

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

---0000000---



TIỂU LUẬN CUỐI KÌ

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
ĐỀ TÀI: MINIMUM SPANNING TREE

Giảng viên hướng dẫn : PGS.TS Nguyễn Thị Hồng Minh

Nhóm sinh viên thực hiện : GR15

Thành viên:

1. Thân Lê Quang Cường
2. Nguyễn Văn Hải Dương
3. Phùng Văn Dũng

Hà Nội - 2022

Mục Lục

I. Những khái niệm bổ trợ	1
1. Đồ thị vô hướng, liên thông, chu trình	1
2. Cây là gì ?	1
II. Cây khung nhỏ nhất (Minimum Spanning Tree)	1
1. Khái niệm	1
2. Thuật toán	2
3. Cài đặt	5
4. Ứng dụng trong thực tế	8
III. Kết luận	12
Tài liệu tham khảo	12

I. Những khái niệm bổ trợ

1. Đồ thị vô hướng, liên thông, chu trình

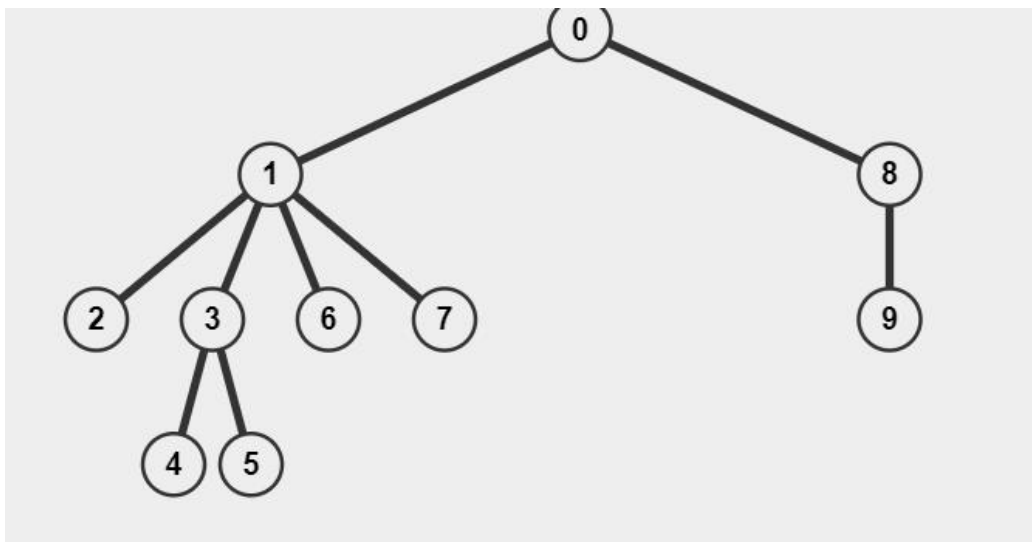
- Đồ thị là một cấu trúc rời rạc bao gồm các đỉnh và các cạnh nối các đỉnh này.

Ký hiệu đồ thị $G = (X, U)$, trong đó : X là tập các đỉnh - là tập các đối tượng nào đó; $U \subset X \times X$ là tập các cạnh, mỗi cạnh (x, y) nối đỉnh x với y ($x, y \in X$).

- Đồ thị vô hướng là đồ thị mà các cạnh của nó không có hướng.
- Một đồ thị được gọi là liên thông (connected) nếu có đường đi giữa mọi cặp đỉnh phân biệt của đồ thị. Ngược lại, đồ thị này được gọi là không liên thông (disconnected).
- Chu trình là 1 thuật ngữ trong lý thuyết đồ thị chỉ một đường đi đóng. Tức là 1 đường đi từ 1 đỉnh bất kỳ qua tất cả các đỉnh trong đồ thị và trở lại đỉnh đầu tiên.

2. Cây là gì ?

Cây là một đồ thị vô hướng liên thông và không có chu trình.

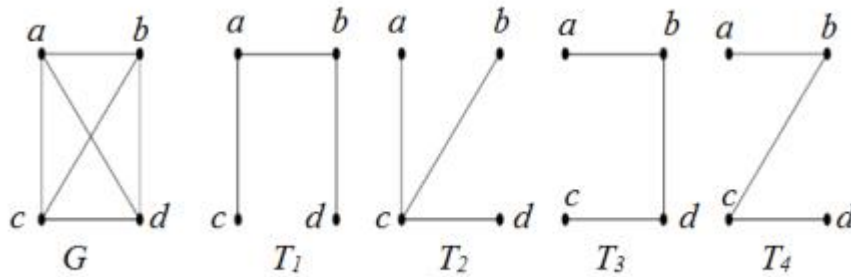


II. Cây khung nhỏ nhất (Minimum Spanning Tree)

1. Khái niệm

- Cây khung (*spanning tree*) của đồ thị là một tập hợp các cạnh của đồ thị thỏa mãn tập cạnh này *không chứa chu trình* và *liên thông* (từ một đỉnh bất kỳ có thể đi tới bất kỳ đỉnh nào khác theo mà chỉ dùng các cạnh trên cây khung)

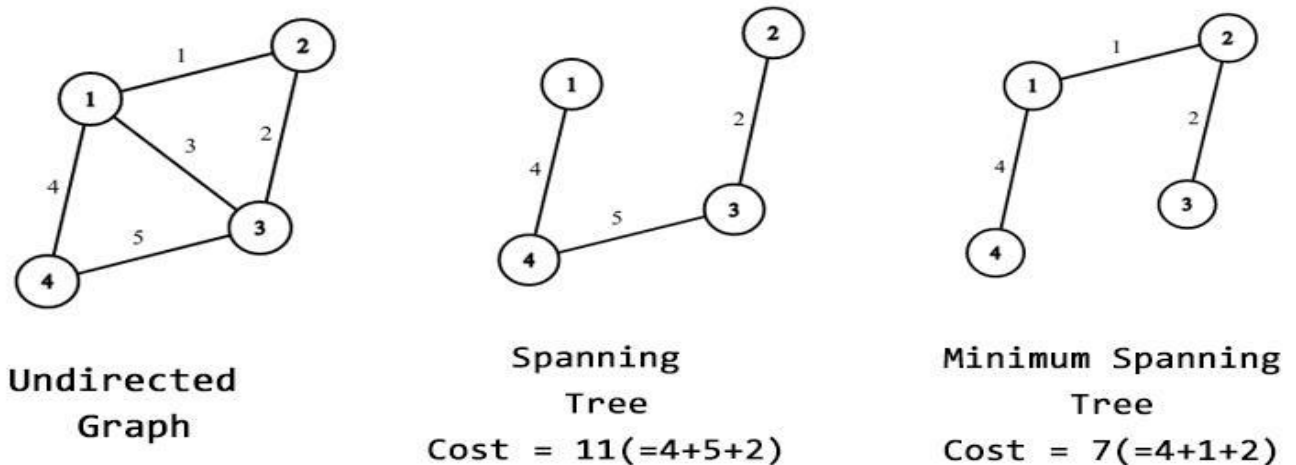
Ví dụ 1: cây khung trong đồ thị vô hướng không trọng số



T_1 , T_2 , T_3 và T_4 là các cây khung của đồ thị G .

- Trong đồ thị có trọng số, cây khung nhỏ nhất (*minimum spanning tree*) là cây khung có tổng trọng số các cạnh trong cây nhỏ nhất.

Ví dụ 2: cây khung nhỏ nhất trong đồ thị vô hướng có trọng số



2. Thuật toán

Giả sử G là một đồ thị vô hướng liên thông và có trọng số.

Khi đó, đồ thị G có cây bao trùm và sẽ có cây bao trùm nhỏ nhất.

a, Thuật toán Prim

Thuật toán Prim tìm cây khung nhỏ nhất dựa trên nguyên tắc: Từ một đỉnh tùy của G , chọn lần lượt đủ $n - 1$ cạnh bằng cách mỗi lần thêm một cạnh kề có trọng số nhỏ nhất (trong số tất cả các cạnh kề với các đỉnh đã được chọn) sao cho các cạnh đã chọn không tạo thành chu trình.

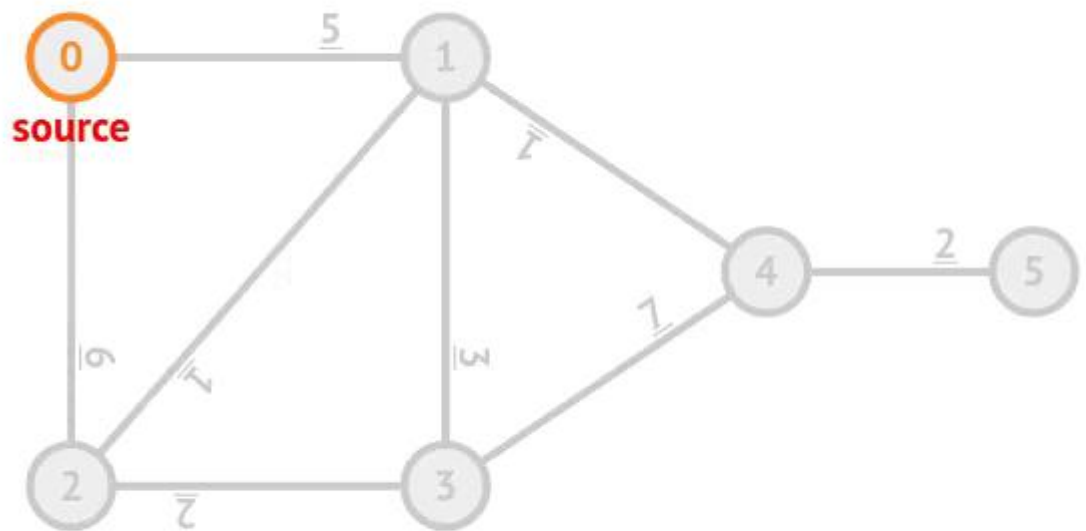
Mô tả như sau:

Bước 1. Chọn ngẫu nhiên 1 đỉnh và thêm vào tập hợp các đỉnh V.

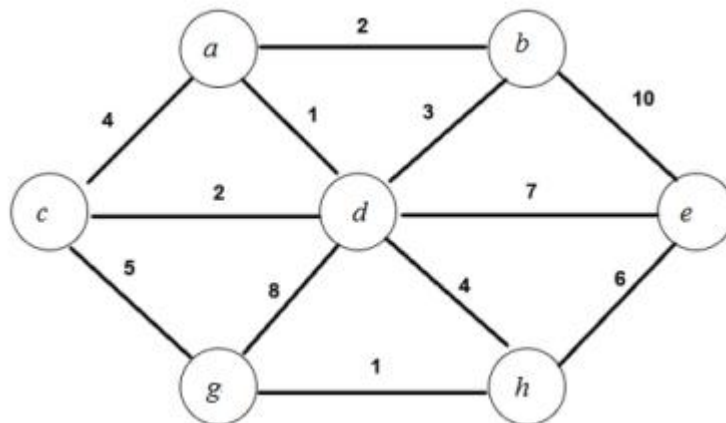
Bước 2. Chọn 1 đỉnh chưa có trong tập V mà có kết nối với 1 đỉnh trong V, cạnh tạo từ 2 đỉnh đó phải là cạnh có trọng số nhỏ nhất và thêm vào tập hợp các cạnh E.

Bước 3. Lặp lại bước 2 cho đến khi cây khung hoàn thành (Cách nhận biết cây khung hoàn thành là tất cả các đỉnh của cây khung đều đã xuất hiện trong V), cây khung nhỏ nhất là cây khung được tạo từ tập các cạnh trong E.

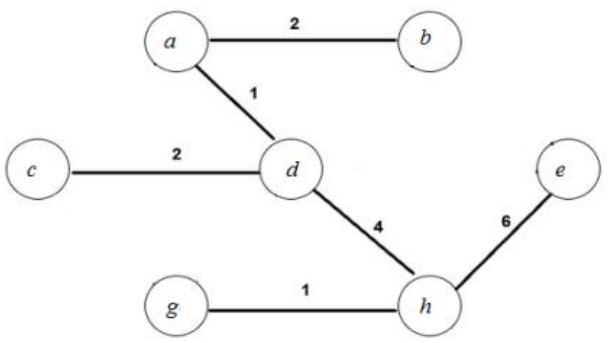
Minh họa:



Ví dụ 3:



Theo thuật toán Prim:

TT	Các cạnh kề	Cạnh kề nhỏ nhất	Đỉnh đã chọn	Cây khung nhỏ nhất
0			a	 <p style="text-align: center;">T_0 Trọng số của CKNN là: $l(T_0) = 16$.</p>
1	$(a,d) = 1$	$(a,d) = 1$	d	
2	$(a,b) = 2$ $(d,c) = 2...$	$(a,b) = 2$	b	
3	$(d,c) = 2$ $(b,e) = 10...$	$(d,c) = 2$	c	
4	$(c,g) = 5$ $(d,h) = 4...$	$(d,h) = 4$	h	
5	$(h,g) = 1$ $(c,g) = 5...$	$(h,g) = 1$	g	
6	$(h,e) = 6...$	$(h,e) = 6$	e	

b, Thuật toán Kruskal

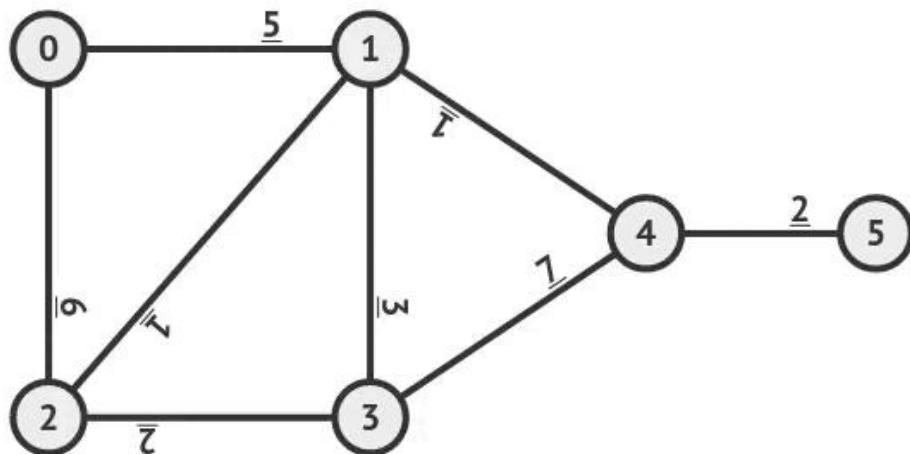
Thuật toán Kruskal: Thuật toán Kruskal tìm cây khung nhỏ nhất dựa trên nguyên tắc: Chọn lần lượt đủ $n - 1$ cạnh bằng cách mỗi lần chọn thêm một cạnh có trọng số nhỏ nhất, trong số các cạnh còn lại chưa được chọn của G , sao cho các cạnh đã chọn không tạo thành chu trình.

Mô tả như sau:

- (1) : Sắp xếp tập cạnh U của G theo thứ tự trọng số tăng
- (2) : Chọn lần lượt các cạnh của dãy trên sao cho các cạnh đã chọn không tạo thành chu trình.

Dừng khi chọn đủ $n - 1$ cạnh.

Ví dụ 4:



3. Cài đặt

- Sử dụng cách triển khai đồ thị bằng ma trận kề vì đó là cách trực quan và dễ hiểu nhất mặc dù chưa phải là cách triển khai có hiệu năng tốt nhất.
- Gồm 1 class static Graph (1 class Result nhỏ bên trong Graph) và 1 hàm main :

- Code :

```
public class PrimAlgorithmAdjacencyMatrix {

    static class Graph{
        int vertices;
        int matrix[][];

        public Graph(int vertices) {
            this.vertices = vertices;
            matrix = new int[vertices][vertices];
        }

        public void addEdge(int source, int destination, int weight) {
            // thêm cạnh
            matrix[source][destination]=weight;

            // thêm cạnh ngược lại trong đồ thị không trọng số
            matrix[destination][source] = weight;
        }

        // trả về đỉnh có trọng số nhỏ nhất chưa nằm trong MST
        int getMinimumVertex(boolean[] mst, int[] key){
            int minKey = Integer.MAX_VALUE;
            int vertex = -1;
            for (int i = 0; i < vertices; i++) {
                if(mst[i] == false && minKey > key[i]){
                    minKey = key[i];
                    vertex = i;
                }
            }
        }
    }
}
```

```
    }  
    return vertex;  
}
```

```
class Result {  
    // trả về đỉnh cha từ đó mà nó đi tới  
    int parent;  
    // chứa trọng số của cạnh để tính tổng trọng số của MST  
    int weight;  
}
```

```
public void primMST(){  
    boolean[] mst = new boolean[vertices];  
    Result[] result = new Result[vertices];  
    int[] key = new int[vertices];  
  
    // Khởi tạo các ptu mảng key đều là số rất lớn  
    (Integer.MAX_VALUE)  
    // Khởi tạo mảng chứa kết quả với tất cả các đỉnh  
    for (int i = 0; i < vertices; i++) {  
        key[i] = Integer.MAX_VALUE;  
        result[i] = new Result();  
    }  
  
    // bắt đầu từ đỉnh 0  
    key[0] = 0;  
    result[0] = new Result();  
    result[0].parent = -1; // đỉnh 0 là đỉnh bắt đầu nên không có cha  
  
    // tạo ra MST  
    for (int i = 0; i < vertices; i++) {  
  
        // tìm đỉnh có key bé nhất chưa nằm trong MST  
        int vertex = getMinimumVertex(mst, key);  
  
        // thêm đỉnh này vào MST  
        mst[vertex] = true;
```



```

// lặp qua tất cả đỉnh kề với đỉnh trên và cập nhật mảng key
for (int j = 0; j < vertices; j++) {
    // kiểm tra xem có cạnh đi từ đỉnh vertex đến j ko
    if(matrix[vertex][j] > 0){
        // kiểm tra xem đỉnh j có ở trong MST chưa
        // nếu ko thì kiểm tra xem key có cần cập nhật ko
        if(mst[j] == false && matrix[vertex][j] < key[j]){
            // cập nhật key
            key[j] = matrix[vertex][j];
            // cập nhật kết quả
            result[j].parent = vertex;
            result[j].weight = key[j];
        }
    }
}
// in ra kết quả
printMST(result);
}

```

```

public void printMST(Result[] result){
    int total_min_weight = 0;
    System.out.println("Minimum Spanning Tree: ");
    for (int i = 1; i < vertices ; i++) {
        System.out.println("Edge: " + i + " - " + result[i].parent +
            " key: " + result[i].weight);
        total_min_weight += result[i].weight;
    }
    System.out.println("Total minimum key: " + total_min_weight);
}

```

```

public void printAdjacentMatrix() {
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            System.out.print(matrix[i][j] + " ");
        }
    }
}

```

```

        System.out.println();
    }
}

public static void main(String[] args) {
    int vertices = 6;
    Graph graph = new Graph(vertices);
    graph.addEdge(0, 1, 4);
    graph.addEdge(0, 2, 3);
    graph.addEdge(1, 2, 1);
    graph.addEdge(1, 3, 2);
    graph.addEdge(2, 3, 4);
    graph.addEdge(3, 4, 2);
    graph.addEdge(4, 5, 6);
    graph.primMST();
}
}

```

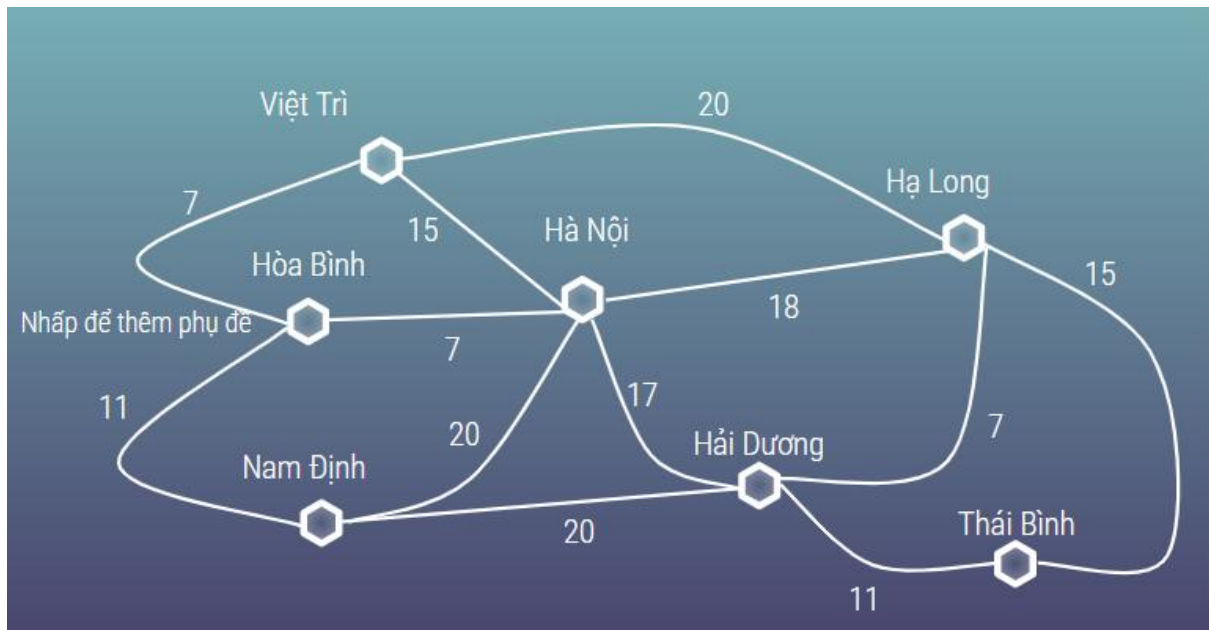
Độ phức tạp thuật toán Prim: phụ thuộc vào cấu trúc dữ liệu sử dụng để cài đặt đồ thị. Ở đây, nhóm em sử dụng ma trận kề vì đây là cách dễ hiểu và dễ thực hiện. Tuy nhiên, độ phức tạp sẽ là $O(v^2)$. Nếu sử dụng danh sách kề và min heap thì độ phức tạp sẽ là $O((v+e) \log v)$ với v là số đỉnh và e là số cạnh của đồ thị.

4. Ứng dụng trong thực tế

a, Xây dựng một hệ thống đường dây tải điện từ nhà máy điện đến các nơi tiêu thụ; sao cho chi phí nhỏ nhất. Mỗi thành phố tương ứng là một đỉnh của đồ thị.

Mô tả

- Giả sử có đồ thị các trạm tải điện của thành phố ở phía Bắc như sau :



- Đầu tiên để giải bài toán, ta ký hiệu cho các thành phố thành các đỉnh trong đồ thị :

Việt Trì : 0

Hòa Bình : 1

Hà Nội : 2

Nam Định : 3

Hải Dương : 4

Hạ Long : 5

Thái Bình : 6

- Từ đồ thị trên ta có ma trận kề :

	0	1	2	3	4	5	6
0	*	7	15	*	*	20	*
1	7	*	7	11	*	*	*
2	15	7	*	20	17	18	*
3	*	11	20	*	20	*	*
4	*	*	17	20	*	7	11
5	20	*	18	*	7	*	15
6	*	*	*	*	11	15	*

- Chạy thuật toán ta được kết quả :

```

21 {15,7,0,20,17,18,0},
22 {0,11,20,0,20,0,0},
23 {0,0,17,20,0,7,11},
24 {20,0,18,0,7,0,15},
25 {0,0,0,0,11,15,0}};
26
27 Graph g = new Graph(conn);
28 g.print();
29 g.Prím();
30 }
31
32

```

Run: testProg

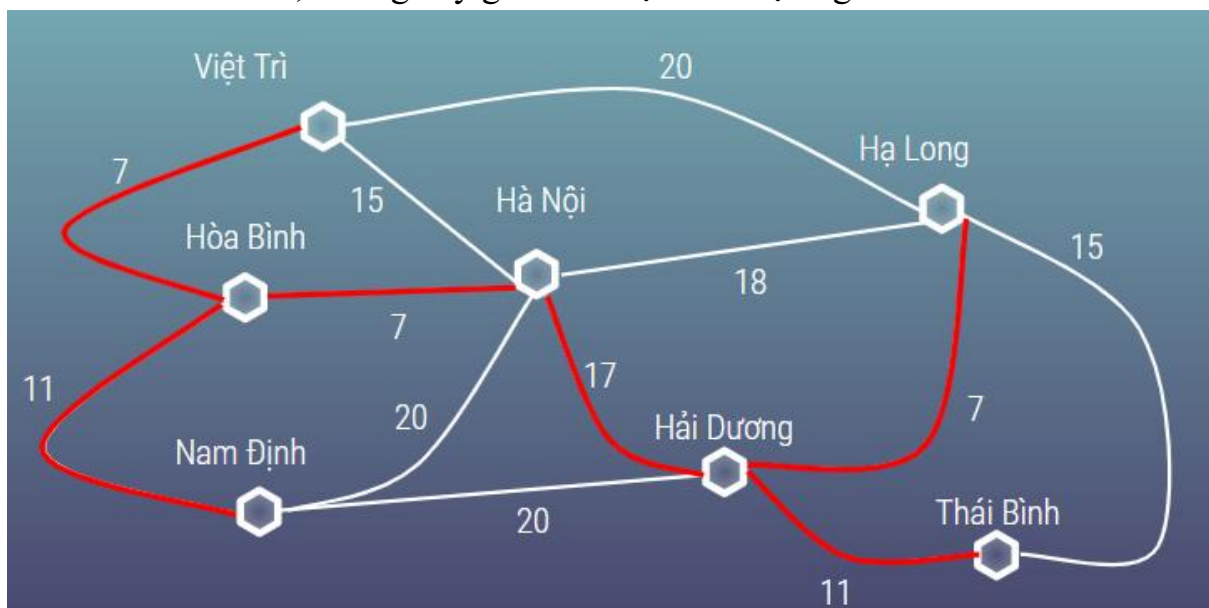
```

ReachSet = 0 1 2 3 4 5 6
0 --> 0
0 --> 1
1 --> 2
1 --> 3
2 --> 4
4 --> 5
4 --> 6

```

Process finished with exit code 0

- Tức là, đường dây giữa các trạm tải điện ngắn nhất sẽ là như sau :



- Quãng đường ngắn nhất là : 60.

Ứng dụng trên là 1 trong những ứng dụng cơ bản, dễ hiểu. Ngoài ra còn các ứng dụng khác như :

➤ Truyền hình cáp

Một ví dụ là một công ty truyền hình cáp đặt cáp đến một khu phố mới. Nếu nó là hạn chế để chôn cáp chỉ theo những con đường nhất định, do đó sẽ có một đồ thị đại diện những điểm đó được nối với nhau bằng những con đường. Một trong số những con đường có thể tồn kém hơn, bởi vì nó là dài hơn, hoặc yêu

cầu các cáp được chôn sâu hơn. Một cây bao trùm cho đồ thị đó sẽ là một tập hợp con của những con đường mà không có chu kỳ nhưng vẫn kết nối đến từng nhà. Có thể có một vài cây khung có thể tồn tại. Một cây bao trùm tối thiểu sẽ phải ràng buộc với tổng chi phí thấp nhất.

➤ Thiết kế mạch điện tử

Trong thiết kế mạch điện tử, nó thường là cần thiết để nối dây một số chân lại với nhau để làm cho nó có điện tương đương. Một cây bao trùm tối thiểu cần chi phí ít nhất của dây để kết nối một tập hợp điểm.

➤ Quần đảo kết nối

Giả sử chúng ta có một nhóm các hòn đảo mà chúng ta muốn liên kết với cây cầu để nó có thể đi du lịch từ một hòn đảo bất kỳ khác trong nhóm. Giả sử thêm rằng chính phủ muốn chi tiêu số tiền tối thiểu về dự án này. Các kỹ sư có thể tính toán chi phí cho một liên kết cầu mỗi cặp có thể có của hòn đảo. Tập hợp các cây cầu đó sẽ giúp ta đi du lịch từ hòn đảo bất kỳ đến tất cả các đảo khác với chi phí tối thiểu cho chính phủ là cây bao trùm tối thiểu

➤ Bài toán xây dựng hệ thống đường sắt.

Giả sử ta muốn xây dựng một hệ thống đường sắt nối n thành phố sao cho hành khách có thể đi từ bất kỳ một thành phố nào đến bất kỳ một trong các thành phố còn lại. Mặt khác trên quan điểm kinh tế đòi hỏi là chi phí xây dựng hệ thống đường phải nhỏ nhất. Rõ ràng đồ thị mà đỉnh là các thành phố còn các cạnh là các tuyến đường sắt nối các thành phố tương ứng với phương án xây dựng tối ưu phải là cây. Vì vậy, bài toán đặt ra dẫn về bài toán tìm cây khung nhỏ nhất trên đồ thị đầy đủ n đỉnh, mỗi đỉnh tương ứng với một thành phố, với độ dài trên các cạnh chính là chi phí xây dựng đường ray nối hai thành phố tương ứng (chú ý là trong bài toán này ta giả thiết là không xây dựng tuyến đường sắt có các nhà ga phân tuyến nằm ngoài các thành phố).

➤ Bài toán nối mạng máy tính.

Cần nối mạng một hệ thống gồm n máy tính đánh số từ 1 đến n . Biết chi phí nối máy i với máy j là $c[i,j]$, $i, j = 1, 2, \dots, n$ (thông thường chi phí này phụ thuộc vào độ dài cáp nối cần sử dụng). Hãy tìm cách nối mạng sao cho tổng chi phí nối mạng là nhỏ nhất.

➤ Các ứng dụng gián tiếp của MST

- Phân tích cụm: phân cụm điểm trong mặt phẳng, phân cụm liên kết đơn, lý thuyết đồ thị phân cụm và phân cụm dữ liệu biểu hiện gen.
- Đăng ký hình ảnh và phân đoạn
- Nhận dạng chữ viết tay của các biểu thức toán học.
- Mô tả thị trường tài chính

III. Kết luận

+) Về mặt lý thuyết, nhóm đã trình bày đầy đủ ứng dụng sử dụng cấu trúc dữ liệu đã học gồm giải thích các khái niệm, thuật toán, mô tả bài toán, cách thực hiện chương trình, ứng dụng thuật toán trên vào bài toán thực tế.

+) Về hoạt động nhóm, nhóm hiểu và vận dụng được kiến thức đã học, biết tìm hiểu kiến thức mới. Xác định được vấn đề, yêu cầu cần hoàn thành trong thời gian nhất định. Khi nhận thấy vấn đề hoặc chỉ ra điểm thiếu sót của chương trình, nhóm có tinh thần cùng nhau giải quyết. Nhóm chúng tôi cũng nhận thấy việc phân chia công việc hợp lý và kết hợp lại một cách linh hoạt giúp nhóm hoàn thành chương trình nhanh hơn, tốt hơn.

+) Về mục tiêu của bài tiểu luận, nhóm đạt được mục tiêu thực hiện đề tài tiểu luận môn học. Làm quen với việc chủ động và tự thực hiện một đề tài học tập, nghiên cứu; nâng cao kỹ năng làm việc nhóm của các thành viên, hiểu biết sâu về cấu trúc dữ liệu và ứng dụng; tăng cường kỹ năng lập trình ứng dụng.

Tài liệu tham khảo

- <https://visualgo.net/en/mst>
- <https://vnoi.info/wiki/algo/graph-theory/minimum-spanning-tree.md>
- <https://graphonline.ru/en/>
- Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser - Data Structures and Algorithms in Java-Wiley

Hết.