Chapter 6: Shortest Path Problem

Discrete Mathematics 2

Lecturer: Nguyen Kieu Linh
Posts and Telecommunications Institute of Technology

Hanoi, 2024
http://www.ptit.edu.vn

Contents

Problem Statement

Length of the path

* Consider a graph $G = \langle V, E \rangle$ with the set of vertices V and the set of edges E.
* For each edge $(u, v) \in E$, we set a real value $a(u, v)$ called weight of the edge, $a(u, v) = \infty$ if $(u, v) \notin E$
* If $v_0, v_1, \ldots, v_k$ is a path of G, $\sum_{i=1}^{k} a(v_{i-1}, v_i)$ is said to be the length of the path

General problem

* Find the shortest (length) path from a vertex $s \in V$ (source vertex) to a vertex $t \in V$ (target vertex)?
* Such path is called the shortest path from s to t, the length of the path $d(s, t)$ is called the shortest distance from s to t
* If does not exist a path from s to t, length of the path $d(s, t) = \infty$

Problem Statement

Find the shortest paths from vertex s to the other ones?

* For graphs with non-negative weights, we can find a solution by using Dijkstra algorithm.
* For graphs with negative weights but do not have negative circuits, we can find a solution by using the Bellman-Ford algorithm.
* For graphs with negative circuits, the problem does not have any solution.

Find the shortest paths between two every vertices

* For graphs with non-negative weights, we can find a solution by applying the Dijkstra algorithm n times.
* For graphs with non-negative circuits, we can find a solution by using the Floyd algorithm.

Contents

Dijkstra algorithm

Purpose

* To find the shortest paths from a vertex s to the other ones.
* Applicable to directed graphs with non-negative weights Idea.

Idea

* Assign a temporary label to each vertex.
* Labels will be re-assign in a loop: In each loop, we will fix the label for one vertex (the label is the shortest distance from s to that vertex).

Dijkstra algorithm

Dijkstra $(s)\{$

Step 1 (Initialize):

$d[s] = 0$; //Assign label 0 to s

$T = V \setminus \{s\}$; // T is the set of vertices with a temporary label

for $(v \in V)\{$ //Using s to assign label to other vertices

$d[v] = a(s, v)$;

$pre[v] = s$;

$\}$

Step 2 (Loop):

while $(T \neq \emptyset)\{$

Find a vertex $u \in T$ such that $d[u] = \min\{\, d[z] \mid z \in T\}$;

$T = T \setminus \{u\}$; //fix the label of u

for $(v \in T)\{$ //Using u to re-assign label to other vertices

if $(d[v] > d[u] + a(u, v))\{$

$d[v] = d[u] + a(u, v)$; //Re-assign label to v;

$pre[v] = u$;

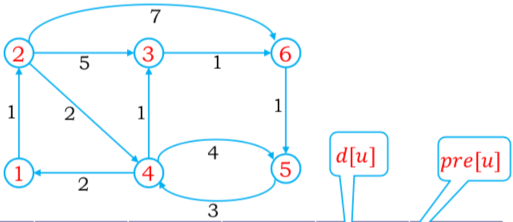
$\}$

$\}$

$\}$

Dijkstra algorithm

Apply Dijkstra algorithm to find the shortest paths from vertex 1 to other vertices of the graph.



Loop	Vertex 1	Vertex 2	Vertex 3	Vertex 4	Vertex 5	Vertex 6
Initialize	0, 1	1, 1 *	∞, 1	∞, 1	∞, 1	∞, 1
1	-	-	6, 2	3, 2 *	∞, 1	8, 2
2	-	-	4, 4 *	-	7, 4	8, 2
3	-	-	-	-	7, 4	5, 3 *
4	-	-	-	-	6, 6 *	-
5						

Contents

Bellman-Ford algorithm

Purpose

- To find the shortest paths from a vertex s to the other ones.
- Applicable to directed graphs without negative circuits (may have negative weights)

Idea

- Assign a temporary label to each vertex.
- Labels will be re-assign in a loop.

Bellman-Ford algorithm

Purpose

* To find the shortest paths from a vertex s to the other ones.
* Applicable to directed graphs without negative circuits (may have negative weights)

Idea

* Assign a temporary label to each vertex.
* Labels will be re-assign in a loop.

Bellman-Ford algorithm

Bellman-Ford$(s)\{$

Step 1 (Initialize):

for $(v \in V)\{$

$d[v] = a(s, v);$

$pre[v] = s;$

$\}$

Step 2 (Loop):

$d[s] = 0;$

for $(k = 1; k \leq n - 1; k++)\{$

for $(v \in V \setminus \{s\})\{$

for $(u \in V)\{$

if $(d[v] > d[u] + a(u, v))\{$

$d[v] = d[u] + a(u, v);$

$truoc[v] = u;$

$\}$

$\}$

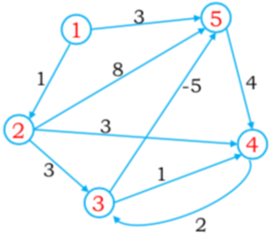
$\}$

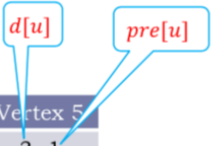
$\}$

$\}$

Bellman-Ford algorithm

Apply Bellman-Ford algorithm to find the shortest paths from vertex 1 to other vertices of the graph.





Loop	Vertex 1	Vertex 2	Vertex 3	Vertex 4	Vertex 5
Initialize	0, 1	1, 1	∞, 1	∞, 1	3, 1
k=1	0, 1	1, 1	4, 2	4, 2	-1, 3
2	0, 1	1, 1	4, 2	3, 5	-1, 3
3	0, 1	1, 1	4, 2	3, 5	-1, 3

Unchanged

Contents

Floyd algorithm

Purpose

* To find the shortest paths between two every vertices of the graph.
* Applicable to directed graphs without negative circuits (may have negative weights).

Idea

* Use a loop procedure.
* Consider each vertex u, for every path (between two arbitrary vertices), if the length of this path is greater than the length of the path through vertex u, we update this path

Floyd algorithm

```
Floyd( ){
    Step 1 (Initialize):
    for (i = 1, i ≤ n; i++){
        for( j = 1, j ≤ n; j++){
            d[i,j] = a(i,j);
            if (a(i,j)! = ∞) next[i,j] = j;
            else next[i,j] = null;
        }
    }
    Step 2 (Loop):
    for (k = 1, k ≤ n; k++){
        for (i = 1, i ≤ n; i++){
            for (j = 1, j ≤ n; j++){
                if (d[i,j] > d[i,k] + d[k,j]){
                    d[i,j] = d[i,k] + d[k,j];
                    next[i,j] = next[i,k];
                }
            }
        }
```

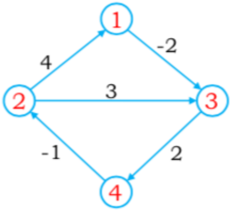
Floyd algorithm

Path recovery

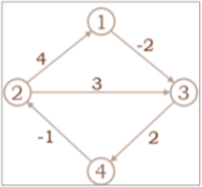
```
Reconstruct-Path( u, v){
        if ( next[u][v] == null)
                <Does not have path from u to v>;
        else{
                path = [u]; // path starts at u
                while(u ≠ v){
                        u = next[u][v];
                        path.append(u); //the next vertex in the path
                }

                return path;
        }
}
```

Floyd algorithm

Apply Floyd algorithm to find the shortest paths between two every vertices of the graph



$k = 0$		j			
		1	2	3	4
i	1	0, 1	∞, null	-2, 3	∞, null
	2	4, 1	0, 2	3, 3	∞, null
	3	∞, null	∞, null	0, 3	2, 4
	4	∞, null	-1, 2	∞, null	0, 4



d[i, j], next[i, j]

Đường đi ngắn nhất giữa một số cặp đỉnh:

1 → 2:
1 – 3 – 4 – 2 K/c: -1

$k = 1$		j			
		1	2	3	4
i	1	0, 1	∞, null	-2, 3	∞, null
	2	4, 1	0, 2	**2, 1**	∞, null
	3	∞, null	∞, null	0, 3	2, 4
	4	∞, null	-1, 2	∞, null	0, 4

$k = 2$		j			
		1	2	3	4
i	1	0, 1	∞, null	-2, 3	∞, null
	2	4, 1	0, 2	**2, 1**	∞, null
	3	∞, null	∞, null	0, 3	2, 4
	4	**3, 2**	-1, 2	**1, 2**	0, 4

1 → 3:
1 – 3
K/c: -2

1 → 4:
1 – 3 – 4
K/c: 0

$k = 3$		j			
		1	2	3	4
i	1	0, 1	∞, null	-2, 3	**0, 3**
	2	4, 1	0, 2	**2, 1**	**4, 1**
	3	∞, null	∞, null	0, 3	2, 4
	4	**3, 2**	-1, 2	**1, 2**	0, 4

$k = 4$		j			
		1	2	3	4
i	1	0, 1	**-1, 3**	-2, 3	**0, 3**
	2	4, 1	0, 2	**2, 1**	**4, 1**
	3	**5, 4**	**1, 4**	0, 3	2, 4
	4	**3, 2**	-1, 2	**1, 2**	0, 4

3 → 1:
3 – 4 – 2 – 1
K/c: 5

3 → 2:
3 – 4 – 2
K/c: 1

Summary

- Shortest path problem
- Dijkstra algorithm and its applications
- Bellman-Ford algorithm and its applications
- Floyd algorithm and its applications

Exercises

Exercise 1. Given a single graph $\mathrm{G} =< \mathrm{V}, \mathrm{E} >$ consisting of 7 vertices is represented as a weighted matrix as follows

$$\begin{bmatrix} 0 & 20 & 5 & 17 & \infty & \infty & \infty \\ 20 & 0 & \infty & 1 & \infty & \infty & 1 \\ 5 & \infty & 0 & 25 & 3 & 10 & \infty \\ 17 & 1 & 25 & 0 & 15 & \infty & \infty \\ \infty & \infty & 3 & 15 & 0 & 1 & \infty \\ \infty & \infty & 10 & \infty & 1 & 0 & 1 \\ \infty & 1 & \infty & \infty & \infty & 1 & 0 \end{bmatrix}$$

Apply Dijkstra's algorithm, find a shortest path from vertex 1 to vertex 7 of the given graph G, specifying the result at each step performed by the algorithm?

Exercise 2. Given a directed graph $G = < V, E >$ consisting of 6 vertices as shown in the figure below, the weights are written on each arc

$$\begin{bmatrix} 0 & 1 & \infty & 4 & \infty & \infty \\ \infty & 0 & 6 & 2 & \infty & \infty \\ \infty & \infty & 0 & \infty & 3 & 1 \\ \infty & \infty & 10 & 0 & 2 & \infty \\ \infty & \infty & \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

Apply Dijkstra's algorithm to find a shortest path from vertex 1 to the remaining vertices of graph G, indicating a shortest path from vertex 1 to vertex 6 .

Exercise 3. Given a single directed graph $G =< V, E >$ consisting of 6 vertices represented as a weighted matrix as follows

$$\begin{bmatrix} 0 & 1 & \infty & 4 & \infty & \infty \\ \infty & 0 & 6 & 2 & \infty & \infty \\ \infty & \infty & 0 & \infty & 3 & 1 \\ \infty & \infty & -3 & 0 & 2 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & -1 & 0 \end{bmatrix}$$

Apply the Bellman-Ford algorithm to find a shortest path from vertex 1 to the remaining vertices of the given graph G, indicating a shortest path from vertex 1 to vertex 6 .

Exercise 4. Given a single directed graph $G =< V, E >$ consisting of 6 vertices represented as a weighted matrix as follows

$$\begin{bmatrix} 0 & -2 & \infty & 5 & \infty & \infty \\ \infty & 0 & 1 & 2 & \infty & \infty \\ \infty & \infty & 0 & \infty & 1 & \infty \\ \infty & \infty & -2 & 0 & 2 & \infty \\ \infty & \infty & \infty & \infty & 0 & -1 \\ \infty & \infty & 2 & \infty & \infty & 0 \end{bmatrix}$$

Apply the Bellman-Ford algorithm to find a shortest path from vertex 1 to the remaining vertices of the given graph G, indicating a shortest path from vertex 1 to vertex 6 .

Exercise 5. Given a single directed graph $G = < V, E >$ consisting of 6 vertices represented as a weighted matrix as follows

$$\begin{vmatrix} 0 & 15 & 5 & 20 & \infty & \infty \\ 1 & 0 & \infty & 17 & 10 & \infty \\ \infty & \infty & 0 & 2 & \infty & 50 \\ 15 & 1 & \infty & 0 & \infty & 70 \\ 20 & 30 & \infty & 10 & 0 & 10 \\ \infty & 18 & \infty & 23 & 20 & 0 \end{vmatrix}$$

Applying Floyd's algorithm, find a shortest path between pairs of vertices (1, 2), (1, 3), (3, 4), (4, 2) of the given graph G, specifying the result at each step follow the algorithm?