TOÁN RỜI RẠC 2

CHƯƠNG 5

Giảng viên: Vũ Văn Thỏa

CHƯƠNG 5:
CÂY VÀ CÂY KHUNG CỦA ĐỒ THỊ

- Định nghĩa và tính chất
- Xây dựng cây khung của đồ thị
- Bài toán tìm cây khung nhỏ nhất

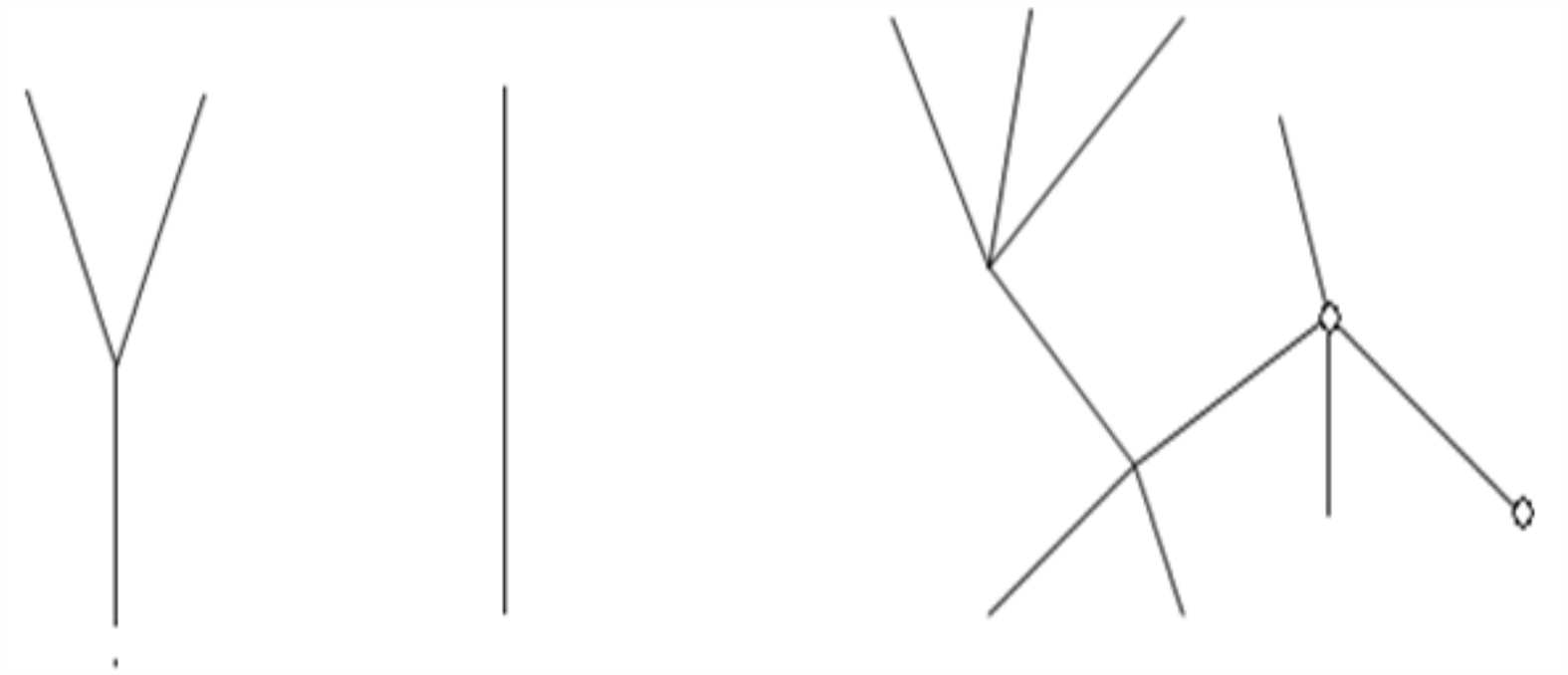
5.1 Định nghĩa và tính chất

- Định nghĩa cây
- Các tính chất

Cây tự do

- Cây tự do (không gốc) T là đồ thị vô hướng liên thông và không có chu trình đơn.
- Đồ thị F là một rừng $\Leftrightarrow$ mỗi thành phần liên thông của F là một cây.
- *Đồ thị F là một rừng $\Leftrightarrow$ F không có chu trình đơn*

Ví dụ 1: Cây tự do



Điều kiện để đồ thị là cây

Định lý 1: T= (V, E) là một đồ thị vô hướng có n đỉnh. Các mệnh đề sau là tương đương:

(1) T là cây;

(2) T không chứa chu trình và có n-1 cạnh;

(3) T liên thông và có n-1 cạnh;

(4) T liên thông và mỗi cạnh của T là cầu ;

(5) Hai đỉnh bất kỳ của T có duy nhất một đường đi nối đến nhau ;

(6) T không chứa chu trình nhưng nếu thêm 1 cạnh thì thu được đúng 1 chu trình.

Cây có gốc

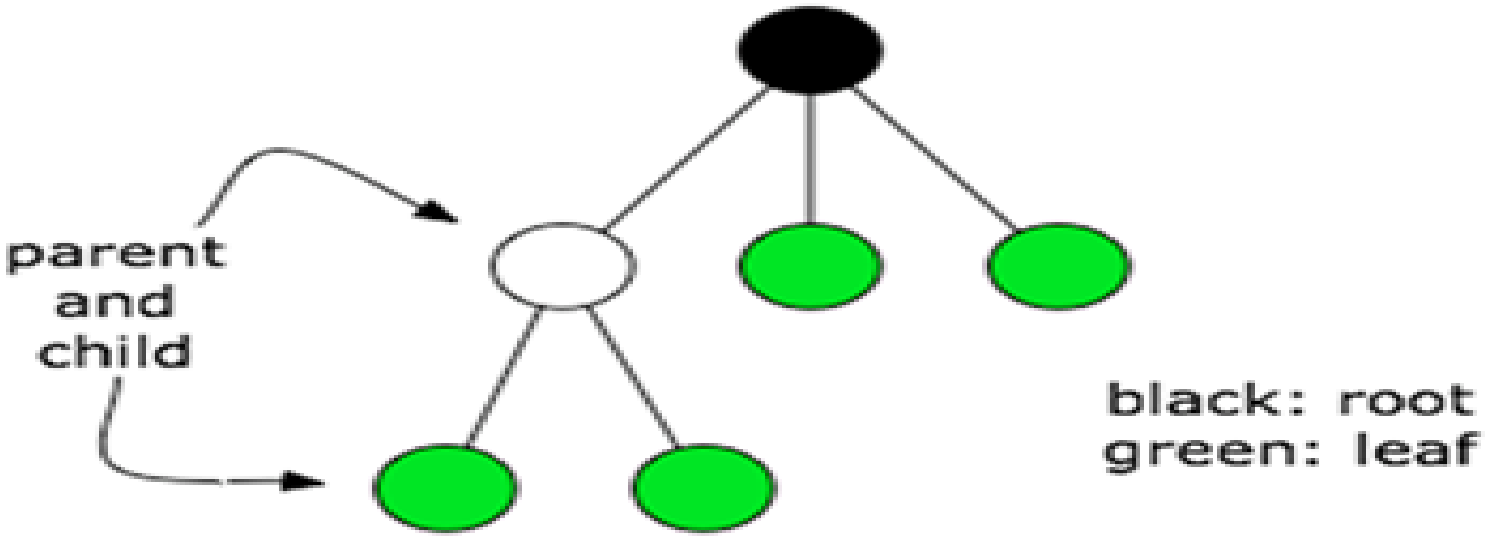
Định nghĩa 1: Cây là tập hợp hữu hạn các nút thỏa mãn:

- Có một nút gọi là gốc
- Có quan hệ phân cấp "cha-con" giữa các nút.

Định nghĩa 2 (đệ qui):

- Nếu T chỉ gồm 1 nút $\Rightarrow$ T là một cây với gốc là chính nút đó
- Nếu $T_1, \ldots, T_n$ ($n \geq 1$) là các cây có gốc tương ứng $r_1, \ldots, r_n$ $\Rightarrow$ T là cây với gốc r được tạo thành bằng cách cho r thành nút cha của các nút $r_1, \ldots, r_n$.

Ví dụ 2: Cây có gốc



Mô hình cây trong thực tế

- Mục lục của một cuốn sách
- Cấu trúc thư mục trên đĩa máy tính.
- Dùng cây để biểu diễn biểu thức số học.

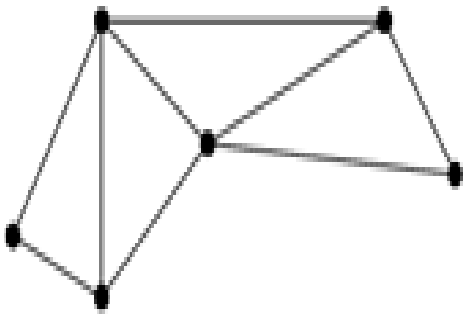
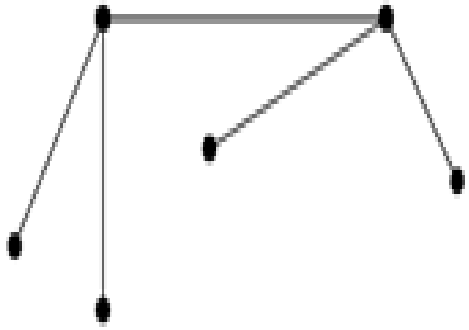
Định nghĩa cây khung

- **Định nghĩa 1**:

Cho G là một đơn đồ thị vô hướng.

Một cây T gọi là *cây khung* của G $\Leftrightarrow$ T là đồ thị con của G và chứa tất cả các đỉnh của G.

Ví dụ 3: Cây khung của đồ thị

 Đồ thị G	 Một cây khung T của G

Điều kiện tồn tại cây khung

Định lý 2:

Một đồ thị vô hướng G có cây khung $\Leftrightarrow$ G liên thông.

5.2 Bài toán tìm cây khung

1) Đặt bài toán:

Input: Đồ thị vô hướng G gồm n đỉnh cho bởi danh sách kề;

Đỉnh u;

Output: Cây khung T của G bắt đầu từ đỉnh u;

2. Xây dựng cây khung bằng DFS

- **Thuật toán TreeDfs(u)**:

Bước 1: Thực hiện DFS(u);

Bước 2 (*Xuất kết quả*):

- Nếu số đỉnh được duyệt bằng n thì xuất kết quả T;

- Nếu số đỉnh được duyệt nhỏ hơn n thì xuất thông báo: “Không có cây khung”;

- **Độ phức tạp tính toán:** Giải thuật tìm cây khung bằng DFS có độ phức tạp O(n).

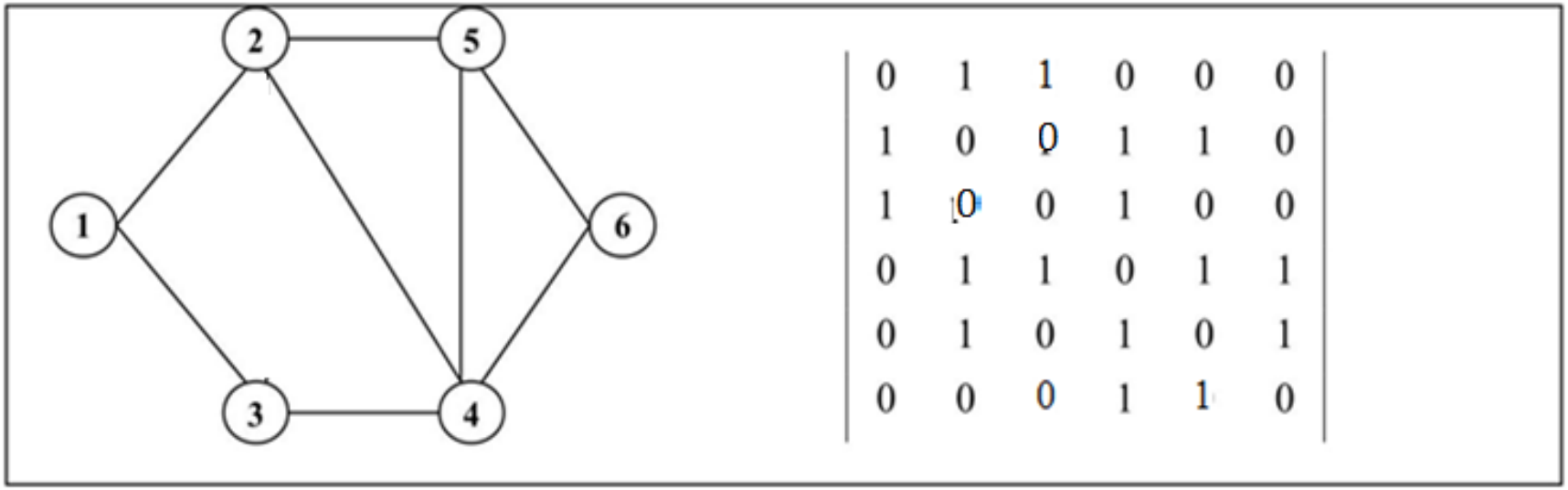
Cài đặt:

```
// G cho bởi ma trận kề a[i][j]
int a[100][100], n, u, vs[100], e[100];
void DfsDequy(int u) { int v;
    vs[u]= 1;
    for (v= 1; v<=n; v++)
        if (vs[v]==0 && a[u][v]==1){
            e[v]= u;
            DfsDequy(v);   }
    }
```

Cài đặt:

```
void TreeDfs(int u) { int v;
    for (v= 1; v<=n; v++) vs[v] = 0;
    DfsDequy(u);
    int dem= 0;
          for (v= 1; v<=n; v++) if (vs[v] ==1) dem++;
     if (dem == n) {
         for (v= 1; v<=n; v++)
                if (e[v] != 0) cout << v << ” ” << e[v] << endl;
            } else
       cout << ”Khong co Cay khung”;
    }
```

Ví dụ 4: Tìm cây khung của G bắt đầu tại u= 1



- **Sử dụng DFS:** n = 6

Dfs(1)= {1(0); 2(1); 4(2); 3(4); 5(4); 6(5)} = V

$\Rightarrow$ Cây khung tìm được:

T= {(1,2), (2,4), (3,4), (4,5), (5,6)}

3. Xây dựng cây khung bằng BFS

- **Thuật toán TreeBfs(u)**:

Bước 1: Thực hiện BFS(u);

Bước 2 (*Xuất kết quả*):

- Nếu số đỉnh được duyệt bằng n thì xuất kết quả T;

- Nếu số đỉnh được duyệt nhỏ hơn n thì xuất thông báo: “Không có cây khung”;

- **Độ phức tạp tính toán:** Giải thuật tìm cây khung bằng BFS có độ phức tạp O(n).

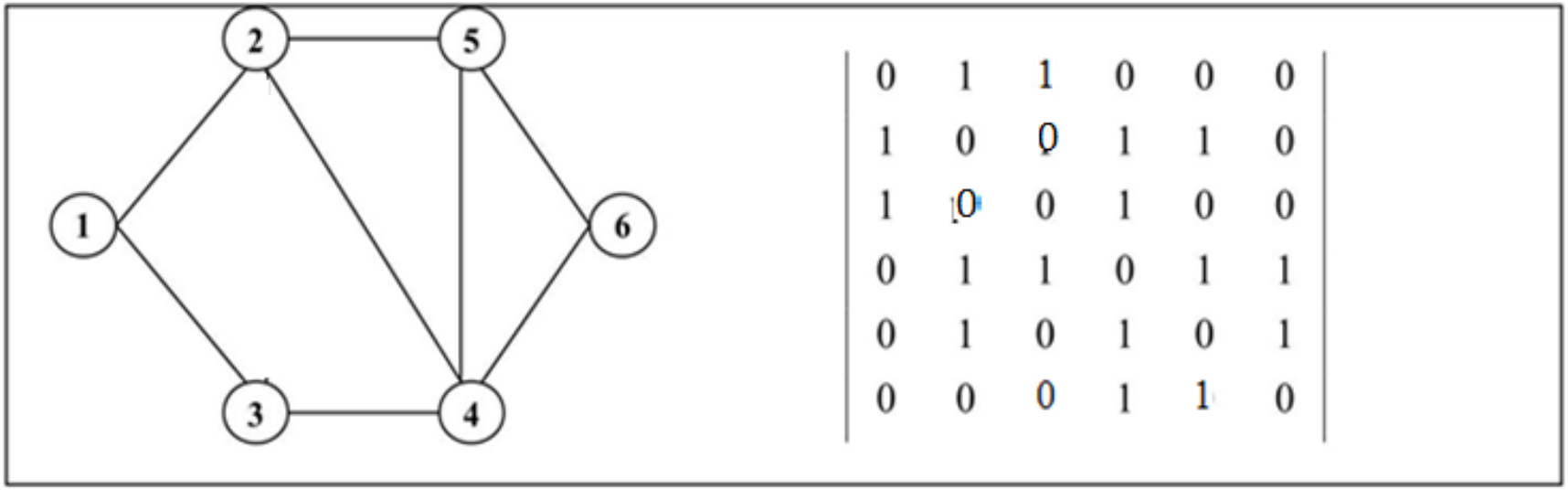
Cài đặt:

```
// G cho bởi ma trận kề a[i][j]
int a[100][100], n, u, vs[100], e[100], q[100];
void Bfs(int u) { int v, dq= 1, cq= 0;
      vs[u]= 1;  e[u]= 0; cq++; q[cq]= u;
     while (dq <= cq){ int i = q[dq]; dq++;
      for (v= 1; v<=n; v++)
          if (vs[v]==0 && a[i][v]==1){
                e[v]= u; vs[v]= 1; cq++; q[cq]= v;
                 }
            }
   }
```

Cài đặt:

```
void TreeBfs(int u) { int v;
    for (v= 1; v<=n; v++) vs[v] = 0;
       Bfs(u);
    int dem= 0;
         for (v= 1; v<=n; v++) if (vs[v] ==1) dem++;
    if (dem == n) {
       for (v= 1; v<=n; v++)
        if (e[v] != 0) cout << v << ” ” << e[v] << endl;
                } else
            cout << ”Khong co Cay khung”;
    }
```

Ví dụ 5: Tìm cây khung của G bắt đầu tại u= 1



- **Sử dụng BFS:** n = 6

  Bfs(1)= {1(0); 2(1), 3(1); 4(2), 5(2); 6(4)} = V

$\Rightarrow$ Cây khung tìm được:

T= {(1,2), (1,3), (2,4), (2,5), (4,6)}

Ghi chú

- Cây khung T tìm kiếm theo chiều rộng bắt đầu từ u gồm các đường đi ngắn nhất xuất phát từ u đến các đỉnh khác.

- Đồ thị đầy đủ K_n có n^{n-2} cây khung khác nhau.

5.3 Bài toán tìm cây khung nhỏ nhất

- Định nghĩa và điều kiện
- Thuật toán tìm cây khung nhỏ nhất

Định nghĩa và điều kiện

- **1. Cây khung nhỏ nhất**

Cho đồ thị vô hướng có trọng số G = (V, E).

Gọi T là một cây khung của G. Trọng số WT của T là tổng trọng số các cạnh thuộc cây.

Cây khung T là cây khung *nhỏ nhất* $\Leftrightarrow$ WT có giá trị nhỏ nhất.

- **2. Điều kiện:**

G vô hướng có cây khung nhỏ nhất $\Leftrightarrow$ G liên thông;

Thuật toán tìm cây khung nhỏ nhất

- Thuật toán Prim
- Thuật toán Kruskal

Thuật toán Prim:

Input: Đồ thị G = (V, E) gồm n đỉnh cho bởi ma trận trọng số a[i][j];
Đỉnh s $\in$ G;

Output: Cây khung bắt đầu tại s nhỏ nhất T và WT;

Khởi tạo: T= $\varnothing$; VT = {s}; WT= 0;

while (V\VT $\neq \varnothing$) {

<Tìm e = (u, v) có trọng số nhỏ nhất, u $\in$ VT, v $\in$ V\VT>;

if (Tìm được e) {T = T $\cup$ {e};

WT= WT + trọng số của e;

VT = VT $\cup$ {v};

}

else return (G không có cây khung); }

return (T và WT);

Ghi chú

- Để thuận tiện cho cài đặt thuật toán Prim, Ký hiệu:

  vs[v] = 1 nếu v $\in$ VT; vs[v] = 0 nếu v $\notin$ VT (v $\in$ V\VT);

  d[v] là trọng số của cạnh nhỏ nhất e= (u,v) với u $\in$ VT và v $\in$ V\VT (Tạm thời);

  e[v] = u $\Leftrightarrow$ Cạnh e = (u, v) $\in$ T (Tạm thời);

- **Thuật toán Prim**:

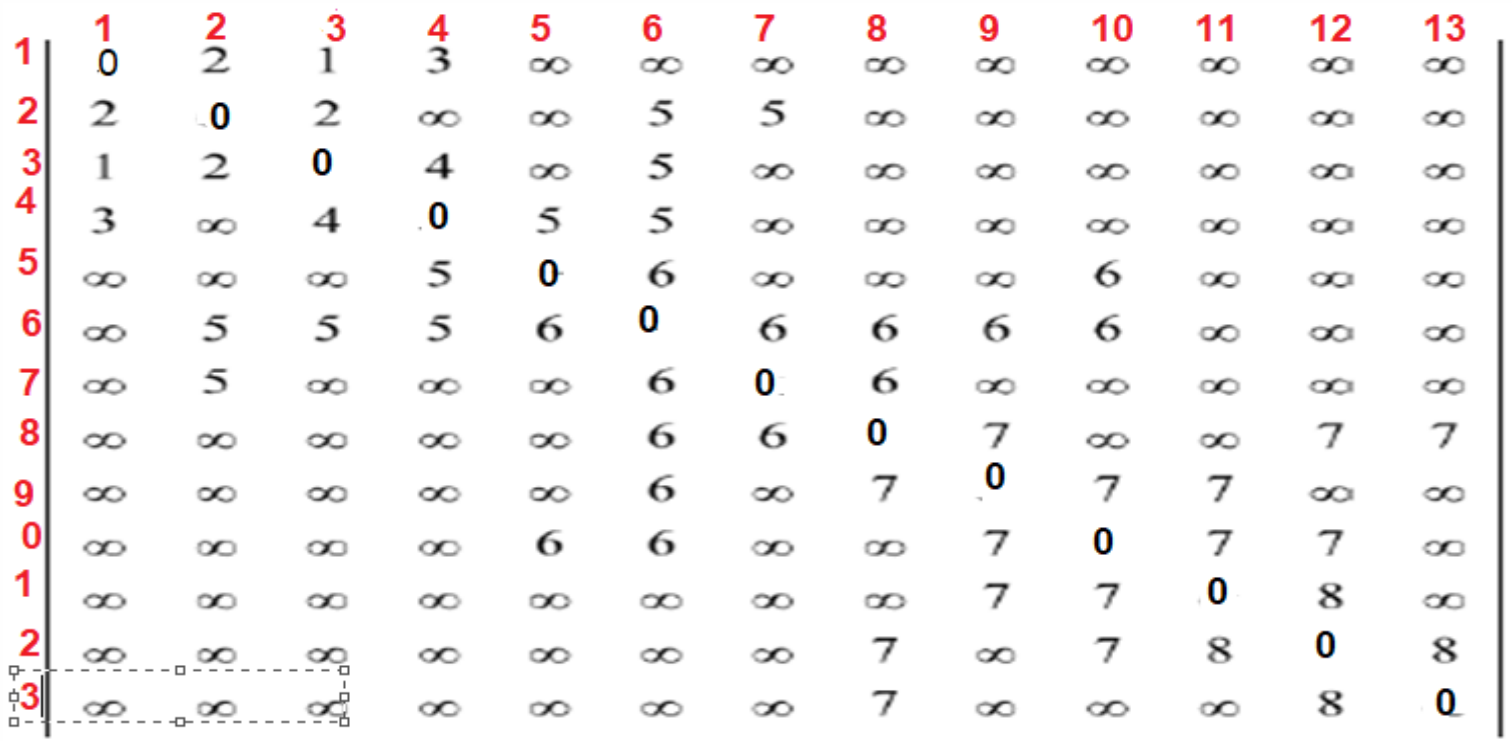
- Khởi tạo: d[v] = a[s][v]; e[v] = s; vs[s] = 1; e[s] = 0;
- Tại mỗi bước tìm u sao cho d[u] = Min{d[v] | vs[v] = 0}
- Nếu tìm được u cần cập nhật d[v]: nếu vs[v] = 0 và d[v] > a[u][v] thì thay thế d[v] = a[u][v]; e[v] = u;
- Nếu không tìm được u thì thông báo “Khong co cay khung”.

Cài đặt thuật toán Prim

```
int n, a[100][100];
int vs[100], d[100], e[100];
void Prim(int s) {
for (int v= 1; v<= n, v++) {
    vs[v]= 0; d[v]= a[s][v]; e[v]= s};
     vs[s]= 1; d[s]= 0; e[s]= 0;
  int wt= 0, dem = 1;
while (dem < n) {
int u = 0;
int min = 30000;
for (v = 1; v<= n; v++)
  if (vs[v]==0 && d[v] < min) {
              min= d[v]; u= v;
}
if (u==0) {
     cout << “Khong co cay khung”;
return;}
    vs[u]= 1; wt= wt + a[u][e[u]];
for (v= 1; v<= n; v++)
  if (vs[v]==0 && d[v] > a[u][v])
    {d[v]= a[u][v]; e[v]= u; }
}
   cout << wt << endl;
for (v= 1; v<= n; v++)
   if (e[v] !=0)
    cout << v << “ “ <<e[v] << endl;
return;
}
```

Ví dụ 6: Tìm cây khung nhỏ nhất

- Cho G:



	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	2	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	2	0	2	∞	∞	5	5	∞	∞	∞	∞	∞	∞
3	1	2	0	4	∞	5	∞	∞	∞	∞	∞	∞	∞
4	3	∞	4	0	5	5	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	5	0	6	∞	∞	∞	6	∞	∞	∞
6	∞	5	5	5	6	0	6	6	6	6	∞	∞	∞
7	∞	5	∞	∞	∞	6	0	6	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	6	6	0	7	∞	∞	7	7
9	∞	∞	∞	∞	∞	6	∞	7	0	7	7	∞	∞
10	∞	∞	∞	∞	6	6	∞	∞	7	0	7	7	∞
11	∞	∞	∞	∞	∞	∞	∞	∞	7	7	0	8	∞
12	∞	∞	∞	∞	∞	∞	∞	7	∞	7	8	0	8
13	∞	∞	∞	∞	∞	∞	∞	7	∞	∞	∞	8	0

- Tìm cây khung nhỏ nhất bắt đầu tại s= 1 sử dụng Prim.

Giải Có n= 13

Lập bảng:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	2	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	2	0	2	∞	∞	5	5	∞	∞	∞	∞	∞	∞
3	1	2	0	4	∞	5	∞	∞	∞	∞	∞	∞	∞
4	3	∞	4	0	5	5	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	5	0	6	∞	∞	∞	6	∞	∞	∞
6	∞	5	5	5	6	0	6	6	6	6	∞	∞	∞

Bước	d[1] |e[1]	d[2] |e[2]	d[3] |e[3]	d[4] |e[4]	d[5] |e[5]	d[6] |e[6]	d[7] |e[7]	d[8] |e[8]	d[9] |e[9]	d[0] |e[0]	d[1] |e[]	d[2] |e[2]	d[3] |e[3]	T	Wt
1	**0|0**	2|1	1|1	3|1	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	∅	0
2		2|1	**1|1**	3|1	∝|1	5|3	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	(1,3)	1
3		**2|1**		3|1	∝|1	5|3	5|2	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	(1,2)	3
4				**3|1**	5|4	5|3	5|2	∝|1	∝|1	∝|1	∝|1	∝|1	∝|1	(1,4)	6
5					**5|4**	5|3	5|2	∝|1	∝|1	6|5	∝|1	∝|1	∝|1	(4,5)	11
6						**5|3**	5|2	6|6	6|6	6|5	∝|1	∝|1	∝|1	(3,6)	16

7	∞	5	∞	∞	∞	6	0	6	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	6	6	0	7	∞	∞	7	7
9	∞	∞	∞	∞	∞	6	∞	7	0	7	7	∞	∞
0	∞	∞	∞	∞	6	6	∞	∞	7	0	7	7	∞
1	∞	∞	∞	∞	∞	∞	∞	∞	7	7	0	8	∞
2	∞	∞	∞	∞	∞	∞	∞	7	∞	7	8	0	8
3	∞	∞	∞	∞	∞	∞	∞	7	∞	∞	∞	8	0

Bước	d[1] |e[1]	d[2] |e[2]	d[3] |e[3]	d[4] |e[4]	d[5] |e[5]	d[6] |e[6]	d[7] |e[7]	d[8] |e[8]	d[9] |e[9]	d[0] |e[0]	d[1] |e[]	d[2] |e[2]	d[3] |e[3]	T	Wt
1	**0|0**	2|1	1|1	3|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∅	0
2		2|1	**1|1**	3|1	∞|1	5|3	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,3)	1
3		**2|1**		3|1	∞|1	5|3	5|2	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,2)	3
4				**3|1**	5|4	5|3	5|2	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,4)	6
5					**5|4**	5|3	5|2	∞|1	∞|1	6|5	∞|1	∞|1	∞|1	(4,5)	11
6						**5|3**	5|2	6|6	6|6	6|5	∞|1	∞|1	∞|1	(3,6)	16
7							**5|2**	6|6	6|6	6|5	∞|1	∞|1	∞|1	(2,7)	21
8								**6|6**	6|6	6|5	∞|1	7|8	7|8	(6,8)	27
9									**6|6**	6|5	7}9	7|8	7|8	(6,9)	33
10										**6|5**	7}9	7|8	7|8	(5,10)	39
11											**7}9**	7|8	7|8	(9,11)	46
12												**7|8**	7|8	(8,12)	53
13													**7|8**	(8,13)	60

Kết quả

Bước	d[1] |e[1]	d[2] |e[2]	d[3] |e[3]	d[4] |e[4]	d[5] |e[5]	d[6] |e[6]	d[7] |e[7]	d[8] |e[8]	d[9] |e[9]	d[0] |e[0]	d[1] |e[]	d[2] |e[2]	d[3] |e[3]	T	Wt
1	**0|0**	2|1	1|1	3|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∅	0
2		2|1	**1|1**	3|1	∞|1	5|3	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,3)	1
3		**2|1**		3|1	∞|1	5|3	5|2	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,2)	3
4				**3|1**	5|4	5|3	5|2	∞|1	∞|1	∞|1	∞|1	∞|1	∞|1	(1,4)	6
5					**5|4**	5|3	5|2	∞|1	∞|1	6|5	∞|1	∞|1	∞|1	(4,5)	11
6						**5|3**	5|2	6|6	6|6	6|5	∞|1	∞|1	∞|1	(3,6)	16
7							**5|2**	6|6	6|6	6|5	∞|1	∞|1	∞|1	(2,7)	21
8								**6|6**	6|6	6|5	∞|1	7|8	7|8	(6,8)	27
9									**6|6**	6|5	7}9	7|8	7|8	(6,9)	33
10										**6|5**	7}9	7|8	7|8	(5,10)	39
11											**7}9**	7|8	7|8	(9,11)	46
12												**7|8**	7|8	(8,12)	53
13													**7|8**	(8,13)	60

Wt = 60; T = {(1,3), (1,2), (1,4), (4,5), (3,6), (2,7), (6,8), (6,9), (5,10), (9,11), (8,12), (8,13)}

Thuật toán Kruskal

Input: Đồ thị G = (V, E) gồm n đỉnh và m cạnh cho dưới dạng danh sách cạnh;

Output: Cây khung nhỏ nhất T và WT;

Thuật toán Kruskal

Khởi tạo:

Sắp xếp các cạnh theo thứ tự tăng của trọng số $e_1, \ldots, e_m$;

T= ∅; WT= 0; k = 0;

for (i=1; i<= m; i++){

if (T ∪ {e_i} không chứa chu trình) {

T = T ∪ {e_i}; WT= WT + trọng số của e_i;

k++;

if (k = n-1) return (T và WT); }

}

Return (G không có cây khung);

Cài đặt thuật toán Kruskal

```
int n, m, d[10000], c[10000], ts[10000];
int vs[100], t[100];
void Kruskal() {
 for (int i= 1; i<= m-1, i++)
    for (int j= i+1;j<= m, j++)
        if (ts[i] > ts[j]) {
    int tg = ts[i]; ts[i]= ts[j]; ts[j]= tg;
       tg = d[i]; d[i]= d[j]; d[j]= tg;
       tg = c[i]; c[i]= c[j]; c[j]= tg; };
      int wt= 0, k= 0;
    for (i= 1; i<= n; i++) vs[i]= 0;
       for (i= 1; i<= m; i++)
 if (!(vs[d[i]] !=0 && vs[d[i]] == vs[c[i]])){
      k++;  t[k]= i; wt= wt + ts[i];
       if (k == n-1) { cout << wt << endl;
           for (j = 1; j <= k; j++)
            cout << d[t[j]] << " " <<c[t[ij] << endl;
      return; }
```

```
int u= d[i], v= c[i];
       if (vs[u] == 0 && vs[v] == 0){
          vs[u]= k; vs[v]= k;}
           else
      if (vs[u]==0 && vs[v]!=0) vs[u]=  vs[v];
           else
     if (vs[u]!=0 && vs[v]==0) vs[v]=  vs[u];
           else
    if (vs[u] < vs[v]) {
          tg= vs[v];
       for (j= 1; j<= n; j++)
          if (vs[j] == tg) vs[j]= vs[u];  }
    else if (vs[v] < vs[u]) {tg= vs[u];
           for (j= 1; j<= n; j++)
            if (vs[j] == tg) vs[j]= vs[v];   }
     }
}
```

Ví dụ 6: Tìm cây khung nhỏ nhất

- Cho G:

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	0	2	1	3	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	2	0	2	∞	∞	5	5	∞	∞	∞	∞	∞	∞
3	1	2	0	4	∞	5	∞	∞	∞	∞	∞	∞	∞
4	3	∞	4	0	5	5	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	∞	5	0	6	∞	∞	∞	6	∞	∞	∞
6	∞	5	5	5	6	0	6	6	6	6	∞	∞	∞
7	∞	5	∞	∞	∞	6	0	6	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	6	6	0	7	∞	∞	7	7
9	∞	∞	∞	∞	∞	6	∞	7	0	7	7	∞	∞
0	∞	∞	∞	∞	6	6	∞	∞	7	0	7	7	∞
1	∞	∞	∞	∞	∞	∞	∞	∞	7	7	0	8	∞
2	∞	∞	∞	∞	∞	∞	∞	7	∞	7	8	0	8
3	∞	∞	∞	∞	∞	∞	∞	7	∞	∞	∞	8	0

- Tìm cây khung nhỏ nhất sử dụng Kruskal.

Giải

- Số đỉnh n = 13, số cạnh m = 26
- Liệt kê các cạnh theo trọng số tăng dần từ trái sang phải, từ trên xuống dưới:

Cạnh	TS	Cạnh	TS	Cạnh	TS	Cạnh	TS
(1,3)	1	(3,6)	5	(6,9)	6	(9,11)	7
(1,2)	2	(4,5)	5	(6,10)	6	(10,11)	7
(2,3)	2	(4,6)	5	(7,8)	6	(10,12)	7
(1,4)	3	(5,6)	6	(8,9)	7	(11,12)	8
(3,4)	4	(5,10)	6	(8,12)	7		
(2,6)	5	(6,7)	6	(8,13)	7	(12,13)	8
(2,7)	5	(6,8)	6	(9,10)	7		

Lập bảng:

Cạnh	TS
(1,3)	1
(1,2)	2
(2,3)	2
(1,4)	3
(3,4)	4
(2,6)	5
(2,7)	5
(3,6)	5
(4,5)	5
(4,6)	5
(5,6)	6
(5,10)	6
(6,7)	6
(6,8)	6
(6,9)	6

Cạnh e	T ∪ {e} không chứa chu trình?	T	Wt	k
(1,3)	Yes	(1,3)	1	1
(1,2)	Yes	(1,2)	3	2
(2,3)	No			
(1,4)	Yes	(1,4)	6	3
(3,4)	No			
(2,6)	Yes	(2,6)	11	4
(2,7)	Yes	(2,7)	16	5
(3,6)	No			
(4,5)	Yes	(4,5)	21	6
(4,6)	No			
(5,6)	No			
(5,10)	Yes	(5,10)	27	7
(6,7)	No			
(6,8)	Yes	(6,8)	33	8
(6,9)	Yes	(6,9)	39	9

Lập bảng (tiếp)

Cạnh	TS
(6,10)	6
(7,8)	6
(8,9)	7
(8,12)	7
(8,13)	7
(9,10)	7
(9,11)	7
(10,11)	7
(10,12)	7
(11,12)	8
(12,13)	8

Cạnh e	T ∪ {e} không chứa chu trình?	T	Wt	k
(6,10)	No			
(7,8)	No			
(8,9)	No			
(8,12)	Yes	(8,12)	46	10
(8,13)	Yes	(8,13)	53	11
(9,11)	Yes	(9,11)	60	12

Kết quả

Từ bảng có cây khung tìm được:

Wt = 60;

T = {(1,3),(1,2),(1,4),(2,6),(2,7),(4,5),(5,10),(6,8),
(6,9),(8,12),(8,13),(9,11)}

Nhận xét:

1) Sự khác nhau giữa thuật toán Prim và thuật toán Kruskal:

- Thuật toán Prim tiếp cận theo hướng chọn đỉnh, trong đó đỉnh v không thuộc cây được lựa chọn nếu cạnh nối v với đỉnh u đã thuộc cây có trọng số tối thiểu.

Khi đó v sẽ được đưa vào cây cùng với cạnh e = (u,v).

- Thuật toán Kruskal tiếp cận theo hướng chọn cạnh nên cần xếp các cạnh theo thứ tự tăng của trọng số.

Khi đó quá trình chọn cạnh sẽ xét lần lượt. Cạnh được bổ sung vào cây nếu không tạo thành chu trình.

2) **Có thể áp dung hai thuật toán trên để tìm cây khung lớn nhất.**

Tổng kết chương 5

- **Về lý thuyết**:

- Khái niệm cây khung và cây khung nhỏ nhất
- Thuật toán Prim
- Thuật toán Kruskal

- **Về các dạng bài tập**

- Viết chương trình mô tả thuật toán.
- Kiểm nghiệm các thuật toán.

Thảo luận


THANK YOU