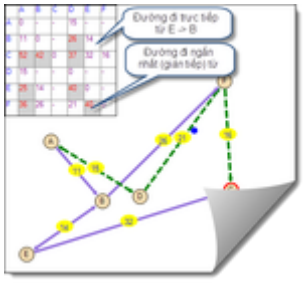
YinYang's Programing Blog

"Education is not a preparation for life; education is life itself"

Algorithm: Floyd–Warshall và bài toán đường đi ngắn nhất giữa moi căp đỉnh trên đồ thi



(https://yinyangit.files.wordpress.com/2011/05/visual-graph-floyd.png)Thuật toán Floyd-Warshall hay còn gọi là Floyd, Roy-Floyd là thuật toán tìm đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị có trọng số được công bố năm 1962 bởi Robert Floyd . Trong bài này tôi sẽ giới thiệu cách cài đặt thuật toán này trong chương trình Y2 Visual Graph (http://wp.me/ptl4u-jT) đã giới thiệu trước đây.

Thuật toán Floyd cài đặt khá đơn giản, dựa trên ý tưởng: Nếu có đường đi từ i tới k và từ k tới j nhỏ hơn đường đi hiện tại từ i tới j thì ta sẽ cập nhật đường đi từ i tới j thành đường đi từ **i tới k cộng với từ k tới j**. Ta gọi k là đỉnh trung gian của i và j. Như vậy sau khi thực hiện thuật toán, sẽ có một số cạnh "ảo" được sinh ra, tức là các cạnh không nối trực tiếp giữa hai đỉnh.

```
1  for k:=1 to n do
2  for i:=1 to n do
3  for j:=1 to n do
4  path[i,j] := min (path[i,j] , path[i,k] + path[k,j] );
```

Như vậy mặc dù có độ phức tạp lên tới O(n^3) nhưng ta chỉ cần thực hiện 1 lần là có thể tìm đường đi ngắn nhất giữa hai đỉnh bất kì.

Tham khảo thêm tại:

http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
(http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm)

Cài đặt bằng C#

Trong bài toán này tôi sử dụng ma trận kề với giá trị data[i,j] = -1 để biểu diễn không có đường đi giữa hai đỉnh i và j. Sau đó tạo thêm một mảng 2 chiều FloydData để lưu kết quả sau quá trình thực hiện thuật toán Floyd.

```
1  public void Floyd()
2  {
3      if (Data == null)
4          return;
5      int n = Data.GetLength(0);
6
7      FloydData = new MatrixCell[n, n];
8
9      for (int i = 0; i < n; i++)
10         for (int j = 0; j < n; j++)
11             FloydData[i, j] = new FloydCell(Data[i, j]);
12
13     for (int i = 0; i < n; i++)
14     {
15         for (int j = 0; j < n; j++)
16         {
17             if (FloydData[j, i].Value > 0)
18             {
19                 for (int k = 0; k < n; k++)
20                 {
21                     if (FloydData[i, k].Value > 0)
22                     {
23                         if (FloydData[j, k].Value < 0 || FloydData[j, i].Val
24                         {
25                             FloydData[j, k].Value = FloydData[j, i].Value +
26
27                             FloydData[j, k].Previous = i;
28                         }
29                     }
30                 }
31             }
32         }
33     }
34 }
```

FloydCell là một struct được định nghĩa để lưu trữ giá trị trọng số của cạnh và đỉnh trung gian:

```
1  struct FloydCell
2  {
3      public int Previous;
4      public int Value;
5
6      public FloydCell(int value)
7          : this(-1, value)
8      { }
9      public FloydCell(int previous, int value)
10     {
11         this.Previous = previous;
12         this.Value = value;
13     }
14 }
```

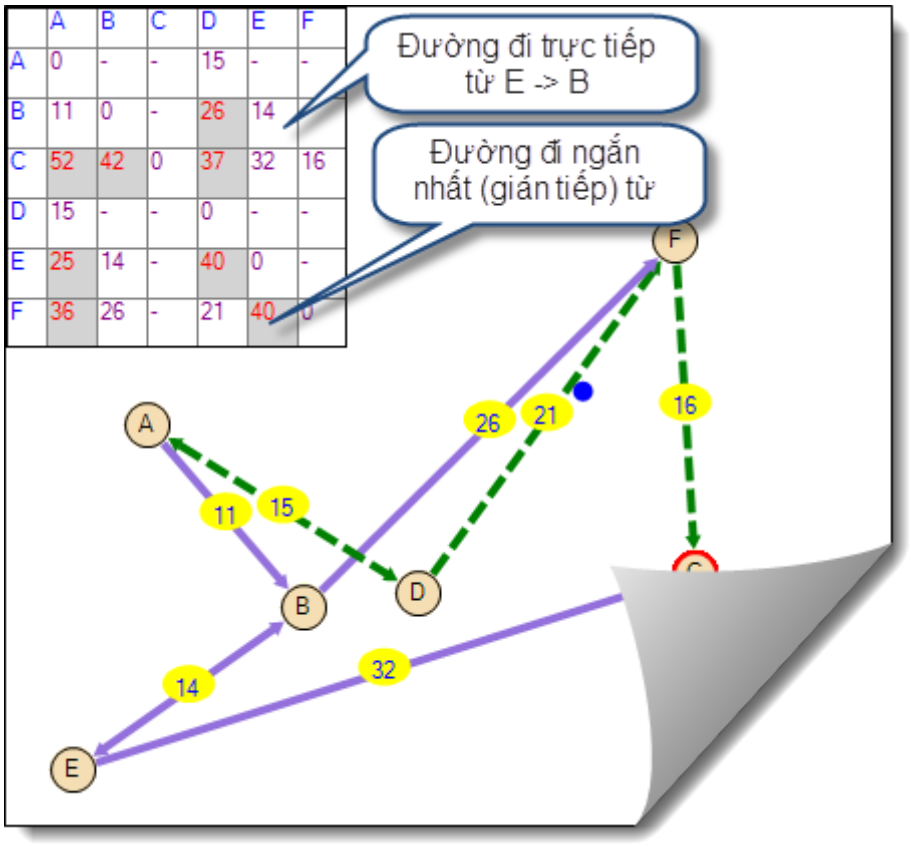
Lý do ta cần phải lưu lại đỉnh trung gian là để lấy được đường đi qua các đỉnh từ đỉnh bắt đầu đến kết thúc. Bạn sẽ thấy rõ hơn công dụng của giá trị này trong phần tiếp theo.

Lấy đường đi ngắn nhất giữa hai đỉnh

```
1  List<Point> _path = new List<Point>();
2  // […]
3  void GetPath(int x,int y)
4  {
5      if(_matrix.FloydData[x,y].Value== -1)
6          return;
7      int i=_matrix.FloydData[x,y].Previous;
8      if(i== -1)
9      {
10         _path.Add(new Point(x+1,y+1));
11         return;
12     }
13     else
14     {
15         GetPath(x,i);
16         _path.Add(new Point(_preNodeIndex+1,i+1));
17         _preNodeIndex=i;
18         GetPath(i,y);
19     }
20 }
```

Phương thức GetPath() trên sẽ tạo ra đường đi giữa hai đỉnh x, y và lưu vào trong collection _path. Giá trị của **FloydData[x,y].Previous** sẽ trả về đỉnh trung gian giữa hai đỉnh x,y và nếu giá trị này bằng -1 tức là có một cạnh nối trực tiếp giữa hai đỉnh này. Ngược lại ta chia đường đi thành hai phần và thực hiện đệ quy để lấy đường đi của từng phần này.

Minh họa trong Y2 Visual Graph



	A	B	C	D	E	F
A	0	-	-	15	-	-
B	11	0	-	26	14	
C	52	42	0	37	32	16
D	15	-	-	0	-	-
E	25	14	-	40	0	-
F	36	26	-	21	40	0

(https://yinyangit.files.wordpress.com/2011/05/visual-graph-floyd.png)

https://yinyangit.wordpress.com (https://yinyangit.wordpress.com)

Bài này đã được đăng trong Algorithms và được gắn thẻ Graph, Shortest Path. Đánh dấu đường dẫn tĩnh.

ıe thought on “Algorithm: Floyd–Warshall và bài toán đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị”

Trần phương nói: *Thứ Hai, Tháng Ba 13, 2017 lúc 6:01 sáng*

0

0

i
Rate This

Anh cho e hỏi có thể áp thuật toán Dijkstra vào để vẽ đồ thị k ạ?

Đã đóng bình luận.