

LỜI NÓI ĐẦU

Bài toán tối ưu là một bài toán muôn thuở trong tin học bởi nó có ứng dụng thực tế cao. Ví dụ như làm thế nào để hoàn thành một công việc nào đó với chi phí ít nhất, hay đi như thế nào để đến đích sớm nhất, v.v. Cùng với sự trợ giúp của máy tính và các thuật toán hiệu quả đã giúp chúng ta giải quyết bài toán tối ưu một cách dễ dàng và chính xác. Một trong số các thuật toán hiệu quả đó là ***thuật toán Dijkstra*** - thuật toán tìm đường đi ngắn nhất và ***thuật toán Prim*** – Thuật toán tìm cây khung nhỏ nhất trên đồ thị có trọng số không âm. Trong bài viết này chúng tôi chỉ nghiên cứu trên đồ thị vô hướng có trọng số không âm.

Với trình độ lý luận còn thiếu chặt chẽ, kiến thức thực tiễn còn non yếu nên chắc chắn còn nhiều sai sót. Do vậy chúng tôi mong nhận được sự giúp đỡ bổ sung khiếm khuyết của quý thầy cô giáo, cùng các bạn trong lớp tạo điều kiện cho bài tìm hiểu này được hoàn chỉnh hơn. Tôi xin chân thành cảm ơn sự giúp đỡ tận tình của thầy giáo Hồ Hữu Linh và tất cả các bạn.

Sinh viên thực hiện:
Nguyễn Thanh Nhựt

PHẦN I: GIẢI THUẬT_LƯU ĐỒ THUẬT TOÁN DIJKSTRA

Thuật toán Dijkstra, mang tên của nhà khoa học máy tính người Hà Lan Edsger Dijkstra, là một thuật toán giải quyết bài toán đường đi ngắn nhất nguồn đơn trong một đồ thị không có cạnh mang trọng số âm.

1. Phân tích.

Dùng ma trận kề để biểu diễn đồ thị $C = (c_{ij})$, c_{ij} = trọng số của cung (i,j) , $c_{ij} = +\infty$ nếu không có cung (i,j) . Một mảng $d[]$ để ghi các độ dài đường đi ngắn nhất từ s tới đỉnh i đang có. Xuất phát $d[s] = 0$ và $d[i] = c_{si}$ nếu i kề s , $d[j] = +\infty$ nếu j không kề s .

2 Giải thuật tìm đường đi ngắn nhất giữa một cặp đỉnh.

Định nghĩa 1.0.

Xét đồ thị có trọng số cạnh $G = (V, E, \omega)$, với hàm trọng số $\omega: E \rightarrow \mathbf{R}$ là ánh xạ từ tập các cạnh E đến tập số thực \mathbf{R} .

Định nghĩa 1.1. Đường đi p từ đỉnh u đến đỉnh v là dãy các cạnh nối tiếp nhau bắt đầu từ đỉnh u kết thúc tại đỉnh v . Đường đi p từ u đến v được biểu diễn như sau: $p = (u = v_0, v_1, \dots, v_k = v)$

Định nghĩa 1.2. Độ dài của đường đi $p = (v_0, v_1, \dots, v_k)$, ký hiệu $\omega(p)$, là tổng các trọng số của các cạnh trên đường đi:

$$\omega(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$$

Định nghĩa 1.3. Gọi $\wp(u, v)$ là tập tất cả đường đi từ u đến v . Độ dài đường đi ngắn nhất từ đỉnh u đến đỉnh v được xác định bởi:

$$d(u, v) = \min \{ \omega(p) \mid p \in \wp(u, v) \}$$

Định nghĩa 1.4. Đường đi ngắn nhất $p_{\min}(u, v)$ từ đỉnh u đến đỉnh v là đường đi có độ dài $d(u, v)$.

3 Giải thuật Dijkstra.

3.1 Nội dung.

Có rất nhiều giải thuật đã được phát triển để giải bài toán tìm đường đi ngắn nhất giữa một cặp đỉnh, trong khuôn khổ bài viết này tôi chỉ xin giới thiệu giải thuật Dijkstra. Giải thuật Dijkstra là một giải thuật để giải bài toán đường đi ngắn nhất nguồn đơn trên một đồ thị có trọng số cạnh mà tất cả các trọng số đều không âm. Nó xác định đường đi ngắn nhất giữa hai đỉnh cho trước, từ đỉnh a đến đỉnh b .

Ở mỗi đỉnh v , giải thuật Dijkstra xác định 3 thông tin: k_v , d_v và p_v .

k_v : mang giá trị boolean xác định trạng thái được chọn của đỉnh v .

Ban đầu ta khởi tạo tất cả các đỉnh v chưa được chọn, nghĩa là:

$k_v = false, \forall v \in V$.

d_v : là chiều dài đường đi mà ta tìm thấy cho đến thời điểm đang xét từ a đến v .

Khởi tạo, $d_v = \infty, \forall v \in V \setminus \{a\}, d_a = 0$.

p_v : là đỉnh trước của đỉnh v trên đường đi ngắn nhất từ a đến b . Đường đi ngắn nhất từ a đến b có dạng $\{a, \dots, p_v, v, \dots, b\}$. Khởi tạo, $p_v = null, \forall v \in V$.

Sau đây là các bước của giải thuật Dijkstra:

B1. Khởi tạo: Đặt $k_v := false \forall v \in V; d_v := \infty, \forall v \in V \setminus \{a\}, d_a := 0$.

B2. Chọn $v \in V$ sao cho $k_v = false$ và $d_v = \min \{d_t / t \in V, k_t = false\}$

Nếu $d_v = \infty$ thì *kết thúc*, không tồn tại đường đi từ a đến b .

B3. Đánh dấu đỉnh $v, k_v := true$.

B4. Nếu $v = b$ thì *kết thúc* và d_b là độ dài đường đi ngắn nhất từ a đến b .

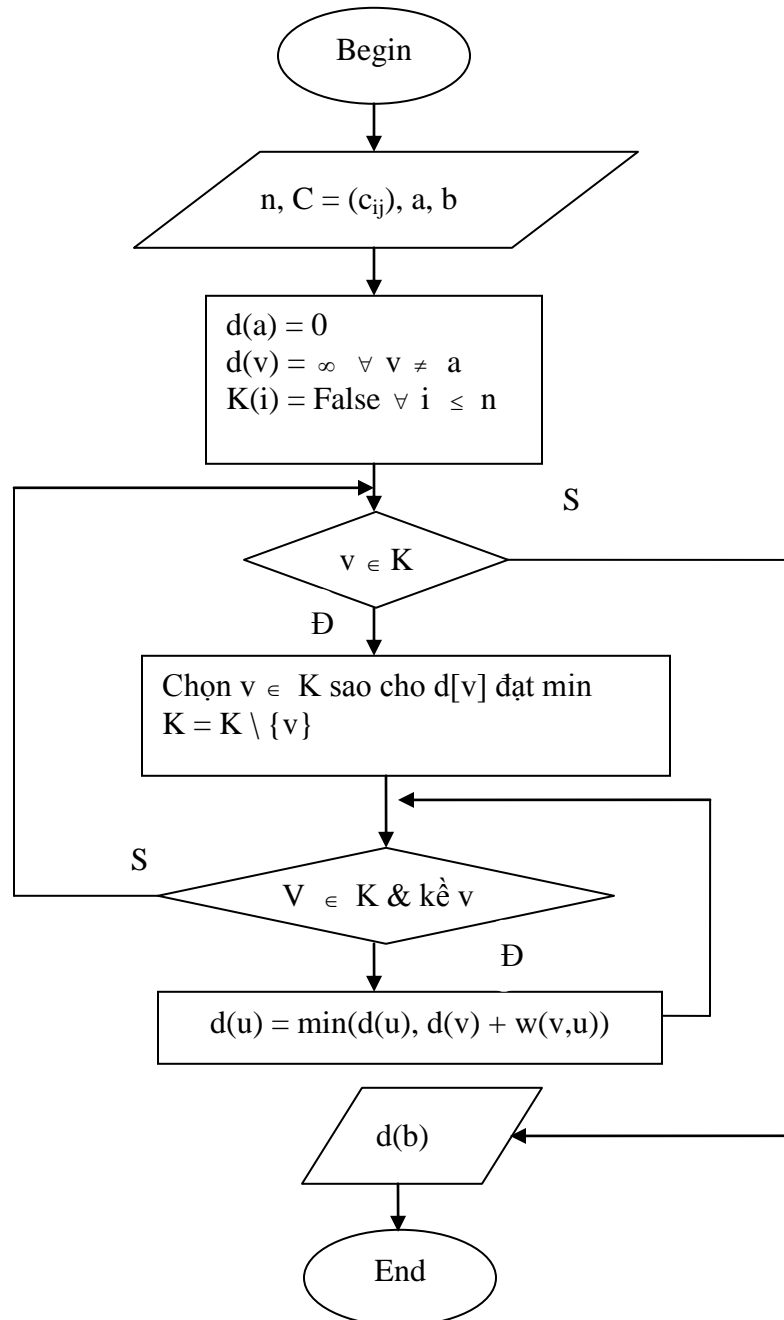
Ngược lại nếu $v \neq b$ sang **B5**.

B5. Với mỗi đỉnh v kề với u mà $k_v = false$, kiểm tra

Nếu $d_v > d_u + w(u, v)$ thì $d_v := d_u + w(u, v)$

Ghi nhớ đỉnh $v: p_v := u$. Quay lại **B2**.

3.2 Lưu đồ thuật toán Dijkstra



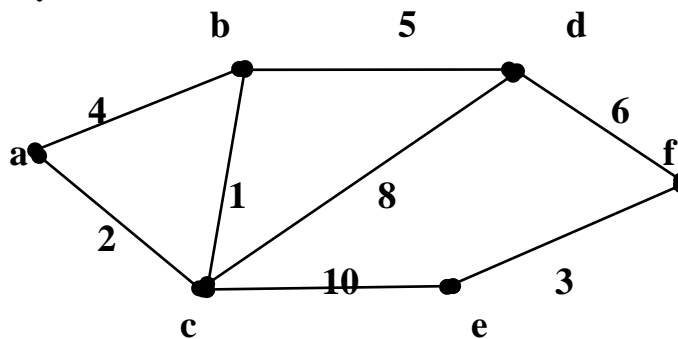
3.3 Bảng dữ liệu chạy thô.

Tạo ma trận như sau

```

6
0 4 2 0 0 0
4 0 1 5 0 0
2 1 0 8 10 0
0 5 8 0 0 6
0 0 10 0 0 3
0 0 0 6 3 0
    
```

Ta có đồ thị như sau:



Bảng chạy thô

V	T	a	b	c	d	e	f
		0	∞	∞	∞	∞	∞
A	bcdef	*	4	2*	∞	∞	∞
C	bdfd		3*	*	10	12	∞
B	cf d		*		8*	12	∞
D	f d				*	12*	14
E	d					*	14
F							*

độ dài từ a \rightarrow f là 14.

4. Cài đặt chương trình

Đề tài : Chương trình tìm đường đi ngắn nhất từ đỉnh a đến đỉnh b theo thuật toán Dijkstra _Sử dụng ngôn ngữ lập trình C++

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <conio.h>

using namespace std;

#define MAX 50
#define INFINITY 10000

int n, end, begin;
int c[MAX][MAX], p[MAX], d[MAX];
bool k[MAX];

void init(){
    ifstream filein( "dijkstra.txt", ios::in);
    filein >> n;

    cout << " v.";
    for(int i=0; i<n; i++){
        cout << setw(3) << i;
    }
    cout << endl;
    for(int i=0; i<n; i++){
        cout << setw(3) << i;
        for(int j=0; j<n; j++){
            filein >> c[i][j];
            if(c[i][j]==0){
                c[i][j]= INFINITY;
                cout << " ";
            } else cout << setw(3) << c[i][j];
        }
        cout << endl;
    }
}

void printResult(){
    int i,j;
```

```
        cout << "\nDuong di ngan nhat tu " << begin << " den " << end << "
la:\n" << end;
        i = p[end];
        while(i!=begin){
            cout << " " << i;
            i = p[i];
        }

        cout << " " << begin << "\nVoi Do dai duong di la: " << d[end]<<"\n\n";
        int getch();
    }

    int minimum(){
        int u, v, dmin = INFINITY;
        for(v = 0; v<n; v++){
            if(!k[v] && d[v] < dmin){
                dmin = d[v];
                u = v;
            }
        }
        return u;
    }

    void dijkstra(){
        int u, v;
        for(v = 0; v<n; v++){
            d[v] = c[begin][v];
            p[v] = begin;
            k[v] = false;
        }

        k[begin] = true;

        while(!k[end]){
            u = minimum();
            k[u] = true;

            if(k[end]) break;
            for(v = 0; v<n; v++){
                if(!k[v] && d[u]+c[u][v] < d[v]){
                    d[v] = d[u] + c[u][v];
                    p[v] = u;
                }
            }
        }
    }
```

```

    }
}

int main(){
    init();
    cout << "\n=====GHOSTPRO2010@GMAIL.COM===== ";
    cout << "\nNhập điểm đầu và điểm cuối: ";
    cin >> begin >> end;
    dijkstra();
    printResult();
    return 0;
}

```

5. Demo chương trình.

- Sau khi cài đặt xong chương trình trên (có chương trình kèm theo) cho ta kết quả như sau: (lấy ví dụ ở mục 3.3)

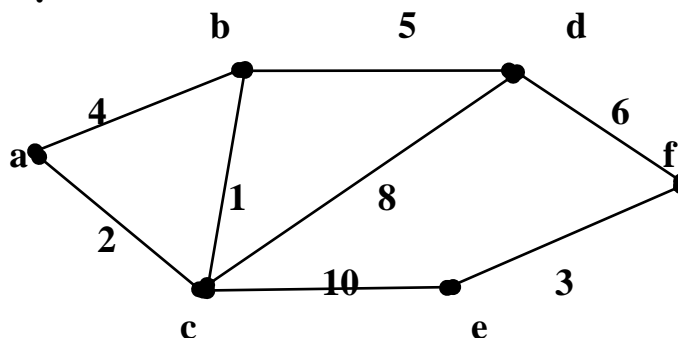
Cho ma trận :

```

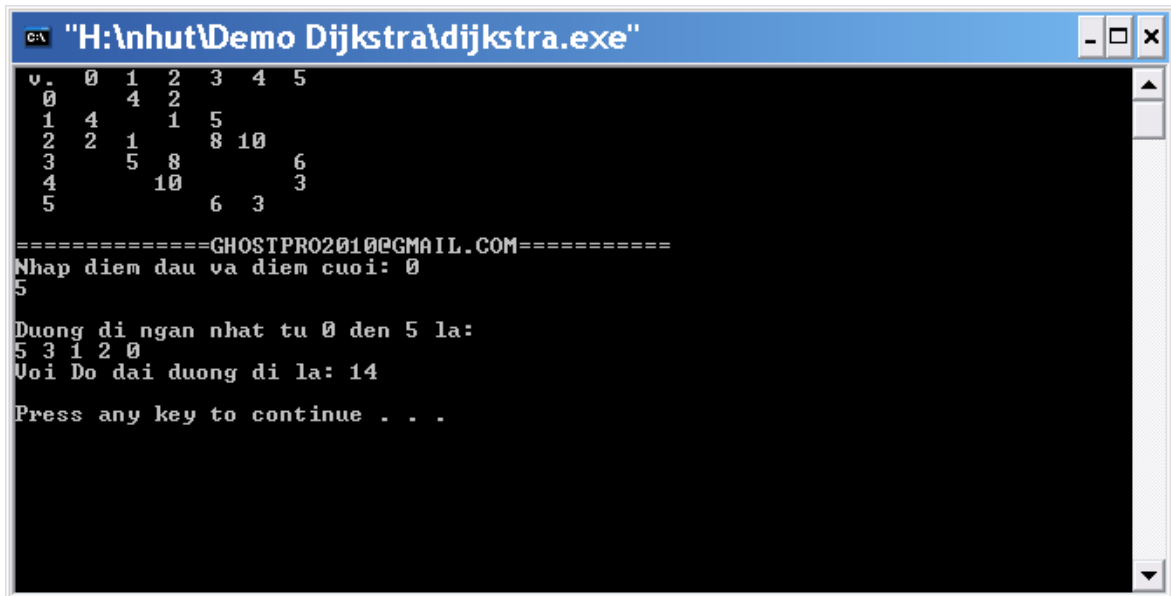
6
0 4 2 0 0 0
4 0 1 5 0 0
2 1 0 8 10 0
0 5 8 0 0 6
0 0 10 0 0 3
0 0 0 6 3 0

```

Ta có đồ thị như sau:



Kết quả của chương trình:



```
"H:\nhut\Demo Dijkstra\dijkstra.exe"
v.  0  1  2  3  4  5
0   0  4  2   5
1   4  0  1   5
2   2  1  0  8 10
3   5  8  0   6
4  10  3  0   0
5   6  3  0   0

=====GHOSTPRO2010@GMAIL.COM=====
Nhap diem dau va diem cuoi: 0
5

Duong di ngan nhat tu 0 den 5 la:
5 3 1 2 0
Voi Do dai duong di la: 14

Press any key to continue . . .
```

PHẦN II: THUẬT TOÁN PRIM

1. Thuật toán Prim

- Có rất nhiều thuật toán để tìm cây khung nhỏ nhất, một trong số đó thường hay sử dụng là phương pháp lân cận gần nhất của Prim. Thuật toán đó có thể phát biểu hình thức như sau:

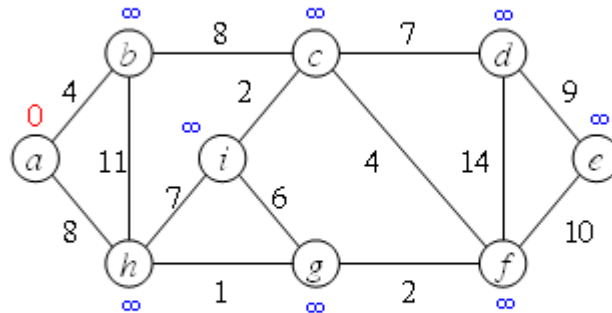
Đơn đồ thị vô hướng $G = (V, E)$ có n đỉnh được cho bởi ma trận trọng số C . Qui ước $c[u, v] = +\infty$ nếu (u, v) không là cạnh. Xét cây T trong G và một đỉnh v , gọi **khoảng cách từ v tới T** là trọng số nhỏ nhất trong số các cạnh nối v với một đỉnh nào đó trong T :

$$d[v] = \min\{c[u, v] \mid u \in T\}$$

- for ($v \in V$) {
 - ▶ $d[v] := +\infty$;
 - ▶ $\text{Free}[v] := \text{True}$; //Chưa có đỉnh nào \in cây
- $d[0] = 0$;
- for ($k = 0$; $k < n$; $k++$)
 - {
 - ▶ $u := \min(d[v] \mid \text{với mọi } v \text{ và } \text{Free}[v] = \text{True})$; //Chọn u là đỉnh có nhãn tự do nhỏ nhất
 - ▶ $\text{Free}[u] = \text{False}$; //Cố định nhãn đỉnh u và kết nạp u vào cây
 - ▶ For ($v \in V$: $(u, v) \in E$) if $d[v] > c[u, v]$
 - {
 - $d[v] := c[u, v]$; $\text{Trace}[v] := u$;
 - }
- Thông báo cây khung gồm có các cạnh $(\text{Trace}[v], v)$ với $v \in V$

2. Thực thi giả thuật Prim.

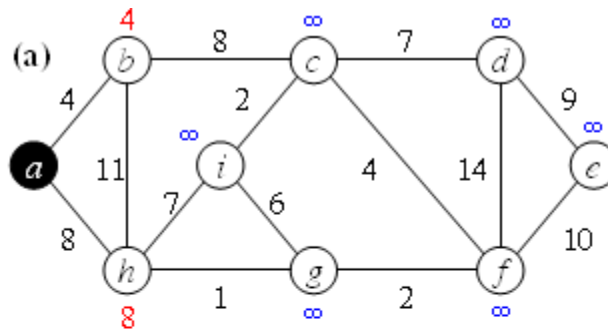
Sau khi khởi động tất cả các đỉnh đều ở trong V.



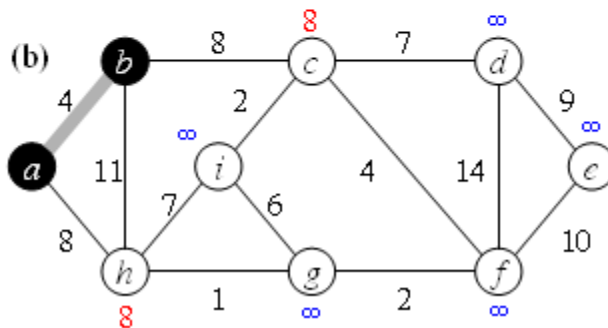
- T (trace) là tập các cạnh xám.

- các đỉnh trong V màu trắng, các đỉnh được đưa ra khỏi V màu đen.

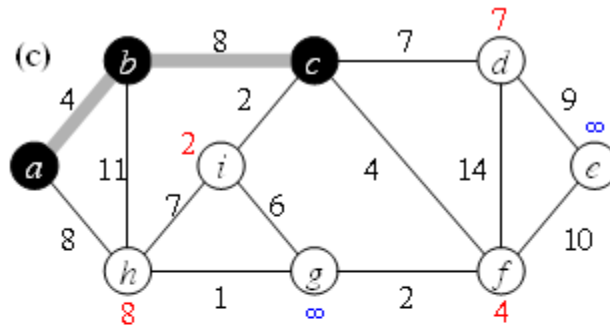
Sau lần lặp 1:



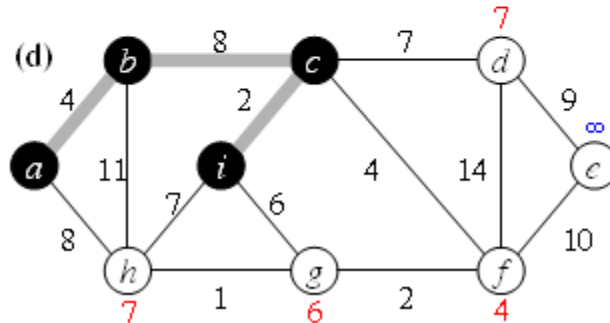
Sau lần lặp 2:



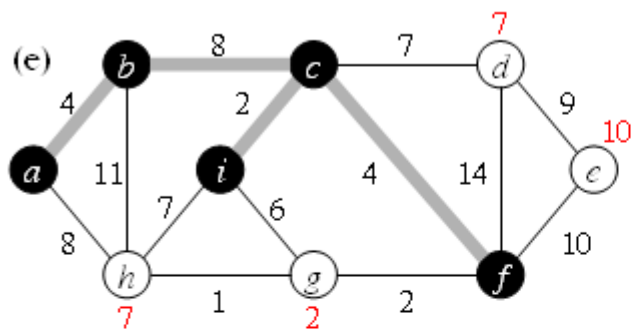
Sau lần lặp 3:



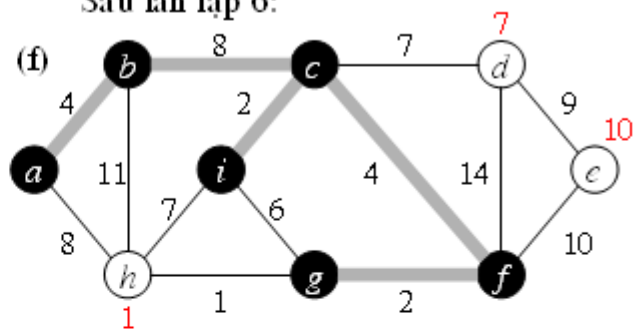
Sau lần lặp 4:



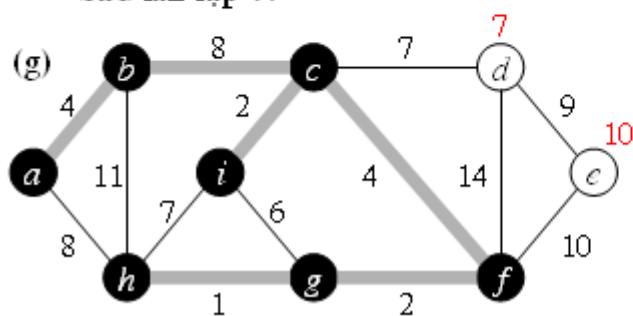
Sau lần lặp 5:



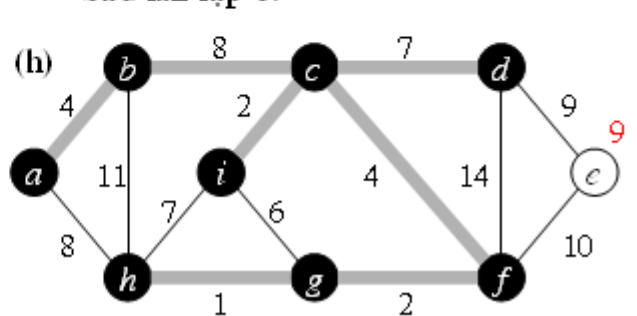
Sau lần lặp 6:



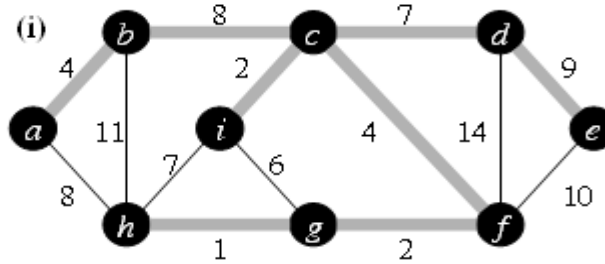
Sau lần lặp 7:



Sau lần lặp 8:



Sau lần lặp 9:



Độ dài nhỏ nhất của cây khung là: $4+8+7+9+4+2+1+2=37$

3. Cài đặt chương trình

```
#include <iostream>
```

```
#include <iomanip>
```

```
#include <fstream>
```

```
#include <conio.h>
```

```
using namespace std;
```

```
#define MAX 50
```

```
#define INFINITY 10000
```

```
int n, minw = 0;
```

```
int c[MAX][MAX], trace[MAX], d[MAX];
```

```
bool Free[MAX];
```

```
void init(){
```

```
    ifstream filein( "prim1.txt", ios::in);
```

```
    filein >> n;
```

```
    cout << " v.";
```

```
    for(int i=0; i<n; i++){
```

```
        cout << setw(3) << i;
```

```
    }
```

```
    cout << endl;
```

```
    for(int i=0; i<n; i++){
```

```
    cout << setw(3) << i;
    for(int j=0; j<n; j++){
        filein >> c[i][j];
        if(c[i][j]==0){
            c[i][j]= INFINITY;
            cout << "  ";
        } else cout << setw(3) << c[i][j];
    }
    cout << endl;
}

bool prim(){
    for(int i=0; i<n; i++){
        Free[i] = 1;// dinh dau tien co nhan khoang cach la 0
        d[i] = INFINITY; //cac dinh khac co nhan khoang cach la + vo
        cung(vo cung lon)
    }
    d[0] = 0;

    for(int k=0; k<n; k++){
        int u = - 1;// Chon dinh u chua duoc ket nap co d[u] nho nhat
        int min = INFINITY;
        for(int i=0; i<n; i++){
            if(Free[i] && d[i]<min){
                min = d[i];
                u = i;
            }
        }

        if(u == -1) return false;//Neu không chọn được u nào có  $d[u] < \infty$  thì đồ thị không liên thông
        Free[u] = false;//Neu chọn được thì đánh dấu u đã bị kết nối
        minw += d[u];
        for(int v=0; v<n; v++){
```

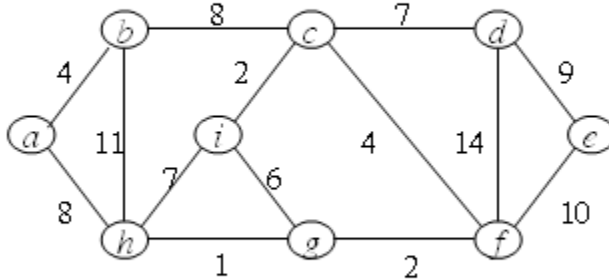
```
        if(Free[v] && d[v]>c[u][v]){ //Tinh lai các nhãn khoảng
cách d[v] ( v chưa được kết nạp)
            trace[v] = u;
            d[v] = c[u][v]; //đỉnh nối với v cho khoảng cách ngắn
nhất là u
        }
    }
}
return true;
}
```

```
void printTreeEdgeList(){
    cout << "\n Cay Khung Nho Nhat La: ";
    for(int v=1; v<n; v++){
        cout << "( " << v << ", " << trace[v] << " ) ";
    }
    cout << "\n\n Do Dai Cay Khung La: " << minw << endl;
}
```

```
int main(){
    init();
    if(!prim()) {
        cout << "Do Thi Da Cho Khong Lien Thong!";
    } else printTreeEdgeList();
    getch();
    return 0;
}
```

3. Demo thuật toán Prim

Cho đồ thị và ma trận tương ứng sau :



	0	1	2	3	4	5	6	7	8
0	0	4	0	0	0	0	0	8	0
1	4	0	8	0	0	0	0	11	0
2	0	8	0	7	0	4	0	0	2
3	0	0	7	0	9	14	0	0	0
4	0	0	0	9	0	10	0	0	0
5	0	0	4	14	10	0	2	0	0
6	0	0	0	0	0	2	0	1	6
7	8	11	0	0	0	0	1	0	7
8	0	0	2	0	0	0	6	7	0

Kết quả của thuật toán Prim :

```

prim3
v. 0 1 2 3 4 5 6 7 8
0 0 4 0 0 0 0 8 0
1 4 0 8 0 0 0 11 0
2 0 8 0 7 0 4 0 2
3 0 0 7 0 9 14 0 0
4 0 0 0 9 0 10 0 0
5 0 0 4 14 10 0 2 0
6 0 0 0 0 0 2 1 6
7 8 11 0 0 0 1 0 7
8 0 0 2 0 0 6 7 0

Cay Khung Nho Nhat La: < 1, 0 > < 2, 1 > < 3, 2 > < 4, 3 > < 5, 2 > < 6, 5 > <
7, 6 > < 8, 2 >
Do Dai Cay Khung La: 37
    
```


KẾT LUẬN

Tóm lại, thông qua bài tìm hiểu này giúp chúng em hiểu sâu sắc hơn về các thuật toán trên đồ thị mà cụ thể là tìm đường đi ngắn nhất giữa hai đỉnh thông qua thuật toán Dijkstra và thuật toán tìm cây khung nhỏ nhất của Prim.

Một lần nữa, xin chân thành cảm ơn thầy Hồ Hữu Linh đã giúp đỡ em trong bài tìm hiểu này. Tuy nhiên, do trình độ do trình độ có hạn nên không tránh khỏi sai sót trong quá trình hoàn thành mong thầy và các bạn giúp đỡ nhiều hơn....

TÀI LIỆU THAM KHẢO

1. Bài giảng Phân tích thiết kế và giải thuật của thầy Hồ Hữu Linh.
2. Các bài tham khảo từ mạng Internet.

*****Nhận xét, hướng phát triển, ý kiến đánh giá của thầy hướng dẫn*****

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

LỜI NÓI ĐẦU	Error! Bookmark not defined.
GIẢI THUẬT_LƯU ĐỒ THUẬT TOÁN DIJKSTRA	Error!
	Bookmark not defined.2
1. Phân tích.	2
2 Giải thuật tìm đường đi ngắn nhất giữa một cặp đỉnh.....	Error! Bookmark not defined.2
3 Giải thuật Dijkstra.....	2
3.1 Nội dung.....	2
3.2 Lưu đồ thuật toán Dijkstra	4
3.3 Bảng dữ liệu chạy thô.	5
4. Cài đặt chương trình	6
5. Demo chương trình	8
THUẬT TOÁN PRIM	10
1. Thuật toán Prim	10
2. Thực thi giải thuật Prim	11
3. Cài đặt chương trình	13
3. Demo thuật toán Prim	16
KẾT LUẬN	17
TÀI LIỆU THAM KHẢO	17