Caminhos Mínimos: Todos para Todos

prof. Fábio Luiz Usberti
prof. Cid Carvalho de Souza

MC521 - Desafios de Programação I

Instituto de Computação - UNICAMP - 2018

Sumário

Caminhos Mínimos de Todos para Todos (APSP)

- Caminhos Mínimos de Todos para Todos ("All-pairs shortest paths" – APSP): Dado um grafo G, com comprimento c_{ij} nas arestas, encontre os caminhos de mínima distância para todos os pares de nós $i, j \in V$.
- Exemplo (UVa 11463 - Commandos): Considere um grafo conexo $G(V, E)$, $|V| \leqslant 100$, com comprimentos não-negativos nas arestas. Dados dois vértices $s, d \in V$, encontre o maior valor possível para $dist[s][i] + dist[i][d]$ considerando $\forall i \in V$, dado que $dist[u][v]$ é a distância de caminho mínimo de u para v.
- O problema acima pode ser resolvido obtendo-se os caminhos mínimos para todos os pares de nós em G.

Caminhos Mínimos de Todos para Todos (APSP)

- Os algoritmos de Dijkstra e Bellman-Ford podem resolver o APSP executando-os uma vez para cada nó (origem). Nesse caso, suas complexidades passam a ser:
  1. Dijkstra: $O(|V|) \cdot O((|V| + |E|) \log |V|) = O(|V||E| \log |V|)$
  2. Bellman-Ford: $O(|V|) \cdot O(|V||E|) = O(|V|^2|E|)$
- O algoritmo de Floyd-Warshall para o APSP foi proposto por Robert W. Floyd e Stephen Warshall em 1962.
- Trata-se de um algoritmo de programação dinâmica cuja complexidade assintótica é $O(|V|^3)$, uma vez que o código é composto por três laços encadeados de nós.
- Em virtude de seu custo computacional, Floyd-Warshall só pode ser aplicado em competições para grafos com $|V| \leqslant 400$.

Subestrutura ótima

- Seja D_{ij}^k a distância de caminho mínimo do vértice i para o vértice j que utiliza somente os vértices do conjunto $[0..k]$ como vértices intermediários do caminho.
- A subestrutura ótima adotada pelo algoritmo Floyd-Warshall pode ser expressa pela seguinte função recursiva:

$$D_{ij}^k = \begin{cases} c_{ij} & k < 0 \\ \min(D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}) & \text{c.c.} \end{cases}$$

- Cabe notar que o caso base ($k < 0$) implica em não utilizar nenhum nó intermediário.

Descrição

- A tabela de PD é preenchida da forma bottom-up, ou seja, camada a camada, iniciando pelo caso base ($k = -1$).
- As camadas seguintes ($k = 0, \ldots, |V| - 1$) são preenchidas a partir da definição do caso geral.
- Cabe observar que, para preencher uma camada k, são necessárias apenas as distâncias da camada anterior $k - 1$. Isso possibilita a economia de memória (de $O(|V|^3)$ para $O(|V|^2)$).
- Assim como no algoritmo de Bellman-Ford, é possível economizar ainda mais memória sobrescrevendo as distâncias da camada k diretamente sobre a camada $k - 1$. Nesse caso, a subestrutura ótima pode ser expressa como:

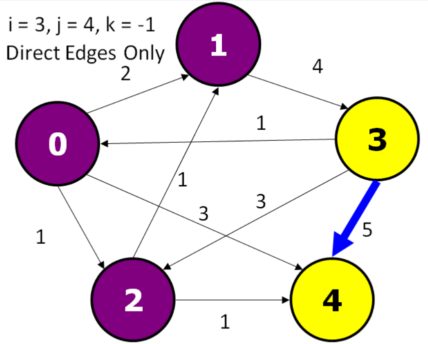
$$D_{ij} = \begin{cases} c_{ij} & \text{(inicialização)} \\ \min(D_{ij}, D_{ik} + D_{kj}) & \text{iteração } 0 \leqslant k \leqslant |V| - 1 \end{cases}$$

Código-fonte

- Uma implementação do algoritmo Floyd-Warshall é apresentada a seguir.

```
int main() {
  int V, E, u, v, w, dist[NODES][NODES];

  scanf("%d %d", &V, &E);
  for (int i = 0; i < V; i++) {
    for (int j = 0; j < V; j++)
      dist[i][j] = INF;
    dist[i][i] = 0;
  }

  for (int i = 0; i < E; i++) {
    scanf("%d %d %d", &u, &v, &w);
    dist[u][v] = w; // directed graph
  }

  // Floyd-Warshall algorithm
  for (int k = 0; k < V; k++)
    for (int i = 0; i < V; i++)
      for (int j = 0; j < V; j++)
        dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);

  return 0;
}
```

Exemplo



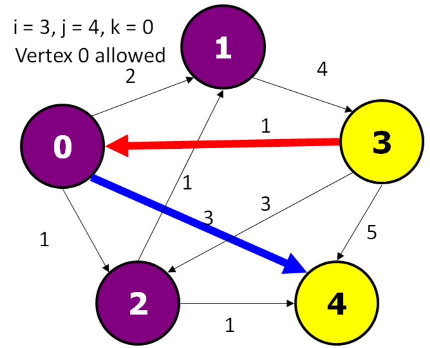
sp(3, 2, -1) = **3** sp(2, 4, -1) = **1** sp(3, 4, -1) = **5**

We will monitor these two values

The current content of Adjacency Matrix D at **k = -1**

k = -1	0	1	2	3	4
0	0	2	1	∞	3
1	∞	0	∞	4	∞
2	∞	1	0	∞	1
3	1	∞	3	0	**5**
4	∞	∞	∞	∞	0

Exemplo

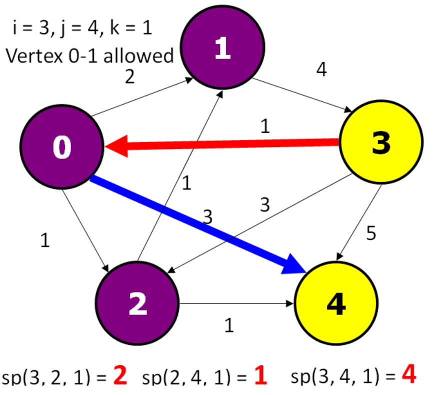


sp(3, 2, 0) = **2** sp(2, 4, 0) = **1** sp(3, 4, 0) = **4**

The current content of Adjacency Matrix D at **k = 0**

k = 0	**0**	**1**	**2**	**3**	**4**
0	0	2	1	∞	3
1	∞	0	∞	4	∞
2	∞	1	0	∞	1
3	1	**3**	**2**	0	**4**
4	∞	∞	∞	∞	0

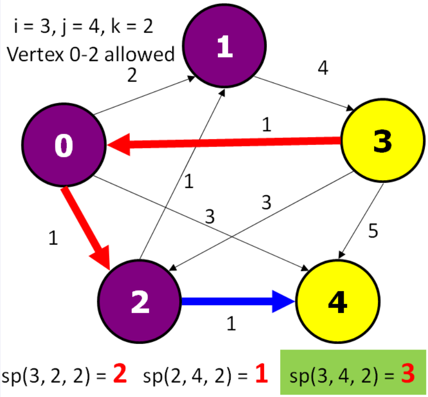
Exemplo



The current content of Adjacency Matrix D at **k = 1**

k = 1	**0**	**1**	**2**	**3**	**4**
0	0	2	1	**6**	3
1	∞	0	∞	4	∞
2	∞	1	0	**5**	1
3	1	3	2	0	**4**
4	∞	∞	∞	∞	0

Exemplo



The current content of Adjacency Matrix D
at **k = 2**

k = 2	**0**	**1**	**2**	**3**	**4**
0	0	2	1	6	**2**
1	∞	0	∞	4	∞
2	∞	1	0	5	1
3	1	3	2	0	**3**
4	∞	∞	∞	∞	0

Floyd-Warshall

Exemplo

k = 1	0	1	2	3	4
0	0	2	1	6	**3**
1	∞	0	∞	4	∞
2	∞	1	0	5	**1**
3	1	3	**2**	0	**4**
4	∞	∞	∞	∞	0

k=1

k = 2	0	1	2	3	4
0	0	2	1	6	**2**
1	∞	0	∞	4	∞
2	∞	1	0	5	1
3	1	3	2	0	**3**
4	∞	∞	∞	∞	0

k=2

Exibindo Solução

- Para imprimir os caminhos correspondentes à solução do algoritmo de Floyd-Warshall, é possível utilizar a matriz *pred*[*i*][*j*] que armazena o nó antecessor de *j* no caminho mínimo de *i* para *j*.

```
// initialize the parent matrix
for (int i = 0; i < V; i++)
   for (int j = 0; j < V; j++)
      pred[i][j] = i;

// Floyd-Warshall algorithm
for (int k = 0; k < V; k++)
   for (int i = 0; i < V; i++)
      for (int j = 0; j < V; j++)
         if (dist[i][k] + dist[k][j] < dist[i][j]) {
            dist[i][j] = dist[i][k] + dist[k][j];
            pred[i][j] = pred[k][j];     // update the pred matrix
         }
//------------------------------------------------------------------------
// when we need to print the shortest paths, we can call the method below:
void printPath(int i, int j) {
   if (i != j) printPath(i, pred[i][j]);
      printf(" %d", j);
}
```

Aplicações

Algoritmo Floyd-Warshall

- O principal propósito do algoritmo de Floyd-Warshall consiste em resolver o problema de caminhos mínimos de todos para todos.
- Existem outros problemas que podem ser resolvidos por esse algoritmo, contanto que a dimensão do grafo seja compatível.
- Outras aplicações para o algoritmo são:

1. Fecho transitivo
2. Caminho minimax
3. Ciclo de custo mínimo
4. Diâmetro do grafo
5. Componentes fortemente conexos

Aplicações

- O fecho transitivo de um grafo consiste em determinar, para todos os pares ordenados de nós ij, se i está conectado a j. Em outras palavras, encontrar o fecho transitivo consiste em determinar uma matriz de conectividade M, tal que $m_{ij} = 1$ se existe um caminho de i para j; $m_{ij} = 0$, caso contrário.
- Para este problema, é possível substituir as operações aritméticas do algoritmo de Floyd-Warshall por operações em bits sobre os elementos da matriz de adjacência do grafo, como mostra o código a seguir:

```
for (int k = 0; k < V; k++)
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      AdjMat[i][j] |= (AdjMat[i][k] & AdjMat[k][j]);
```

Aplicações

- O problema do caminho minimax consiste em encontrar um caminho de um nó i até um nó j cuja aresta de maior comprimento no caminho seja mínima. Em outras palavras, a distância de um caminho é definida pelo comprimento de sua maior aresta.
- Para este problema, mantém-se a inicialização normal do algoritmo de Floyd-Warshall, ou seja, $dist[i][j] = c_{ij}$, se a aresta ij existir e $dist[i][j] = \infty$, caso contrário.
- A alteração do algoritmo ocorre no caso geral da função recursiva.

Caminho Minimax

Aplicações

- Seja D_{ij}^{k} a distância do caminho minimax do nó i ao nó j, utilizando como (possíveis) nós intermediários $[0..k]$.
- A função recursiva pode ser expressa como: $D_{ij}^{k} = \min(D_{ij}^{k-1}, \max(D_{ik}^{k-1}, D_{kj}^{k-1}))$

```
for (int k = 0; k < V; k++)
  for (int i = 0; i < V; i++)
    for (int j = 0; j < V; j++)
      dist[i][j] = min(dist[i][j], max(dist[i][k], dist[k][j]));
```

Ciclo de Custo Mínimo

Aplicações

- O algoritmo de Floyd-Warshall, assim como o algoritmo de Bellman-Ford, consegue detectar a presença de ciclos negativos no grafo.
- Além disso, Floyd-Warshall também consegue detectar o ciclo mais barato (ou espessura) de um grafo, seja ele negativo ou positivo.
- Para isso, inicia-se a diagonal da matriz de PD com valores infinitos ($dist[i][i] = INF$).
- Em seguida, executa-se o algoritmo normalmente. Se ao final da execução $dist[i][i] < INF$ então $dist[i][i]$ terá o valor do custo do ciclo mínimo que contém o nó i.
- Para determinar o custo do ciclo mais barato no grafo, se houver, basta imprimir o menor valor armazenado na diagonal da matriz.

Aplicações

- O diâmetro do grafo consiste em encontrar o maior caminho mínimo do grafo.
- O problema UVa 1056 - Degrees of Separation, que fez parte da lista de problemas da final mundial do ICPC em 2006, consiste em determinar o diâmetro do grafo.
- Esse problema pode ser resolvido determinando-se os caminhos mínimos para todos os pares de nós (Floyd-Warshall). Em seguida, avalia-se todos os elementos da matriz em busca do caminho mais caro (menor do que infinito).

Componentes Fortemente Conexos

Aplicações

- Um componente fortemente conexo (SCC) de um grafo orientado consiste em um subgrafo orientado tal que, para qualquer par de vértices u e v, é possível encontrar um caminho de u para v e um caminho de v para u.
- Um algoritmo assintoticamente ótimo para encontrar componentes fortemente conexos consiste no algoritmo de Tarjan, com complexidade $O(|V| + |E|)$.
- Como o algoritmo de Tarjan é relativamente extenso, em competição pode ser preferível encontrar SCCs a partir do algoritmo de Floyd-Warshall, contanto que as dimensões dos grafos sejam compatíveis ($|V| \leqslant 400$).
- Para isso, primeiramente determina-se o fecho transitivo do grafo. Em seguida, dado o componente conexo de um nó i, determina-se os demais nós $j \in V$ desse componente conexo verificando a condição `AdjMat[i][j] && AdjMat[j][i]`.

Referências

1. S. Halim e F. Halim. Competitive Programming 2, Second Edition Lulu (www.lulu.com), 2011. (IMECC – 005.1 H139c)
2. S. S. Skiena, M. A. Revilla. Programming Challenges: The Programming Contest Training Manual, Springer, 2003.
3. T.H. Cormen, C.E. Leiserson, R.L.Rivest e C. Stein. Introduction to Algorithms. 2nd Edition, McGraw-Hill, 2001. (no. chamada IMECC – 005.133 Ar64j 3.ed.)
4. U. Manber. Introduction to Algorithms: A Creative Approach. Addison-Wesley. 1989. (no. chamada IMECC – 005.133 Ec53t 2.ed.)