CHƯƠNG 6: CÂY BAO TRÙM TỐI TIỂU

GVGD: Ths NGUYỄN CHÍ THANH
Email: thanh.mt@ou.edu.com

MỤC TIÊU

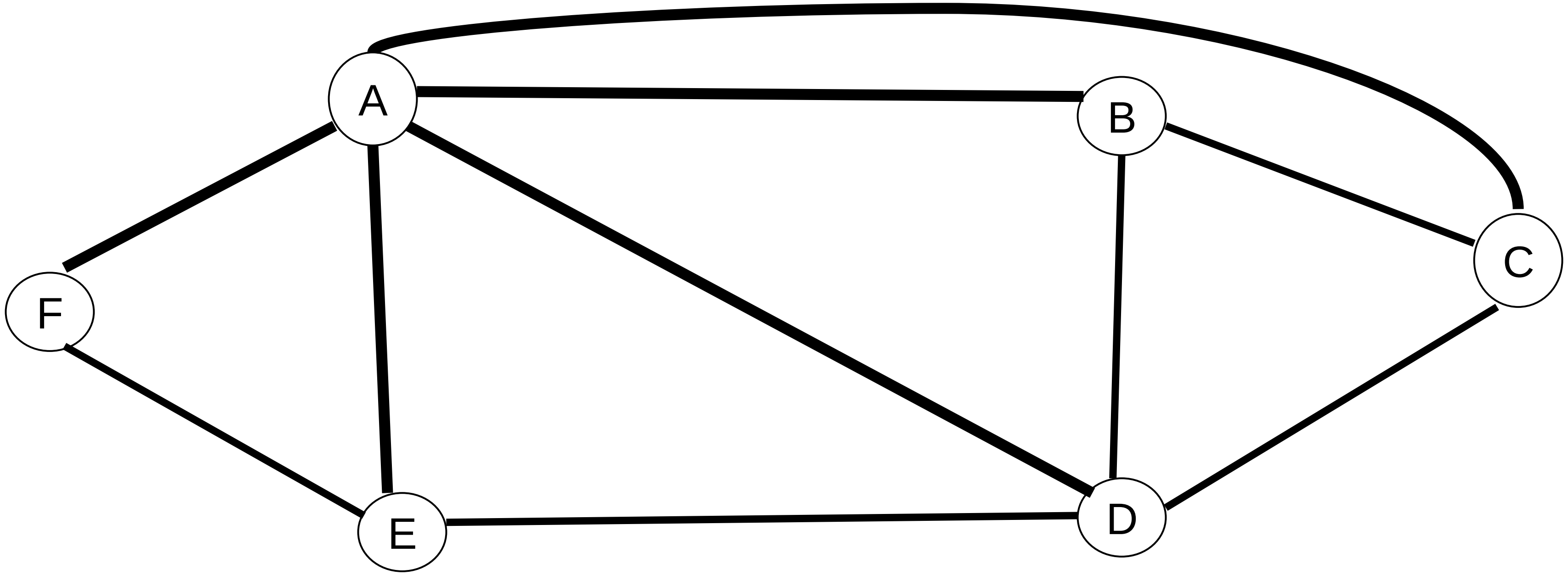
- Hiểu được khái niệm "Cây bao trùm".
- Hiểu được khái niệm "Cây bao trùm tối tiểu".
- Hiểu và cài đặt được thuật toán Prim
- Hiểu và cài đặt được thuật toán Kruskal

NỘI DUNG

- Cây bao trùm
- Cây bao trùm tối tiểu
- Bài toán tìm cây bao trùm tối tiểu
- Thuật toán Prim
- Thuật toán Kruskal

6.1. CÂY BAO TRÙM (Spanning Tree)

Ví dụ 6.1 Cho đồ thị G=(V, E) như bên dưới

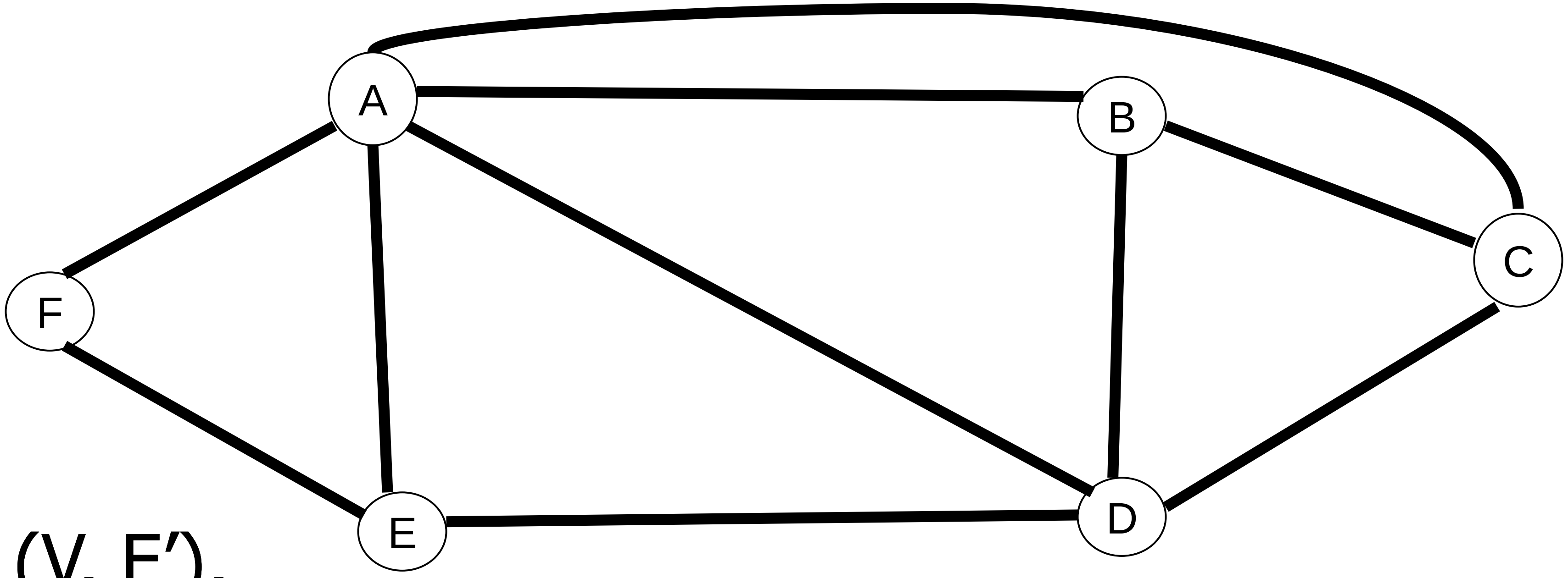


ĐỊNH NGHĨA CÂY BAO TRÙM

- Cho đồ thị liên thông G = (V, E), V là tập đỉnh, E là tập cạnh của G
- Nếu T = (V, E′), với E′ $\subseteq$ E, và T là một cây (có nghĩa T không có chu trình)

Thì ta nói T là cây bao trùm

Xét ví dụ 6.1



T1 = (V, E′),

V= {A, B, C, D, E, F};

E′ = {(A,C), (B,D), (C,D), (A,E), (E,F)} $\subseteq$ E;

NHẬN XÉT

- Trên một đồ thị G (liên thông) có thể có nhiều cây bao trùm, gọi tập các cây khung trên G là Sp(G).
- Ta có thể tìm cây bao trùm bằng các thuật giải BFS, DFS (Duyệt hết tất cả các đỉnh, mỗi lần duyệt đỉnh u nạp đỉnh vào trong T (với điều kiện T $\leftarrow$ {u}, T không tạo ra chu trình))
- Nếu giả sử G là đồ thi có trọng số, thì *cây T cũng là cây bao trùm có trọng số*.

6.2. CÂY BAO TRÙM TỐI TIỂU

(Minimum Spanning Tree)

ĐỊNH NGHĨA CÂY BAO TRÙM TỐI TIỂU

- Cho G = (V, E) là một đồ thị có trọng số,
- T là cây bao trùm tối tiểu khi:

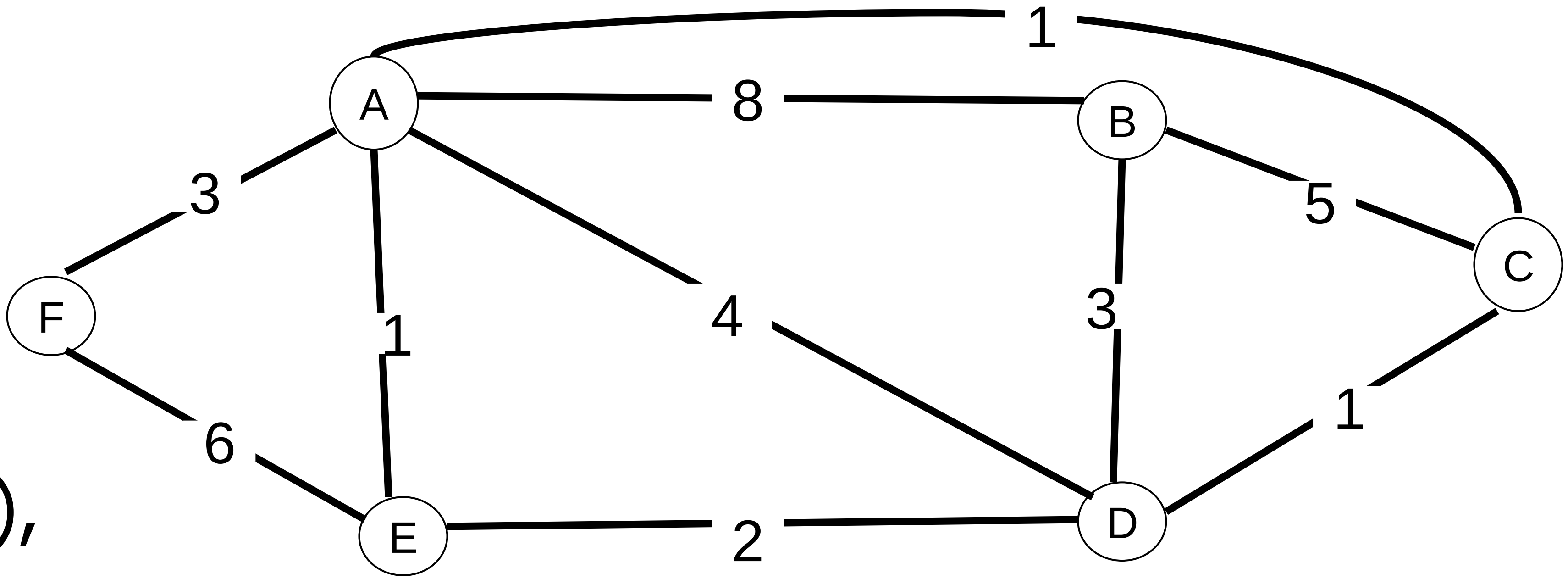
$$w(T) = \min\{w(T) \ / \ T \in Sp(G)\}$$

w(T): tổng trọng số của các cạnh trên cây T;

Sp(G): là tập tất cả cây bao trùm trên G;

Cây bao trùm tối tiểu là 1 cây bao trùm, có tổng trọng số là *tối tiểu* trên tập các cây khung Sp(G);

Xét lại ví dụ Cho đồ thị vô hướng có trọng số G=(V, E) như bên dưới



T = (V, E′),

V= {A, B, C, D, E, F};

E′ = {(A,E,1), (A,C,1), (C, D,1), (B,D,3), (A,F,4)} ⊆ E;

w(T) = 9

?? TÌM CÂY BAO TRÙM TỐI TIỂU

BÀI TOÁN TÌM CÂY BAO TRÙM TỐI TIỂU

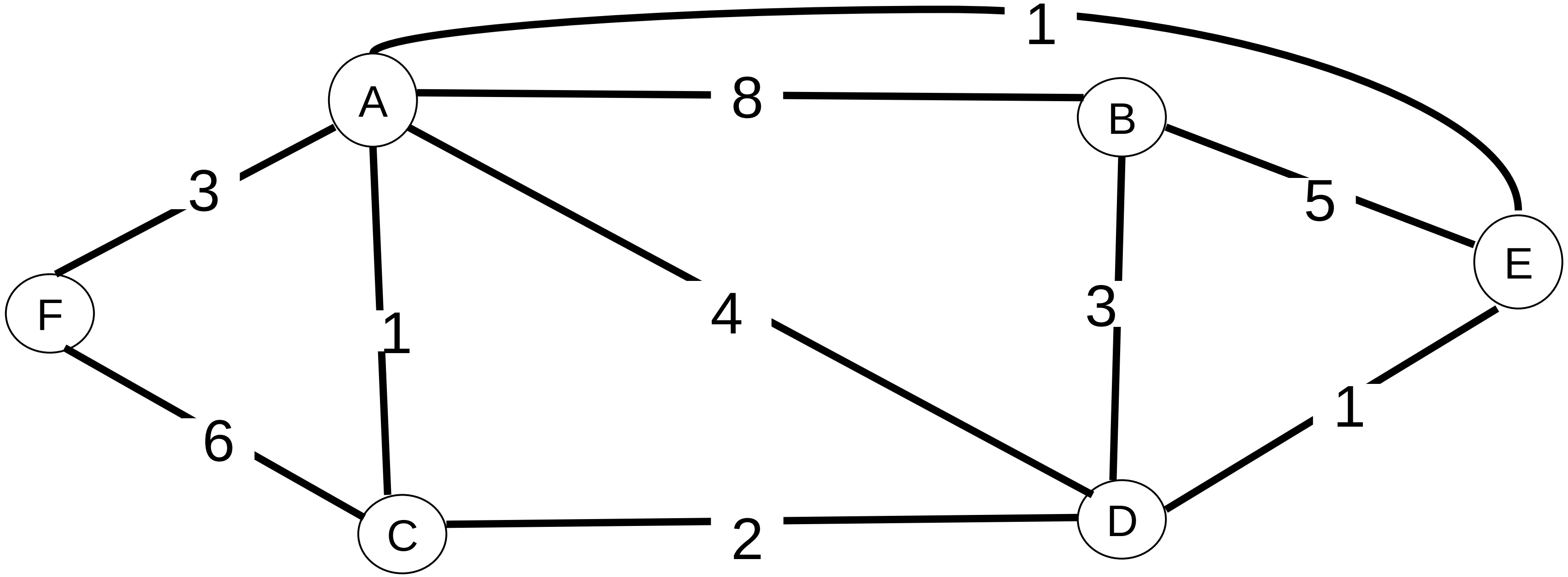
- Đầu vào: G = (V, E) là một đồ thị liên thông vô hướng có trọng số
- Đầu ra: T (cây bao trùm tối tiểu)

6.3. THUẬT GIẢI PRIM

Ý TƯỞNG

- Bước 0: bắt đầu từ một đỉnh *u* bất kì, và gọi *u* là đỉnh đang xét
- Bước 1: tìm tất cả các đỉnh v kề đỉnh đang xét, cho các cạnh này vào tập cạnh chuẩn bị xét Etemp;
- Bước 2: từ Etemp lấy ra một cạnh *e*, sao cho:
  - $\forall\ e_i \in$ Etemp/{*e*}, w(*e*) $\leq$ w(e_i); (*w(e) là trọng số của cạnh e*)
  - **Edges**(T) $\cup$ {*e*} $\Rightarrow$ T không tạo ra chu trình;
- Bước 3: Nếu không lấy được *e* nào hoặc **Vertices**(T) = V thì dừng (T là cây khung tối tiểu), ngược lại thì gọi *u* $\in$ *e*, u $\notin$ **Vertices**(T) là đỉnh đang xét; quay lại *bước 1*.

Minh họa ý tưởng giả sử bắt đầu từ đỉnh A



Đỉnh đang xét A C E D F

Tập E = {(A, B, 8), (A, D, 4), (A, F, 3), (C, F, 6), (C, D, 2), (E, B, 5)}{(D, B, 3)}

Cây T = {(A, C, 1), (A, E, 1), (E, D, 1), (A, F, 3), (D, B, 3)} **w(T) = 9**

CÀI ĐẶT THUẬT GIẢI PRIM

KHAI BÁO CÁC BIẾN TRONG CHƯƠNG TRÌNH

```
// khai bao ma tran bang
mang hai chieu
# define MAX 20
int a[MAX][MAX];
int n; // so dinh cua do thi
```

```
// khai bao TapE
int E1[MAX];
int E2[MAX];
int wE[MAX];
int nE=0; // so phan tu tap E
```

```
// khai bao TapE
int T1[MAX];
int T2[MAX];
int wT[MAX];
int nT=0; // so phan tap T
```

THỦ TỤC PRIM

```
void prim(int s) // s là đỉnh bắt đầu
{
	int u=s, min, i, d1 d2;
	while(nT<n-1)
	{
		for(int v=0;v<n;v++)
		   if(a[u][v]!=0)
		      if (TonTai(v, T2, nT)==0)
		      {
		         E1[nE]=u;   E2[nE]=v;
		         wE[nE]=a[u][v];  nE++;
		      }
		for(i=0;i<nE;i++)
		   if (TonTai(E2[i], T2, nT)==0)
		   {
		      min=wE[i]; d1=E1[i];
		      d2=E2[i]; break;
		   }
		for(;i<nE;i++)
		   if(TonTai(E2[i], T2, nT)==0)
		      if(min>wE[i])
		      {     min=wE[i]; d1=E1[i];
		            d2=E2[i];
		      }
		T1[nT]=d1; T2[nT]=d2;
		wT[nT]=a[d1][d2];
		a[d1][d2]=0; a[d2][d1]=0;
		nT++;
		XoaCanhE(d1, d2);
		u=d2;
	}
}
```

```
int TonTai(int d, int D[], int nD)
{
	for(int i=0;i<nD;i++)
		if(D[i]==d)
			return 1;
	return 0;
}
```

```
void output()
{
	int tong=0;
	for(int i=0;i<nT;i++)
	{
		cout<<endl<<"("<<T1[i]<<","<<T2[i]
	<<") = "<<wT[i];
		tong+=wT[i];
	}
	cout<<"\n  Tong = "<<tong;
}
```

```
void XoaViTriE(int i)
{
		for(int j=i;j<nE;j++)
		{
			E1[j]=E1[j+1];
			E2[j]=E2[j+1];
			wE[j]=wE[j+1];
		}
		nE--;
}
```

```
void XoaCanhE(int u, int v)
{
	for(int i=0;i<nE;i++)
	if(E1[i]==u&&E2[i]==v)
	{
		XoaViTriE(i);
		break;
	}
}
```

ĐỘ PHỨC TẠP THUẬT GIẢI PRIM

- Nếu G = (V, E) là đồ thị liên thông có trọng số; n là số đỉnh của G, và m là số cạnh của G: n = |V|, m = |E|, thì ta có độ **phức tạp của thuật giải Prim là:**

$$\boxed{\approx O(n * \max(n,m))}$$

6.4. THUẬT GIẢI KRUSKAL

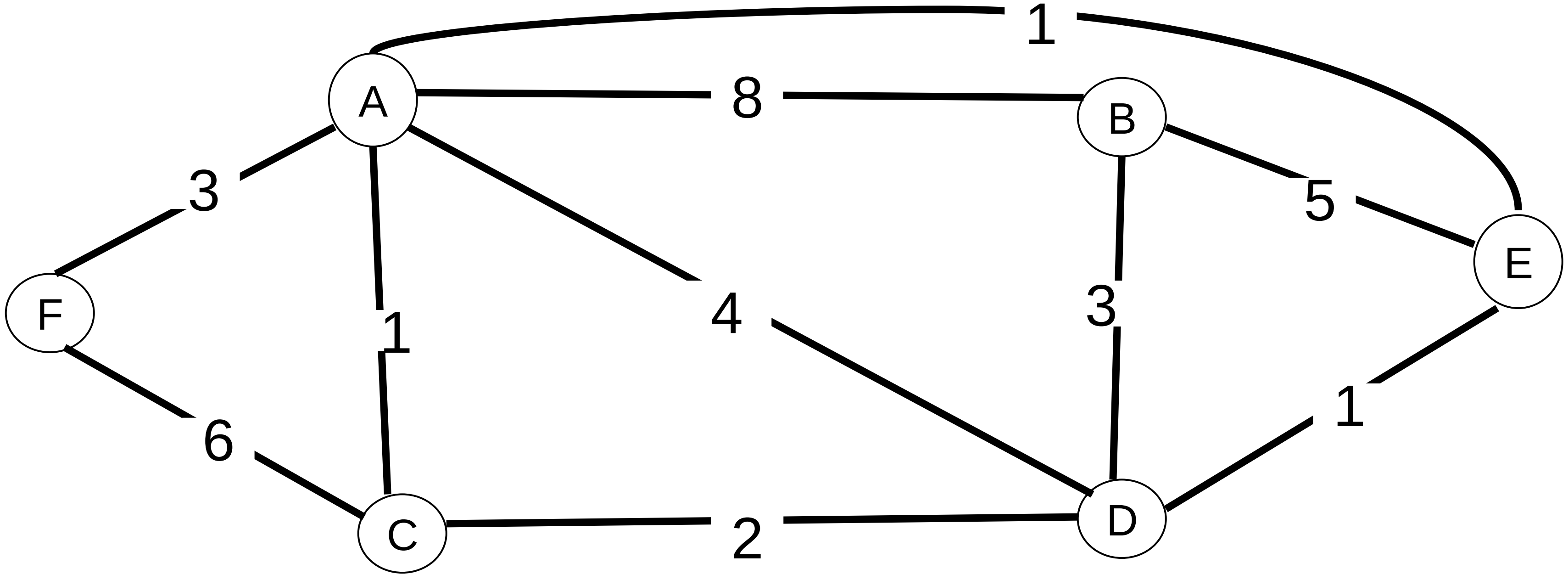
Ý TƯỞNG THUẬT GIẢI KRUSKAL

- Bước 1: từ E lấy ra một cạnh *e*, sao cho:
  - $\forall e_i \in E$, w(*e*) $\leq$ w(e_i) (w(*e*) là trọng số của cạnh *e*)
  - **Edges**(T) $\cup$ {*e*} $\Rightarrow$ T không tạo ra chu trình;
- Bước 2: Nếu không lấy được *e* nào hoặc V = **Vertices**(T) thì dừng (T là cây khung tối tiểu), ngược lại thì quay lại bước 1;

KRUSKAL CẢI TIẾN

- Bước 1: Sắp xếp E (tăng theo *trọng số* của cạnh)
- Bước 2: Lấy từ E ra một cạnh *e*, sao cho:
  - **Edges**(T) $\cup$ {*e*} $\Rightarrow$ T không tạo ra chu trình;
- Bước 3: V = **Vertices**(T) thì dừng (T là cây khung tối tiểu), ngược lại thì quay lại bước 2;

Minh họa ý tưởng giả sử bắt đầu từ đỉnh



Tập E = {(A,B,8), (A,D,4), (A,F,3), (B,D,3),

(B,E,5), (C,D,2), (C,F,6), (D,E,1)}

Cây T = { (A,C,1) ,(A,E,1),(E, D,1) ,(A,F,3) ,(B,D,3) }

CÀI ĐẶT THUẬT GIẢI KRUSKAL

KHAI BÁO CÁC BIẾN

```
// khai bao ma tran bang
mang hai chieu
# define MAX 20
int a[MAX][MAX];
int n; // so dinh cua do thi
```

```
// khai bao TapE
int E1[MAX];
int E2[MAX];
int wE[MAX];
int nE=0; // so phan tu tap E
```

```
// khai bao TapE
int T1[MAX];
int T2[MAX];
int wT[MAX];
int nT=0; // so phan tap T
```

THỦ TỤC KRUSKAL

```
void kruskal()
{
   for(int i=0;i<nE;i++)
   {
      if(TonTai(E1[i], T1, nT)==1  && TonTai(E2[i], T2, nT)==1)
           continue;
      if (TonTai(E1[i], T2, nT)==1  && TonTai(E2[i], T1, nT)==1)
           continue;
      T1[nT]=E1[i];
      T2[nT]=E2[i];
      wT[nT]=wE[i];
      nT++;
      if(nT==n-1)
            break;
   }
}
```

```
void taoE()
{
   for(int i=0;i<n;i++)
      for(int j=0;j<n;j++)
            if(a[i][j]!=0)
            {
                  E1[nE]=i;
                  E2[nE]=j;
                  wE[nE]=a[i][j];
                  a[i][j]=0;
                  a[j][i]=0;
                  nE++;
            }
}
```

```
void SapXepE()
{
   for(int i=0;i<nE-1;i++)
      for(int j=i+1;j<nE;j++)
            if(wE[i]>wE[j])
            {
               swap(wE[i],wE[j]);
               swap(E1[i],E1[j]);
               swap(E2[i],E2[j]);
            }
}
```

ĐỘ PHỨC TẠP THUẬT GIẢI

- Nếu G = (V, E) là đồ thị liên thông có trọng số; n là số đỉnh của G, và m là số cạnh của G: n = |V|, m = |E|, thì ta có độ **phức tạp của thuật giải Kruskal là:**

$$\boxed{\approx O(\max(n^2, m^2))}$$

TÀI LIỆU THAM KHẢO

1. **Thomas H.Cormen**, **Charles E.Leiserson**, **Ronald L. Rivest**, **Cliffrod Stein**, (Chapter 23) *Introduction to Algorithms*, Third Edition, 2009.

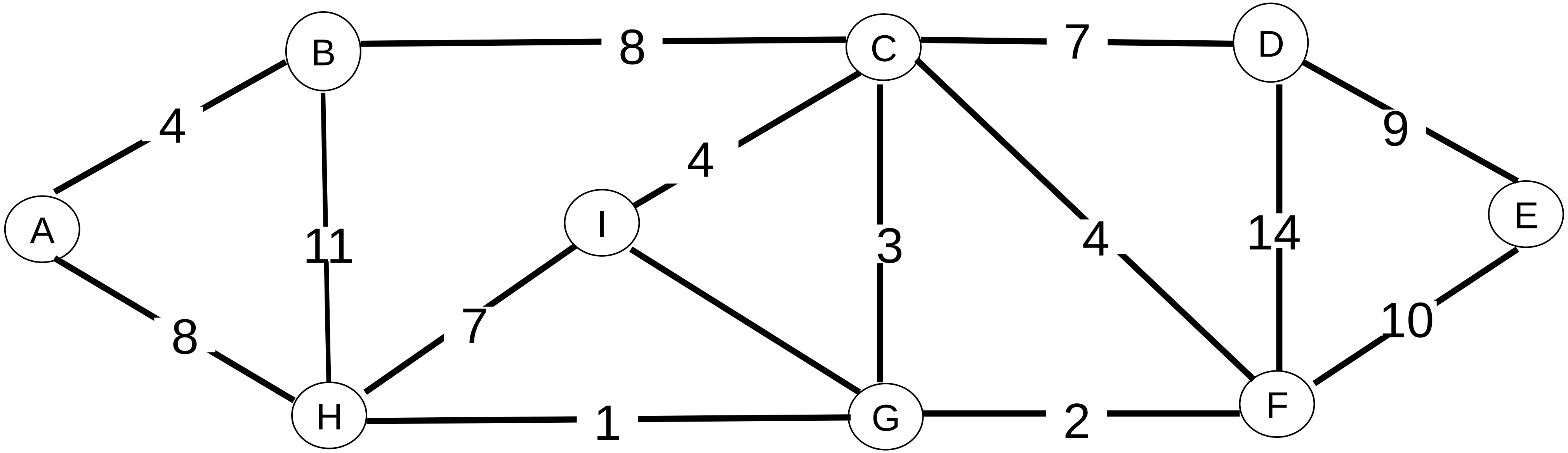
2. **Judith l. Gersting,** (Chapter 6&7) *mathematical structures for computer science*, 2014.

Bài 1: Trình bày lại ý tưởng của thuật giải Prim

Bài 2: Trình bày lại ý tưởng của thuật giải Kruskal

Bài 3: Hãy cho biết sự khác biệt giữa hai ý tưởng của thuật giải Prim và thuật giải Kruskal

Đồ thị G = (V, E)



Bài 4: Viết chương trình thực hiện tìm cây khung tối tiểu cho đồ thị G bằng thuật giải Prim. (*dựa trên phương pháp biểu diễn đồ thị bằng ma trận kề*)

Bài 5: Viết chương trình thực hiện tìm cây khung tối tiểu của G bằng thuật giải Kruskal. (*dựa trên phương pháp biểu diễn đồ thị bằng ma trận kề*)

Bài 6: Viết chương trình thực hiện tìm cây khung tối tiểu của G bằng thuật giải Kruskal cải tiến. (*dựa trên phương pháp biểu diễn đồ thị bằng ma trận kề*)

Bài 7: Viết chương trình thực hiện tìm cây khung tối tiểu cho đồ thị G bằng thuật giải Prim. (*dựa trên phương pháp biểu diễn đồ thị bằng danh sách kề*)

Bài 8: Viết chương trình thực hiện tìm cây khung tối tiểu của G bằng thuật giải Kruskal. (*dựa trên phương pháp biểu diễn đồ thị bằng danh sách kề*)

Bài 9: Viết chương trình thực hiện tìm cây khung tối tiểu của G bằng thuật giải Kruskal cải tiến. (*dựa trên phương pháp biểu diễn đồ thị bằng danh sách kề*)

BÀI TẬP LÀM THÊM

Bài 10: Thực hiện biểu diễn G (bằng ma trận kề) lên trên máy tính và đặt tên là do_thi_1.txt (dạng file TEXT) sau đó thực hiện viết chương trình tìm cây khung tối tiểu cho đồ thị G:

10.1. Bằng thuật giải Prim

10.2. Bằng thuật giải Kruskal

10.2. Bằng thuật giải Kruskal cải tiến