International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 15, Issue 4 [April 2026] PP: 29-34

Using Dijkstra and Floyd Algorithms from Graph Theory to Solve Real-World Shortest Path Problems

Ho Thi Hong Lien
Vietnam–Korea University of Information and Communication Technology, The University of Danang

Abstract
This paper presents approaches for solving practical shortest path problems by modeling them as weighted graphs and applying classical graph theory algorithms, including Dijkstra's algorithm and the Floyd algorithm. Several real-world scenarios such as selecting economical travel routes, minimizing total travel cost, and determining graph centers are analyzed.
Keywords: *shortest path, Dijkstra algorithm, Floyd algorithm, graph theory.*

Date of Submission: 01-04-2026 Date of acceptance: 10-04-2026

I. Introduction

In daily life we frequently face situations requiring optimal route selection. These can be modeled as weighted graphs where intersections are vertices and road segments are edges with weights representing distances, times, or costs. This paper discusses modeling real-world routing problems and solving them using Dijkstra's and Floyd's algorithms.

II. Content

2.1. Theoretical Background

2.1.1. Shortest Path Between two Vertices

Problem statement

Given a weighted graph G = (V, E). Let w(i,j) be the weight of edge (i,j). The length of a path

$$\mu=v_0\rightarrow v_1\rightarrow v_2\rightarrow \ldots\rightarrow v_{n-1}\rightarrow v_n$$

is the sum of its edge weights:

$$L(\mu)=\sum_{i=1}^{n} w(v_{i-1},v_i)$$

Given two vertices a and z, the problem is to find the shortest path from a to z.

Dijkstra's Algorithm

Dijkstra's algorithm finds the shortest path from vertex a to vertex z in a weighted graph in which all weights w(i,j) > 0. Each vertex x is assigned a label L(x). When the algorithm terminates, L(z) is the length of the shortest path from a to z.

Input: A connected weighted graph G = (V, E) with w(i, j) > 0 for all edges (i, j), and vertices a, z.

Output: The shortest path length and the actual shortest path.

Method:

(1) Assign L(a) := 0. For all vertices x ≠ a, assign L(x) = ∞. Let T := V

(2) Select v ∈T with the smallest L(v). Set T := T – {v}

(3) If z ∉ T → stop.

L(z) is the shortest distance from aaa to z. By tracing backward from z through the recorded predecessors, we obtain the shortest path. Otherwise, proceed to Step 4.

(4) For each vertex x ∈ T adjacent to vvv, assign:

$$L(x) := \min\{L(x), L(v) + w(v, x)\}$$

If the value of L(x) changes, record vertex v as the predecessor of x for later reconstruction of the shortest path. Return to step 2.

The label-setting method of Dijkstra's algorithm

We construct a label table in which the columns correspond to the vertices and the rows correspond to the iterations of label updates in Step 4. The underlined labels indicate the smallest label selected in Step 2, and the removed vertex is recorded on the right.

After vertex *z* is removed, we trace back from *z* to *a* following the recorded labels in the table to obtain the shortest path.

2.1.2. Shortest Path Between All Pairs of Vertices

Problem Statement

Given a connected directed weighted graph G=(V,E). Let w(i,j) denote the weight of edge (i,j). The length of a path

$$\mu=v_0\rightarrow v_1\rightarrow v_2\rightarrow \ldots\rightarrow v_{n-1}\rightarrow v_n$$

is the sum of its edge weights

$$L(\mu)=\sum_{i=1}^{n} w(v_{i-1},v_i)$$

The problem is to determine the shortest path between every pair of vertices in the graph.

Floyd's Algorithm

This algorithm determines the shortest-path distances between all pairs of vertices in a connected directed weighted graph.

- **Input:** A connected graph G=(V,E), where V={1,2,…,n}, and each directed edge (i,j) has a positive weight w(i,j)>0.
- **Output:** A matrix D=[d(i,j)], where d(i,j) represents the length of the shortest path from vertex i to vertex j for all pairs (i,j).

Method:

(1) Initialization step: Let D_0 be the initial matrix $D_0 = [d_0(i,j)]$, where

$d_0(i,j) = w(i,j)$ if the edge (i,j) exists,

$d_0(i,j) = +\infty$ if the edge (i,j) does not exists.

(In particular, if there is no self-loop at vertex i, then $d_0(i,i) = +\infty$).

Set k:=0

(2) **Termination check:** If k = n, stop. The matrix $D = D_n$ is the matrix of shortest-path distances. Otherwise, set k:=k+1 and proceed to Step (3).

(3) **Compute matrix** D_k from D_{k-1}

For every pair (i,j), i=1..n, j=1..n, perform:

If $d_{k-1}(i,j) > d_{k-1}(i,k) + d_{k-1}(k,j)$ then set

$$d_k(i,j) := d_{k-1}(i,k) + d_{k-1}(k,j)$$

otherwise, set

$$d_k(i,j) := d_{k-1}(i,j)$$

Return to Step (2).

Floyd's algorithm can be applied to both directed and undirected graphs. For an undirected graph, each undirected edge (u,v) is replaced by two directed edges (u,v) and (v,u) with equal weights. However, in this case, the diagonal entries should be set to 0.

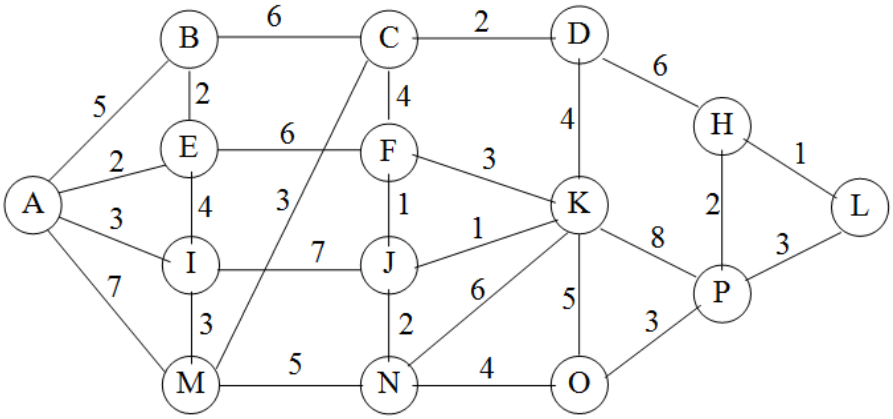
2.2. Modeling Several Graph Problems for Finding the Shortest Path

2.2.1. Selecting the Most Cost-Efficient Route

Problem 1:

The transportation network connecting tourist spots in an eco-tourism area can be modeled as a weighted graph, in which the weight of each edge represents either the distance or the time required to travel from one tourist spot to another.

The figure below shows 15 tourist spots labeled A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, and P. The edges connecting pairs of vertices indicate that there is a direct path between those tourist spots, and the numbers on the edges represent their corresponding lengths. Determine the most cost-efficient travel route from point A to all other points.



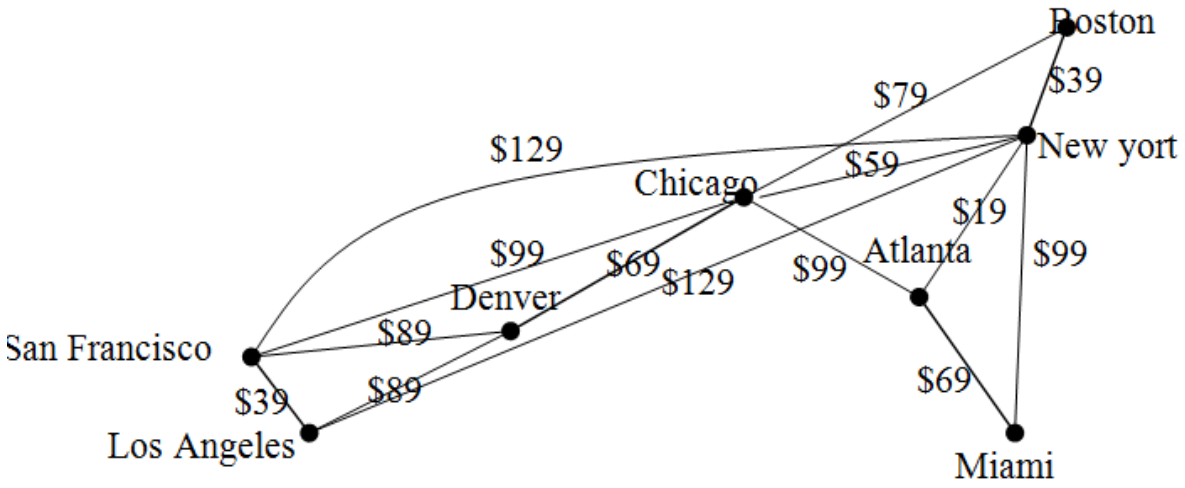
Applying Dijkstra's algorithm to find the shortest paths from vertex A to the remaining vertices in the graph, we obtain the following table::

L(A)	L(B)	L(C)	L(D)	L(E)	L(F)	L(H)	L(I)	L(J)	L(K)	L(L)	L(M)	L(N)	L(O)	L(P)
0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
–	5	∞	∞	**2**	∞	∞	3	∞	∞	∞	7	∞	∞	∞
–	4	∞	∞	–	8	∞	**3**	∞	∞	∞	7	∞	∞	∞
–	**4**	∞	∞	–	8	∞	–	10	∞	∞	6	∞	∞	∞
–	–	10	∞	–	8	∞	–	10	∞	∞	**6**	∞	∞	∞
–	–	9	∞	–	**8**	∞	–	10	∞	∞	–	11	∞	∞
–	–	9	∞	–	–	∞	–	**9**	11	∞	–	11	∞	∞
–	–	**9**	∞	–	–	∞	–	–	10	∞	–	11	∞	∞
–	–	–	11	–	–	∞	–	–	**10**	∞	–	11	∞	∞
–	–	–	11	–	–	∞	–	–	–	∞	–	**11**	13	18
–	–	–	**11**	–	–	∞	–	–	–	∞	–	–	13	18
–	–	–	–	–	–	14	–	–	–	∞	–	–	**13**	18
–	–	–	–	–	–	**14**	–	–	–	∞	–	–	–	16
–	–	–	–	–	–	–	–	–	–	**15**	–	–	–	16
–	–	–	–	–	–	–	–	–	–	–	–	–	–	**16**

From the table above, we can determine the shortest path and its length from point A to the other points. For example, the shortest path from A to P is: A → E → F → J → K → O → P, with a total length of 16.

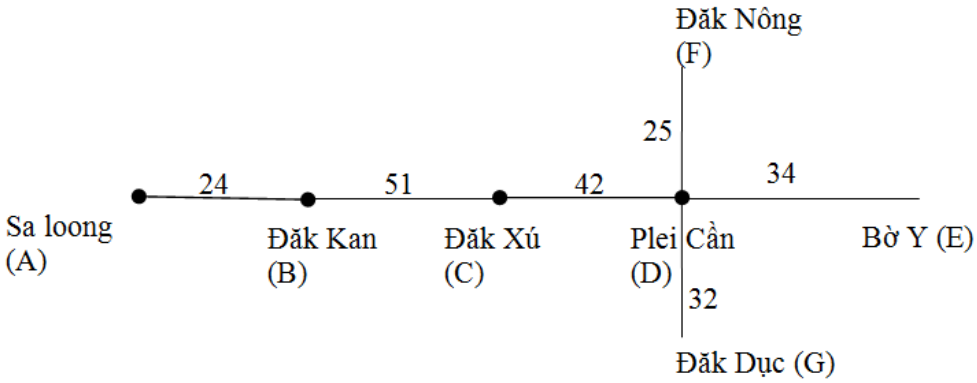
Problem 2: We need to model an airline network. Each city is represented by a vertex, and each flight is an edge connecting the corresponding vertices. If the problem requires considering the distance between cities, we assign to each edge a weight based on the distance between the corresponding cities. If we are concerned with the flight duration, we assign the flight time as the weight of each edge. If the problem considers the ticket price of each flight, we assign the ticket price as the weight corresponding to the edge connecting the two cities.

The following figure shows the ticket prices for flights between the corresponding cities. Determine the route from San Francisco to Miami with the minimum total ticket cost.



Applying Dijkstra's algorithm to find the shortest path from San Francisco to Miami, we obtain the following result:
The shortest path is: **San Francisco → New York → Atlanta → Miami**,
and the length of this shortest path is **L(z) = 217**.

2.2.2. Minimum Total Cost Problem:
Problem 3: A new school is to be built in one of the seven communes of Ngọc Hồi district so that students from all seven communes can attend. The question is: *Which commune should be chosen for the school so that the total travel cost for all students is minimized?*
To solve this problem, we model each commune as a vertex of a graph. Each edge represents the road from one commune to another, and on each edge we assign the total travel cost for the students of that commune to the adjacent commune. Suppose the locations of the communes are modeled by the following graph:



Apply the Floyd algorithm to find the shortest paths between all pairs of vertices in the graph using the initial matrix D_0 =

Đỉnh	A	B	C	D	E	F	G
A	0	24					
B	24	0	51				
C		51	0	42			
D			42	0	34	25	32
E				34	0		
F				25		0	
G				32			0

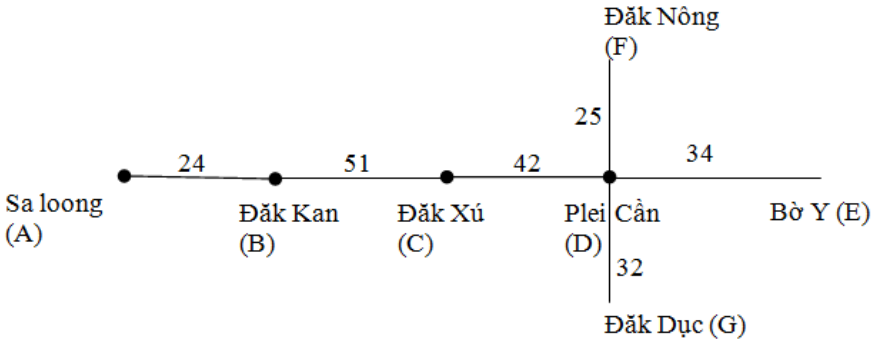
Finally, we obtain the matrix of the shortest distances between the vertices, denoted by $D = D_7 =$

Đỉnh	A	B	C	D	E	F	G
A	0	24	75	117	151	142	149
B	24	0	51	93	127	118	125
C	75	51	0	42	76	67	74
D	117	93	42	0	34	25	32
E	151	127	76	34	0	59	66
F	142	118	67	25	59	0	57
G	149	125	74	32	66	57	0

The sums of the corresponding rows are: 658, 538, 385, 343, 513, 468, and 503. Thus, D is the unique point with the minimum total. Therefore, the school should be built in Plei Cần commune.

2.2.3. The Minimax Problem (the Graph Center Problem):

Problem 4: A hospital is to be built in one of the seven communes of Ngọc Hồi district to serve the residents of all seven communes. Which commune should be selected for the hospital so that the residents of the farthest commune can reach the hospital in the shortest possible distance? The layout of the communes in Ngọc Hồi district is given as follows:



Similar to the problem of selecting a location to build a school (Problem 3), we apply the Floyd algorithm to find the shortest paths between all pairs of vertices in the graph, using the initial matrix $D_0 =$

Đỉnh	A	B	C	D	E	F	G
A	0	24					
B	24	0	51				
C		51	0	42			
D			42	0	34	25	32
E				34	0		
F				25		0	
G				32			0

Finally, we obtain the shortest-distance matrix between the vertices, denoted by $D = D_7 =$

Đỉnh	A	B	C	D	E	F	G
A	0	24	75	117	151	142	149
B	24	0	51	93	127	118	125
C	75	51	0	42	76	67	74
D	117	93	42	0	34	25	32
E	151	127	76	34	0	59	66
F	142	118	67	25	59	0	57
G	149	125	74	32	66	57	0

The eccentricities of the corresponding vertices are: A – 151, B – 127, C – 76, D – 117, E – 151, F – 142, and G – 149. We observe that C is the unique vertex with the minimum eccentricity. Therefore, the hospital should be built in Đăk Xú commune.

Conclusion

This paper has briefly presented Dijkstra's algorithm and the Floyd algorithm, as well as their applications to problems such as selecting the most economical route, the minimum–sum problem (the problem of choosing a location for building a school), and the minimax problem (the graph center problem). The aim of this work is to explore more deeply the applications of shortest-path algorithms to real-world problems that require finding cost-efficient routes. I hope that this paper contributes to bringing mathematics closer to practical contexts and helps address a wider range of real-life problems.

REFERENCES

[1] Rosen, K. H. (1991). *Discrete Mathematics and Its Applications*. McGraw–Hill Book Company.
[2] Trần Quốc Chiến (2006). *Giáo trình Lý thuyết đồ thị* [Graph Theory Textbook]. University of Da Nang.
[3] Nguyễn Gia Định (2003). *Giáo trình Toán rời rạc* [Discrete Mathematics Textbook]. Hue University Publishing House.
[4] Nguyễn Gia Định (2008). *Bài tập Toán rời rạc* [Discrete Mathematics Exercises]. Hue University Publishing House.
[5] Nguyễn Đức Nghĩa & Nguyễn Tô Thành (2007). *Toán rời rạc* [Discrete Mathematics]. Vietnam National University, Hanoi Publishing House.
[6] Đặng Huy Ruận (2005). *Lý thuyết Đồ thị và ứng dụng* [Graph Theory and Applications]. Science and Technics Publishing House.
[7] Đỗ Đức Giáo (2008). *Toán rời rạc* [Discrete Mathematics]. Vietnam National University, Hanoi Publishing House.