1 Caminho mínimo - Algoritmo de Dijskstra

1.1 Introdução

Dado dois vértices nesta rede, queremos determinar o menor caminho entre eles.

Uma primeira questão é como representar os valores associados às arestas neste grafo. Isto pode ser feito através da matriz de pesos.

1.1.1 Definição

Seja D um digrafo simples cujas arestas possuem "pesos" associados, digamos, a cada aresta (v_i, v_j) está associado um número real $w_{ij} \geq 0$ (que pode representar comprimento, distância, valor, etc).

Vamos definir $w_{ii} = 0$ para todo i e $w_{ij} = \infty$ quando não existe uma aresta entre os vértices v_i e v_j.

Assim a **matriz de pesos** é uma matriz $n \times n$ definida como $W = [\, w_{ij}]$, onde n é o número de vértices.

1.1.2 Observação

Em geral

- $w_{ij} \neq w_{ji}$ e
- $w_{ij} + w_{jk}$ pode ser menor ou igual a w_{ik}.

1.2 Ideia do Algoritmo de Dijkstra

Vamos supor que queremos encontrar o caminho mínimo entre os nós s e t em uma rede dada. A ideia consiste em:

- Rotular os vértices do digrafo.
- A partir do vértice inicial s, proceder em direção ao vértice final t (seguindo as arestas orientadas) rotulando os vértices com as suas distâncias ao vértice s, medidas até aquele momento.
- A cada estágio do algoritmo teremos vértices que possuem rótulos temporários e vértice com rótulos permanentes.
- O rótulo de um vértice v_j é feito permanente quando este rótulo representa a menor distância de s até v_j.
- Começamos associando rótulo permanente igual a zero para o vértice s, e um rótulo temporário igual a ∞ para os outros $n-1$ vértices do grafo.
- A cada iteração, um novo vértice recebe um rótulo permanente de acordo com as seguintes regras:
  1. Cada vértice v_j com um rótulo temporário, recebe um novo rótulo temporário dado por:

$$min\{\text{rótulo de } v_j,\ (\text{rótulo de } v_i)\ + w_{ij}\},$$

onde v_i é o vértice que recebeu rótulo permanente da iteração anterior e w_{ij} é o valor da aresta entre o vértice v_i e o vértice v_j.

2. Encontre o menor valor entre os rótulos temporários. Este será o rótulo permanente do respectivo vértice. Em caso de empate selecione qualquer um dos candidatos e atribua rótulo permanente ao escolhido.

- Repetir 1 e 2 até que o vértice destino, t, receba um rótulo permanente.

1.3 Como recuperar o caminho?

A partir do vértice destino t, verificamos o vértice com rótulo permanente usado na obstenção do rótulo de t.

1.4 Implementação

Em uma possível implementação deste algoritmo vamos precisar armazenar as seguintes informações:

- Indicação se um vértice v_k possui rótulo permanente ou temporário.
- Guardar a menor distância entre o vértice inicial s e o vértice v_k.
- Guardar o vértice com rótulo permanente que deu origem a um novo rótulo (importante para recuperar o caminho).

Assim vamos podemos definir a seguinte estrutura de dados:

1. Entrada de dados: matriz de pesos W ($O(n^2)$).
2. A dificuldade é distinguir, a cada iteração, os vértices com rótulos permanentes e os vértices com rótulo temporário. Utilizamos um vetor lógico (ou binário) n-direcional:

$$final(v) = \begin{cases} True & \text{se o rótulo do vértice } v \text{ é permanente,} \\ False & \text{se o rótulo do vértice } v \text{ é temporário.} \end{cases}$$

3. Precisamos também de um vetor n-direcional para guardar as distâncias acumuladas do vértice inicial s aos outros vértices v_i. Vamos chamar este vetor de $dist$.
4. Como recuperar o caminho? Sabemos que a menor distância será dada por $dist(t)$. Mas qual é este caminho? Cada vez que o rótulo de um vértice é modificado precisamos saber a partir de que vértice foi calculado o novo rótulo. Matendo um vetor n-direcional $pred$ tal que
   - $pred(v)$ indica o vértice com rótulo permanente que deu origem ao rótulo do vértice v,
   - e se v for o vértice inicial, então $pred(s) = -1$; temos que o menor caminho é dado por:

$$s, pred(pred(\ldots)), \ldots, pred(pred(t)), pred(t), t.$$

Considere um digrafo $D(V, A)$, com n vértices e sua matriz de pesos $W = [w_{ij}], n \times n$.

Queremos encontrar o menor caminho entre o vértice s e o vértice t no digrafo D.

Defina os vetores:

- $final(i)$ - indica se o vértice v_i recebeu rótulo permanente (potencial) ou não;
- $dist(i)$ - indica a distância acumulada do vértice inicial s até o vértice v_i;
- $pred(i)$ - indica o vértice com rótulo permanente que deu origem ao rótulo do vértice v_i.

1.5 Observações

- A complexidade em termos de tempo computacional do algoritmo é dado por:

O laço principal deste algoritmo, no pior caso, é executado $n-1$ vezes. Isto acontece quando o vértice final, t, é o último a receber um rótulo permanente. Para cada execução deste laço, precisamos examinar uma linha da matriz de pesos, e atualizar os vetores $dist$ e $pred$, ou seja, um tempo proporcional a n. Assim o tempo total é da ordem $2n(n-1)$. A complexidade é $O(n^2)$.

Observe que independente do número de arestas no grafo.

- É possível, usando o algoritmo de Dijkstra encontrar o menor caminho entre o vértice s e todos os outros vértices do grafo. O que deve ser modificado?
- O algoritmo de Dijkstra funciona apenas se $w_{ij} \geq 0$ para todos i, j.