Tópicos Avançados em Algoritmos

Prof. Hamilton José Brumatto

Bacharelado em Ciência da Computação - UESC

23 de maio de 2019

Caminhos Mínimos de Todos os Pares

O Problema

- Quando tratamos o problema de caminhos mínimos, consideramos algoritmos que define os caminhos mínimos de uma única origem.
- Dado um grafo qualquer, se quisermos saber o caminho mínimo entre dois pares: origem, e destino, aplicamos um algoritmo para o caminho mínimo a partir da origem e encontramos o resultado.
- Se quisermos buscar esta informação para vários pares, então teremos de realizar o algoritmo a cada par consultado.'

O Problema

- O custo para o algoritmo de Djikstra para uma origem é: $O((V+E)\log V)$.
- Se considerarmos todos os pares, precisamos realizar esta ação V vezes, isto se guardarmos o resultado numa tabela, caso contrário iremos repetir para a mesma origem mais de uma vez.
- Em melhor tempo, falamos de $O(V^2+EV)\log V)$ lembrando que E pode ser $O(V^2)$, ou seja: $O(V^3)\log V)$.
- Temos um algoritmo mais organizado que pode construir a tabela em $O(V^3)$.

O Algoritmo de Floyd-Wharshall

- Este algoritmo é baseado em programação dinâmica.
- Para o algoritmo, dado um caminho simples: $p = \langle v_1, v_2, \ldots, v_l \rangle$, um vértice *intermediário* é qualquer vértice diferente de v_1 e v_l.
- Um vértice intermediário é um vértice no conjunto $\{v_2, \ldots, v_{l-1}\}$

O Algoritmo de Floyd-Wharshall

- Sejam $V = \{1, 2, \ldots, n\}$ os vértices de G.
- Considere um subconjunto $\{1, 2, \ldots, k\}$ de vértices para algum k.
- Para qualquer par de vértices $i, j \in V$, considere todos o caminhos desde i até j cujos vértices intermediários são todos traçados a partir de $\{1, 2, \ldots, k\}$.
- Seja p um caminho simples de peso mínimo entre eles (i, j).
- O relacionamento depende do fato de k ser ou não um vértice intermediário de p.

O Algoritmo de Floyd-Wharshall

- Se k não é um vértice intermediaário de p, então todos os vértices intermediários estão no conjunto $\{1, 2, \ldots, k-1\}$. O caminho mais curto de i até j com todos os vértices intermediários no conjunto $\{1, 2, \ldots, k-1\}$ também é o caminho mais curto com todos os vértices intermediários no conjunto $\{1, 2, \ldots, k\}$.
- Se k é um vértice intermediário de p, então desmembramos p em $\langle i, \ldots, k \rangle$ e $\langle k, \ldots, j \rangle$. Então p_1 é o caminho mais curto de i, k com todos vértices intermediários de $\{1, \ldots, k-1\}$ (já que k não é um vértice intermediário, e p_2 é o caminho mais curto de k, j com todos vértices intermediários de $\{1, \ldots, k-1\}$

O Algoritmo de Floyd-Wharshall

- Seja $d_{ij}^{(k)}$ o peso de um caminho mais curto desde o vértice i até o vértice j para o qual todos os vértices intermediários estão no conjunto $\{1, 2, \ldots, k\}$. Quando $k = 0$, um caminho desde o vértice i até o vértice j sem vértices intermediários com numeração mais alta que $k = 0$ não tem absolutamente nenhum vértice intermediário. Tal caminho tem apenas uma aresta, logo $d_{ij}^{(0)} = \omega_{ij}$.
- A solução recursiva:

$$d_{ij}^{(k)} = \begin{cases} \omega_{ij} & \text{se } k = 0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{se } k \geqslant 1 \end{cases}$$

- $d_{ij}^{(n)}$ fornece a resposta final para p_{ij}.
- A matriz $D^n = (d_{ij}^{(n)})$ contém os caminhos mínimos de todos os pares $delta(i,j)$

O Algoritmo de Floyd-Wharshall

- A solução dinâmica (bottom-up): Seja $W n \times n$ uma matriz de incidência onde $W_{i,j} = \omega_{i,j}$ o peso do arco que liga o vértice i ao vértice j (∞ se não houver tal arco). O algoritmo de Floyd-Wharshall pode ser escrito como:

Algoritmo $\textsc{Floyd-Wharshall}(W)$
  $n \leftarrow linhas(W)$
  $D \leftarrow W$
  para $k \leftarrow 1$ **até** n **faça**
    para $i \leftarrow 1$ **até** n **faça**
      para $j \leftarrow 1$ **até** n **faça**
        $d_{ij} \leftarrow \min(d_{ij}, d_{ik} + d_{kj})$
  retorne D

URI-1539: Empresa de Telecom

Cesário é um analista da Algar Telecom, e está trabalhando em um projeto de análise da rede de telefonia móvel. Ele terá que desenvolver um sistema que analise o alcance de cada uma das antenas dessa rede, e que defina os custos operacionais para o envio de dados de de dispositivo para outro, baseando-se na distancia entre as antenas. O objetivo minimizar esses custos, encontrando a melhor rota disponível. Os cálculos também visam descobrir se é possível estabelecer um caminho entre dois dispositivos, de forma a detectar graves problemas na rede.

Mesmo com todos os dados disponíveis para processamento, Cesário tem enfrentado problemas na implementação devido a alta complexidade desse algoritmo, por isso você foi contratado para ajudá-lo. O seu objetivo é analisar todas as antenas da rede da Algar Telecom, observando as suas coordenadas e raios de alcance; verificar quais as antenas possíveis de serem acessadas (dentro do raio de alcance); e calcular o menor caminho entre duas antenas determinadas.

URI-1539: Entrada/Saída

Entrada:
A entrada é composta de vários casos de testes. Sendo que, a primeira linha contém um inteiro não negativo, **N** ($2 \leqslant$**N**$\leqslant 100$), que indica o número de antenas disponíveis para interconexão na rede. Seguem-se **N** linhas, cada uma contendo três números inteiros **X** ($0 \leqslant$**X**$\leqslant 1000$), **Y** ($0 \leqslant$**Y**$\leqslant 1000$) e **R** ($1 \leqslant$**R**$\leqslant 1000$), que descrevem a posição da antena, coordenadas **X** e **Y**, e o seu raio de alcance **R** (separados por espaço em branco). A linha seguinte contém outro inteiro não negativo, **C** ($1 \leqslant$**C**$\leqslant 100$), que descreve a quantidade de cálculos à serem realizados nessa rede. As **C** linhas seguintes contém 2 inteiros cada, $\mathbf{A}_1$ ($1 \leqslant \mathbf{A}_1 \leqslant \mathbf{N}$) e $\mathbf{A}_2$ ($1 \leqslant \mathbf{A}_2 \leqslant \mathbf{N}$), que descrevem o índice das antenas a serem utilizadas e também separadas por espaço em branco.
O fim das entradas é sinalizado por um número 0.

Saída:
Para cada caso de teste, deve-se imprimir **C** linhas, sendo que cada uma representa a distância do menor caminho entre as duas antenas. Os valores devem ser INTEIROS, ou seja, a parte real deve ser truncada (não arredondada), e sempre com uma quebra de linha. Caso não seja identificada uma rota entre as antenas, deve ser impresso o valor -1.

URI - 1539 - Abordagem

- São 4 etapas:
  1. Construímos o grafo, com a informação para cada nó do ID do vértice, posição X,Y e alcance R.
  2. Construímos a matriz de distâncias entre cada vértice, colocando -1 se não for alcançável (∞)
  3. Aplicamos o algoritmo de Floyd-Wharshall na matriz de distâncias.
  4. Para cada par de vértices, imprimimos a distância.

Etapa 1: construir o grafo

```
class no {
public:
 int id, x, y, r;
 no() {id = -1; x=-1; y=-1; r=-1;}
 no(int _id, int _x, int _y, int _r):  id(_id), x(_x), y(_y), r(_r) { }
};

int main() {
 int n, c, x, y, r;
 double d[100][100];
 no g[100];

 cin >> n;
 while(n) {
  for(int i=0; i<n; i++) {
   cin >> x >> y >> r;
   g[i] = no(i,x,y,r);
  }
```

Etapa 2: construir a matriz de distâncias

```
for(int i=0; i<n; i++) // todo mundo com ∞
 for(int j=0; j<n; j++)
  d[i][j]=-1;

double dist;
for(int i=0; i<n; i++)
 for(int j=0; j<n; j++) {
  if(i==j) d[i][j]=0;
  else {
   dist = sqrt((g[i].x-g[j].x)*(g[i].x-g[j].x)+
                (g[i].y-g[j].y)*(g[i].y-g[j].y));
   if(dist <= g[i].r)
   d[i][j] = dist;
  }
 }
```

Etapa 3: Floyd-Wharshall

```
for(int k=0; k<n; k++) {
 for(int i=0; i<n; i++)
  for(int j=0; j<n; j++)
   if(!(d[i][k]<0) && !(d[k][j]<0)) { //existe caminho por k
    if(d[i][j]<0) d[i][j] = d[i][k]+d[k][j]; // d_ij = ∞
    else if(d[i][j] > d[i][k]+d[k][j]) d[i][j] = d[i][k]+d[k][j];
  }
}
```

Etapa 4: Queries e prints!

```
  cin >> c;
  while(c--) {
   int i, j;
   cin >> i >> j; i--; j--;
   cout << (int) d[i][j] << endl;
  }
  cin >> n;
 }
 return 0;
}
```