho no caminho mais curto.
4. **Marcação como visitado:** Após examinar todos os vizinhos do nó selecionado, marcamos o nó selecionado como visitado e o removemos do conjunto de nós não visitados.
5. **Iteração:** Repetimos os passos 2 a 4 até que todos os nós tenham sido visitados.

3. Funcionamento

Vamos exemplificar o funcionamento do algoritmo em um grafo teórico para encontrar o caminho mais curto de um nó de origem "A" para todos os outros nós. Suponha o seguinte grafo ponderado:



1. **Inicialização:** Atribuímos distância infinita para todos os nós, exceto o nó "A", que possui distância 0. Definimos o nó "A" como o nó de origem.
2. **Escolha do nó de menor distância:** Começamos com o nó "A" e o marcamos como visitado.

3. **Relaxamento das arestas:** Atualizamos as distâncias conhecidas para os vizinhos de "A":
 - A distância para "B" é 2 (A->B).
 - A distância para "C" é 3 (A->C).
4. **Marcação como visitado:** Marcamos o nó "A" como visitado e o removemos do conjunto de nós não visitados.
5. **Iteração:** Neste ponto, o nó "C" tem a menor distância conhecida (3). Selecionamos "C" como o novo nó atual.
6. **Relaxamento das arestas:** Atualizamos a distância conhecida para o vizinho "E":
 - A distância para "E" é 6 (C->E).
7. **Marcação como visitado:** Marcamos o nó "C" como visitado e o removemos do conjunto de nós não visitados.
8. **Iteração:** O nó com a menor distância conhecida é "B" (distância = 2). Selecionamos "B" como o novo nó atual.
9. **Relaxamento das arestas:** Atualizamos as distâncias conhecidas para os vizinhos de "B":
 - A distância para "D" é 5 (B->D).
10. **Marcação como visitado:** Marcamos o nó "B" como visitado e o removemos do conjunto de nós não visitados.
11. **Iteração:** O nó com a menor distância conhecida é "E" (distância = 6). Selecionamos "E" como o novo nó atual.
12. **Relaxamento das arestas:** Não há vizinhos não visitados a partir de "E", portanto, não há atualizações a serem feitas.
13. **Marcação como visitado:** Marcamos o nó "E" como visitado e o removemos do conjunto de nós não visitados.
14. **Iteração:** O único nó restante é "D". Selecionamos "D" como o novo nó atual.
15. **Relaxamento das arestas:** Não há vizinhos não visitados a partir de "D", portanto, não há atualizações a serem feitas.
16. **Marcação como visitado:** Marcamos o nó "D" como visitado e o removemos do conjunto de nós não visitados.

Após concluir todas as iterações, obtemos as distâncias mínimas a partir do nó "A" para todos os outros nós:

```
Distâncias mínimas a partir de 'A':  
A: 0  
B: 2  
C: 3  
D: 7  
E: 6  
F: 13
```

4. Conclusão

O algoritmo de Dijkstra é uma abordagem eficiente e confiável para encontrar o caminho mais curto em um grafo com pesos não negativos. Ele garante que, ao final de sua execução, todas as distâncias dos nós de origem aos demais nós são mínimas. No entanto, é importante observar que o algoritmo pode não funcionar corretamente em grafos com pesos negativos, pois poderia entrar em um ciclo infinito ao tentar melhorar as distâncias.

Sua aplicação é ampla e essencial em diversos domínios, tornando-se uma ferramenta valiosa para otimização de rotas, planejamento logístico e outras tarefas que requerem a determinação do caminho mais eficiente entre dois pontos em um grafo ponderado.

O algoritmo de Dijkstra é amplamente utilizado e possui diversas aplicações em sistemas de transporte, redes de computadores, jogos, entre outros cenários onde a otimização de caminhos é fundamental.

Ao entender o funcionamento detalhado do algoritmo de Dijkstra, é possível apreciar suas vantagens e limitações, bem como a importância de garantir que o grafo em questão atenda aos requisitos de pesos não negativos para obter resultados precisos e confiáveis.

5. Referências

- Edsger W. Dijkstra. "A note on two problems in connexion with graphs." *Numerische Mathematik*, 1(1), 269–271, 1959.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.