ISSN 2354-0575

TƯƠNG QUAN GIỮA CÁCH BIỂU DIỄN ĐỒ THỊ VÀ SỐ CẠNH CỦA ĐỒ THỊ

Nguyễn Hoàng Điệp, Nguyễn Thị Hải Năng, Ngô Thanh Huyền, Trịnh Thị Nhị
Trường Đại học Sư phạm Kỹ thuật Hưng Yên

Ngày nhận: 28/1/2016
Ngày xét duyệt: 02/3/2016

Tóm tắt:

Các bài toán trong lý thuyết đồ thị được ứng dụng trong nhiều lĩnh vực như bài toán tìm đường đi ngắn nhất giữa hai thành phố trong mạng giao thông. Để giải quyết một bài toán trên đồ thị, ngoài việc lựa chọn được một thuật toán tốt thì việc lựa chọn cấu trúc dữ liệu thích hợp để biểu diễn đồ thị đầu vào và cài đặt thuật toán cũng không kém phần quan trọng.

Bài báo này trình bày về hai cách biểu diễn đồ thị trên máy tính là ma trận trọng số và danh sách cạnh, trong đó cách biểu diễn đồ thị bằng ma trận trọng số thích hợp hơn cho các đồ thị dày cạnh còn biểu diễn bằng danh sách cạnh thì thích hợp hơn cho các đồ thị ít cạnh. Điều này được chứng minh bằng thực nghiệm trong giải quyết bài toán tìm đường đi ngắn nhất giữa hai đỉnh của một đơn đồ thị có trọng số.

Từ khóa: *Lý thuyết đồ thị, đường đi ngắn nhất, Dijkstra, ma trận trọng số, danh sách cạnh.*

1. Đặt vấn đề

Một thuật toán tốt và phù hợp luôn quan trọng và thường được quan tâm đầu tiên khi giải quyết một bài toán. Tuy nhiên, điều này là chưa đủ để có một sản phẩm tốt. Trong các bài toán nói chung và lý thuyết đồ thị nói riêng thì việc lựa chọn cấu trúc dữ liệu thích hợp với bài toán cũng quan trọng không kém.

Nếu thuật toán tốt nhưng cách biểu diễn dữ liệu tồi hoặc không thích hợp với dữ liệu đầu vào của bài toán thì thường cho kết quả không tốt như bộ nhớ sử dụng nhiều hay thời gian chạy lớn. Bài báo này tiếp cận vấn đề biểu diễn dữ liệu thích hợp với số lượng cạnh của đồ thị phục vụ việc cài đặt trên một bài toán của lý thuyết đồ thị.

Bài toán tìm đường đi ngắn nhất trên đồ thị là một bài toán quan trọng của lý thuyết đồ thị có nhiều ứng dụng trong thực tế [1]. Cho tới hiện nay, có nhiều thuật toán được đưa ra để giải quyết bài toán này, trong đó thuật toán được biết đến nhiều nhất là thuật toán Dijkstra. Thuật toán do nhà toán học Edsger W.Dijkstra đưa ra năm 1956 và được công nhận năm 1959 [4]. Thuật toán này dựa trên ý tưởng của nhà toán học Ford và Bellman, Dijkstra cải tiến và thiết kế thuật toán theo chiến lược "tham lam". Đây là một thuật toán tốt có độ phức tạp thời gian đa thức, ngày nay nó vẫn cải tiến và sử dụng [1,4,6,7].

Trong bài này, nhóm tác giả đã lựa chọn đồ thị đầy đủ để đại diện cho đồ thị có nhiều cạnh và chọn đồ thị vòng để đại diện cho đồ thị có ít cạnh. Cùng một bài toán tìm đường đi ngắn nhất từ một đỉnh tới một đỉnh, chạy trên cùng một máy tính, cùng một hệ điều hành, cùng cài đặt thuật toán Dijkstra. Tổng hợp thời gian chạy trên dữ liệu là đồ thị nhiều và ít cạnh, nhiều và ít đỉnh, biểu diễn bằng danh sách cạnh và ma trận kề. Từ kết quả đã tổng hợp, so sánh và nhóm đã đi tới kết luận được rằng việc lựa chọn cách biểu diễn dữ liệu có ảnh hưởng tới thời gian giải quyết bài toán!

2. Cơ sở lý thuyết

Đơn đồ thị có trọng số $G = (V, E)$ là một cặp trong đó V là tập các đỉnh, E là tập các cặp gồm hai phần tử khác nhau của V gọi là các cạnh trong đó không có cạnh nào bị lặp lại và mỗi cạnh được gắn với một số được gọi là trọng số của cạnh.

Đường đi từ đỉnh u đến đỉnh v trên đồ thị $G=(V, E)$ là dãy các đỉnh $x_0, x_1,..., x_{n-1}, x_n$ trong đó $u = x_0, v = x_n, (x_i, x_{i-1}) \in E, i = 0, 1, 2,..., n$-1 [2].

Đường đi có độ dài nhỏ nhất từ đỉnh u đến đỉnh v trên đồ thị có trọng số $G = (V, E)$ là đường đi mà tổng của các trọng số trên các cạnh của nó là nhỏ nhất.

Bài toán đường đi ngắn nhất: [2] Tìm đường đi có độ dài nhỏ nhất từ một đỉnh xuất phát $s \in V$ đến đỉnh cuối (đích) $t \in V$.

Thuật toán Dijkstra [2] Thiết kế dựa trên cơ sở gán nhãn tạm thời cho các đỉnh, nhãn của mỗi đỉnh cho biết độ dài đường đi ngắn nhất từ đỉnh s đến nó.

Ý tưởng chính của thuật toán là *giảm nhãn tới cực tiểu tại mỗi đỉnh*. Thuật toán như sau:

```
for v ∈ V do
{
    d[v]:=a[s,v];
    Truoc[v]:=s;
    xet[v]=false;
}
d[s]:=0;
sdx=0;
while (sdx<n || xet[T])
{
```

```
u=dinhconhanminchuaxet();
xet[u]=true;
sdx++;
for v ∈ ke(u) do
  if d[v]>d[u]+a[u,v] then
  {
    d[v]:=d[u]+a[u,v];
    Truoc[v]:=u;
  }
}
```

trong đó:

d[v]: Nhãn của đỉnh *v*;

Truoc[v]: Tên của một đỉnh kề ngay trước *v* trên đường đi;

Xet[v]: Đánh dấu đỉnh *v* nhãn đã cực tiểu hay chưa.

Đồ thị đầy đủ n đỉnh ($n \geq 3$) ký hiệu bởi K_n là đơn đồ thị vô hướng mà giữa hai đỉnh bất kỳ của nó luôn có cạnh nối [2].

Đồ thị vòng ký hiệu bởi C_n, $n \geq 3$ gồm *n* đỉnh $v_1, v_2, v_3, ..., v_{n-1}, v_n$ và *các cạnh* $(v_1, v_2), (v_2, v_3) ... (v_{n-1}, v_n), (v_n, v_1)$ [2].

Biểu diễn đồ thị có trọng số [2] $G = (V, E)$, $|V| = n$, $|E| = m$ trên máy tính .

Biểu diễn đồ thị trên máy tính bằng ma trận trọng số (MTTS), dùng ma trận kích thước $n \times n$, $A = (a_{i,j}: i, j = 1, 2, ..., n)$. Với các phần tử được xác định theo qui tắc sau đây:

$a_{i,j}$ = Trọng số cạnh, nếu có cạnh từ *i* sang *j* hay $(i, j) \in E$, $i, j = 1, 2, . . ., n$.

$a_{i,j} = 0$, trong trường hợp còn lại tức là không có cạnh (i, j).

Biểu diễn đồ thị trên máy tính bằng danh sách cạnh (DSC), dùng ma trận kích thước $m \times 3$, $E = (e_{i,j}: i = 1, 2,..., m, j = 1, 2, 3)$ cạnh thứ *i* ($e_{i,1}$, $e_{i,2}$) có trọng số $e_{i,3}$.

Biểu diễn đồ thị có trọng số nhiều cạnh, đại diện là đầy đủ có trọng số K_n bằng ma trận trọng số cần $n \times n$ phần tử nhớ, còn biểu diễn K_n bằng DSC cần $n \times (n - 1) \times 3$ phần tử nhớ, gấp 3 lần bộ nhớ so với biểu diễn bằng MTTS.

Mặt khác, biểu diễn đồ thị vòng có trọng số C_n bằng MTTS cần $n \times n$ phần tử nhớ, còn biểu diễn C_n bằng DSC cần $6 \times n$ phần tử nhớ so với $n \times n$ phần tử nhớ.

Vậy, để tiết kiệm bộ nhớ thì trong hai phương pháp biểu diễn đồ thị trên máy tính MTTS và DSC ở trên, nếu đồ thị dày cạnh nên dùng MTTS, còn nếu đồ thị thưa cạnh nên dùng DSC.

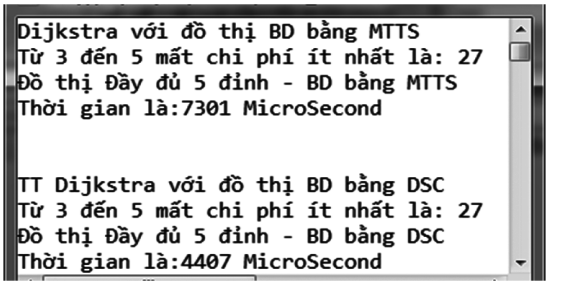
3. Kết quả

Chương trình cài đặt trên máy tính thinkpad T420 với cấu hình Core i5-2450M CPU - 2.5 GHz, DDR Ram 4G, hệ điều hành Windows 7, ngôn ngữ lập trình là C#. Để đảm bảo tính khách quan, dữ liệu sử dụng trong chương trình được nhóm tác giả lấy một cách ngẫu nhiên.

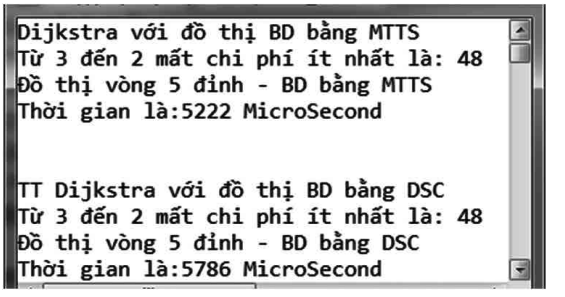
Đồ thị đơn *n* đỉnh K_n có trọng số không âm biểu diễn bằng MTTS tương ứng là ma trận vuông kích thước $n \times n$ với các phần tử nhận giá trị nguyên dương và đường chéo chính bằng không. Do đó tạo đồ thị ngẫu nhiên tương ứng tạo một ma trận vuông ngẫu nhiên thỏa mãn yêu cầu trên. Tương tự C_n biểu diễn bằng MTTS tương ứng là ma trận vuông $n \times n$ với 2 đường chéo phụ và $a_{1,n}$, $a_{n,1}$ khác không.

Sau khi nhóm tạo ngẫu nhiên K_n và C_n với *n=5, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500, 1000, 2000* biểu diễn bằng MTTS, kết quả sẽ được chuyển sang biểu diễn bằng DSC; sau đó lấy 2 đỉnh S và T ngẫu nhiên từ các đỉnh của đồ thị; tiếp đó chạy thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh S tới đỉnh T; cuối cùng nhóm ghi lại thời gian chạy thuật toán tính theo micro second (MS) và so sánh kết quả dưới dạng biểu đồ được kết quả như Hình 9.

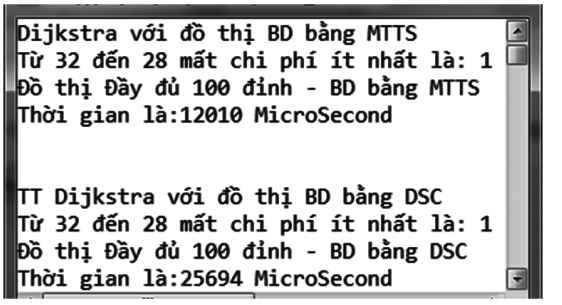
Kết quả chương trình tìm đường đi ngắn nhất từ S tới T trên đồ thị đầy đủ và đồ thị vòng với số đỉnh nhỏ là: 5 đỉnh; 100 đỉnh và đồ thị nhiều đỉnh là: 1000 đỉnh; 2000 đỉnh như sau:



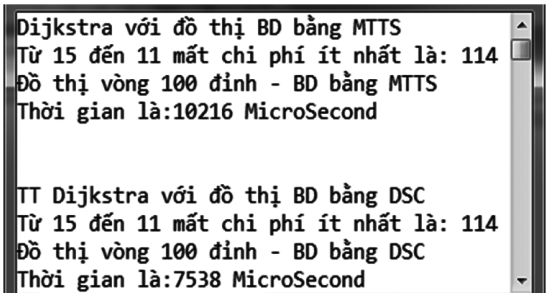
Hình 1. *Kết quả chạy trên đồ thị* K_5



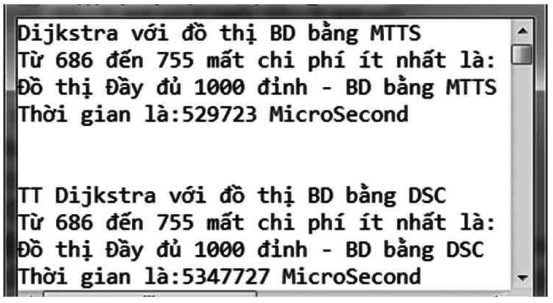
Hình 2. *Kết quả chạy trên đồ thị* C_5



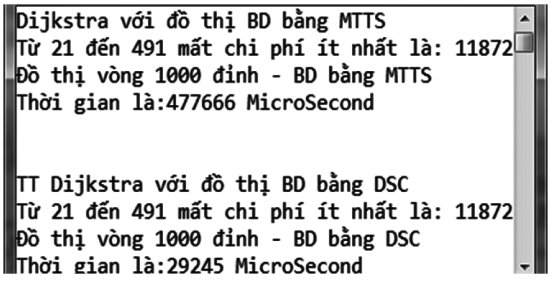
Hình 3. *Kết quả chạy trên đồ thị* K_{100}



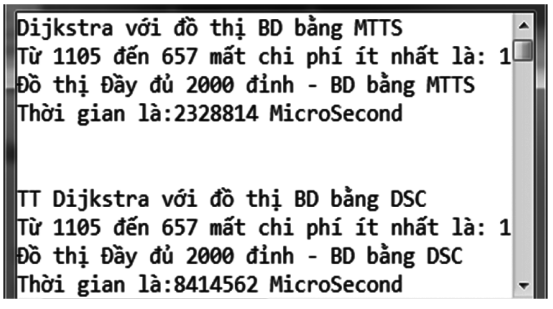
Hình 4. *Kết quả chạy trên đồ thị* C_{100}



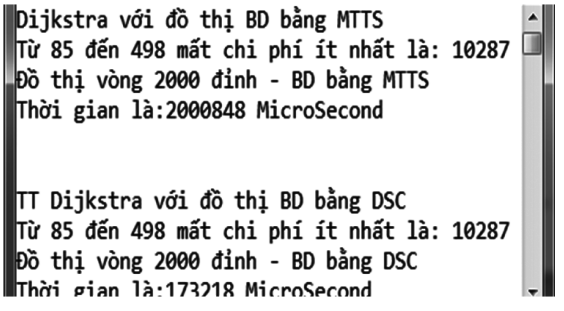
Hình 5. *Kết quả chạy trên đồ thị* K_{1000}



Hình 6. *Kết quả chạy trên đồ thị* C_{1000}



Hình 7. *Kết quả chạy trên đồ thị* K_{2000}



Hình 8. *Kết quả chạy trên đồ thị* C_{2000}

Trong Hình 1 và 2, đồ thị 5 đỉnh và số cạnh đồ thị chưa tới 20 cạnh thì thời gian chạy thuật toán trên đồ thị nhanh chưa tới 10 mili giây trên cả 2 cách biểu diễn đồ thị, thời gian hoàn thành chương trình hơn kém nhau vài mili giây: K_5 tương ứng là 7,301MS và 4,407MS; C_5 là 5,222MS và 5,786MS.

Hình 3 và 4, thời gian chạy trên đồ thị K_{100} và C_{100} biểu diễn theo hai cách biểu diễn đồ thị khác nhau không nhiều: 13,684MS trên K_{100} và 2,678MS trên C_{100}.

Khi số đỉnh của đồ thị lớn khoảng 1000 thì thời gian cần thiết để thực hiện thuật toán bắt đầu có sự tăng lên đáng kể, và thời gian biểu diễn theo 2 loại đồ thị theo 2 cách khác nhau tương đối lớn. Cụ thể với K_{1000} (Hình 5) biểu diễn bởi MTTS là 529,723MS và DSC là 5,347,727MS gấp 10 lần so với biểu diễn K_{1000} bằng MTTS. Với C_{1000}, thời gian chạy trên đồ thị biểu diễn bởi MTTS là 477,666MS gấp 20 lần so với biểu diễn bởi DSC là 29,245MS.

Hình 7 và 8 cho thấy khi kích thước đồ thị tăng lên là 2000 thì thời gian chạy thuật toán hơn kém nhau lớn, K_{2000} biểu diễn bằng MTTS mất khoảng 2 giây trong khi DSC khoảng 8 giây, C_{2000} biểu diễn bằng MTTS mất khoảng 2 giây trong khi DSC chỉ khoảng 1/10 giây.

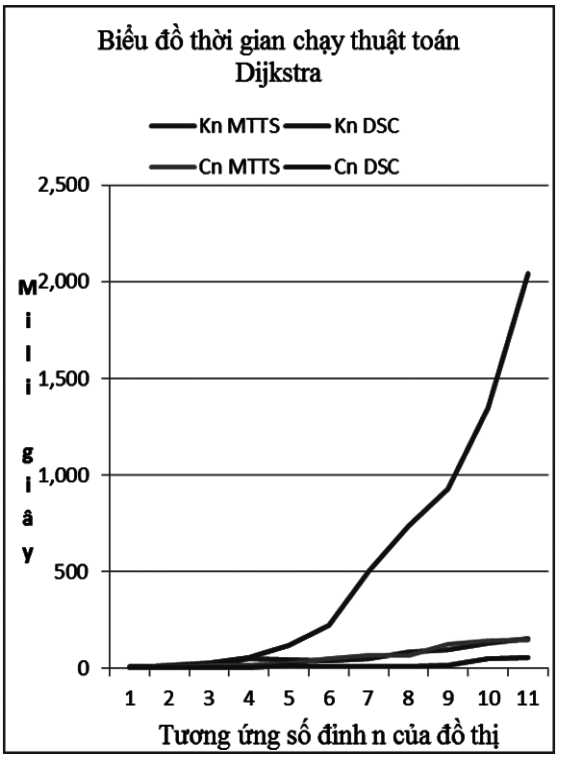
Nói cách khác khi số đỉnh của đồ thị nhỏ thì thời gian cần hoàn thành chương trình là tính theo mili giây, còn khi kích thước của đồ thị lớn (1000 hay 2000) thì thời gian cần hoàn thành chương trình là tính theo giây và có sự chênh lệch thời gian nhiều, thời gian chờ kết quả sẽ lâu hơn khi chọn cách biểu diễn dữ liệu đầu vào không phù hợp.

Bảng 1. *Thời gian chạy thuật toán*

n	Kn MTTS	Kn DSC	Cn MTTS	Cn DSC
5	7,301	4,407	5,222	5,786
50	10,241	17,979	8,924	6,267
100	12,010	25,694	10,216	7,538
150	52,460	53,311	16,738	6,393
200	44,058	121,471	24,439	12,934
250	36,877	220,901	51,123	11,401
300	48,226	503,152	67,575	9,821
350	83,381	740,255	69,445	8,190
400	92,918	925,273	123,038	17,584
450	130,303	1,345,006	142,348	48,025
500	152,279	2,043,115	145,963	12,730
1000	529,723	5,347,727	477,666	29,245
2000	2,328,814	8,414,562	2,000,848	173,218

So sánh kết quả dưới dạng biểu đồ được kết

quả như hình sau:



Hình 9. *Thời gian chạy (minisecond)thuật toán Dijkstra*

Hình 9 ở trên thể hiện rõ những điều sau: Đồ thị biểu biễn bởi MTTS phụ thuộc tuyến tính so với số lượng đỉnh của đồ thị; khi số lượng cạnh tăng lên thì thời gian cần thiết để hoàn thành không tăng lên nhiều; Đồ thị vòng C_n biểu diễn bởi DSC có thời gian chạy rất nhỏ; đồ thị đầy đủ cùng n đỉnh K_n biểu diễn bởi nó cần thời gian rất lớn.

Đường màu đỏ trong Hình 9 so với đường xanh da trời cho thấy biểu diễn K_n khi n lớn bằng DSC mất nhiều thời gian hơn biểu diễn bởi MTTS.

Đường màu xanh lá và màu tím cùng tỷ lệ thuận với kích thước của đồ thị, trong đó đường màu xanh lá cây tăng nhanh hơn đường màu tím, chỉ ra rằng thời gian chạy thuật toán Dijkstra với cùng đồ thị C_n thì biểu diễn bởi MTTS cần nhiều thời gian hơn DSC để hoàn thành chương trình.

Đường màu xanh lá và màu xanh da trời trong Hình 9 gần như trùng khớp lên nhau, điều này nói lên thời gian chạy thuật toán Dijkstra với cách biểu diễn đồ thị MTTS trên đồ thị dày cạnh hay ít cạnh không khác nhau nhiều.

Đường màu tím và màu đỏ trong Hình 9 thể hiện sự khác nhau rõ rệt nhất, khi n tăng lên thì đường màu đỏ tăng nhanh nhất còn đường màu tím tăng chậm nhất. Như vậy, DSC phù hợp nhất cho C_n nhưng không thích hợp nhất là K_n.

Như vậy, với đồ thị ít đỉnh thì biểu diễn đồ thị cách nào không quan trọng, nhưng đồ thị nhiều đỉnh thì nên cân nhắc lựa chọn cách biểu diễn đồ thị cho thích hợp. Trong hai cách biểu diễn đồ thị là MTTS và DSC, nếu đồ thị ít cạnh MTTS không quá xấu nhưng DSC sẽ cho kết quả nhanh hơn; nếu đồ thị dày cạnh nên chọn MTTS.

4. Kết luận

Nói chung, thuật toán tốt và biểu diễn dữ liệu phù hợp đều là nhân tố quan trọng ảnh hưởng tới độ phức tạp thời gian của thuật toán, do đó giải quyết một bài toán trên máy tính cả thuật toán và cách thức biểu diễn dữ liệu đều cần được chọn lựa cẩn thận.

Với việc cài đặt trên cùng cấu hình phần cứng, cùng một thuật toán giải quyết, cùng một bài toán trên các đồ thị đại diện sinh ngẫu nhiên thì những nội dung trình bày ở trên đã chứng minh được rằng thời gian chạy của chương trình phụ thuộc vào việc biểu diễn dữ liệu đầu vào.

Phương pháp biểu diễn dữ liệu không những ảnh hưởng tới thời gian cài đặt - chạy thuật toán Dijkstra mà còn đúng với các thuật toán khác của lý thuyết đồ thị. Điều này sẽ được nhóm tác giả chứng minh trong nhưng bài viết khác.

Tài liệu tham khảo

[1]. Bast H, Mehlhorn K, Schafer G, Tamaki H, *A Heuristic for Dijkstra's Algorithm with Many Targets and its Use in Weighted Matching Algorithms*, Algorithmica 2003; 36:75–88.
[2]. Kenneth H.Rosen, *Discrete Mathematics and its Applications*, McGraw Hill, 4th Edition 1999.
[3]. Kenneth H.Rosen, John G.Michels, Jonathan L.Gross, JerroldW.Grossman, Douglas R.Shier, *Handbook of Discrete and Combinatorial Mathematics*, CRC press, 1999.
[4]. Vlad Gogoncea, Gabriel Murariu, *The Use of Dijkstra's Algorithm in Waste Management Problem*, The Annals of "DUNĂREA DE JOS" University of Galati Fascicle V, Technologies in Machine Building, ISSN 1221-4566 2010.
[5]. https://en.wikipedia.org/wiki/Shortest_path_problem
[6]. http://www.ms.unimelb.edu.au/~moshe/620-261/dijkstra/dijkstra.html
[7]. http://hkuccst9003.blogspot.com/2011/10/application-of-different-shortest-path.html

ISSN 2354-0575

THE RELATED BETWEEN REPRESENT GRAPHS AND THE NUMBER OF IT

Abstract:

Graph theory is used in many applications such as finding the shortest path between two cities in a transportation network. To solve a problem by graph theory, picking up a good algorithm is important as chosing a suitable data structure in representing weighted graphs.

This paper presents weight matrices, lists all edges to represent weight graphs, and shows the first way which is better than the second way in graphs having many edges, in contrast, the second way is better than the first way in graphs that has few edges. The problem of finding the shortest path on weight graphs is proved by experiment using Dijkstra algorithm to verify.

Keywords: *Graph theory, shortest path, Dijkstra algorithm, weighted matrices, list edges.*