BỘ GIÁO DỤC VÀ ĐÀO TẠO

HUTECH
Đại học Công nghệ Tp.HCM

Lý thuyết và Bài tập

LÝ THUYẾT ĐỒ THỊ

Bùi Phú Khuyên

ÔN TẬP LÝ THUYẾT ĐỒ THỊ
(KHÔNG ĐÁP ÁN)

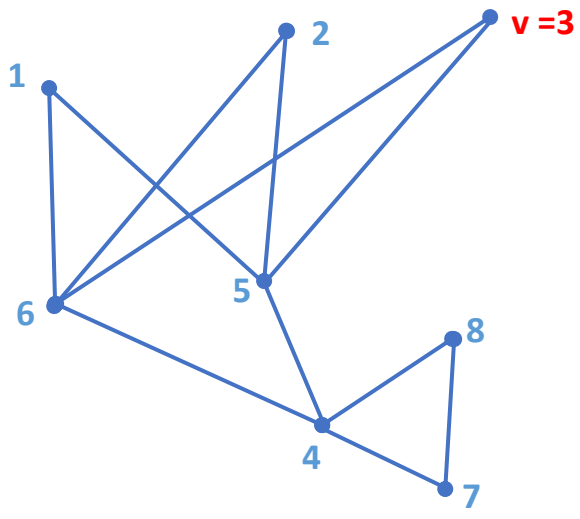
1. Đồ thị Euler:

- Thuật toán:

```
Input: Danh sách các đỉnh kề Adjacent_Vetices
Output: Chu trình Euler CE

Stack = ∅;
CE = ∅;
Chọn u là 1 đỉnh bất kỳ của đồ thị;
Push u;
while (Stack != ∅)
{
        x = top (Stack);
        if (Adjacent_Vetices(x) != 0)
        {
                y = Đỉnh đầu trong danh sách Adjacent_Vetices(x);
                Push y;
               Adjacent_Vetices(x) = Adjacent_Vetices(x)\{y};
               Adjacent_Vetices(y) = Adjacent_Vetices(y)\{x}; // Bỏ cạnh (x,y)
        }
        else
        {
                Pop x;
                CE ← x;
        }
}
```

- Ví dụ mẫu: Cho đồ thị sau, tìm chu trình Euler với đỉnh bắt đầu v = 3.



+ Bước 1: Tìm các đỉnh kề trong đồ thị:

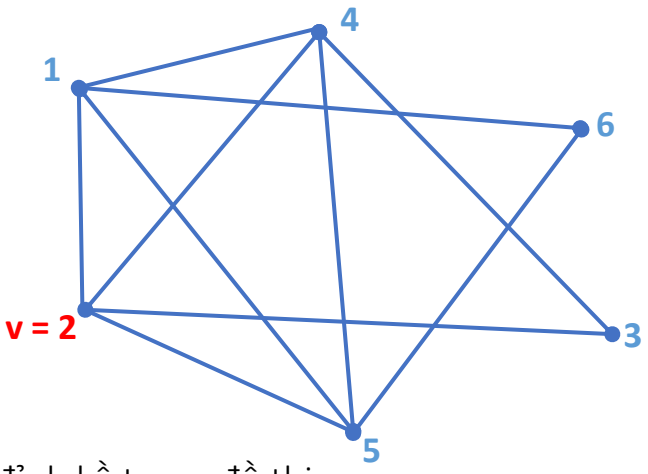
v	Các đỉnh kề của v
1	5, 6
2	5, 6
3	5, 6
4	5, 6, 7, 8
5	1, 2, 3, 4
6	1, 2, 3, 4
7	4, 8
8	4, 7

+ Bước 2: Chạy thuật toán Euler:

Stack	CE
3	
3, 5	
3, 5, 1	
3, 5, 1, 6	
3, 5, 1, 6, 2	
3, 5, 1, 6, 2, 5	
3, 5, 1, 6, 2, 5, 4	
3, 5, 1, 6, 2, 5, 4, 6	
3, 5, 1, 6, 2, 5, 4, 6, 3	
3, 5, 1, 6, 2, 5, 4	3, 6
3, 5, 1, 6, 2, 5, 4, 7	3, 6
3, 5, 1, 6, 2, 5, 4, 7, 8	3, 6
3, 5, 1, 6, 2, 5, 4, 7, 8, 4	3, 6
	3, 6, 4, 8, 7, 4, 7, 4, 5, 2, 6, 1, 5,3

→ Chu trình Euler: 3, 6, 4, 8, 7, 4, 7, 4, 5, 2, 6, 1, 5,3

- Bài tập: Cho đồ thị sau, tìm chu trình Euler với đỉnh bắt đầu v = 2



+ Bước 1: Tìm các đỉnh kề trong đồ thị:

v	Các đỉnh kề của v
1	
2	
3	
4	
5	
6	

+ Bước 2: Chạy thuật toán Euler:

Stack	CE

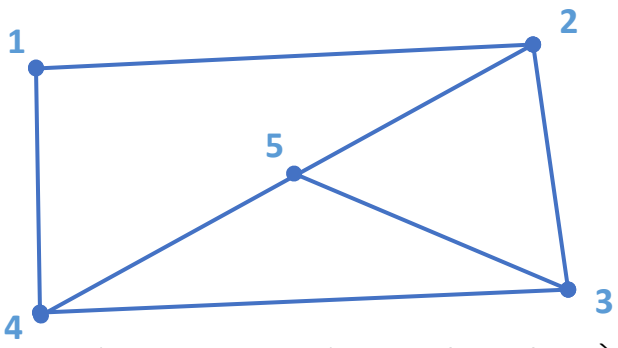
→ Chu trình Euler: .....

2. Đồ thị Hamilton:

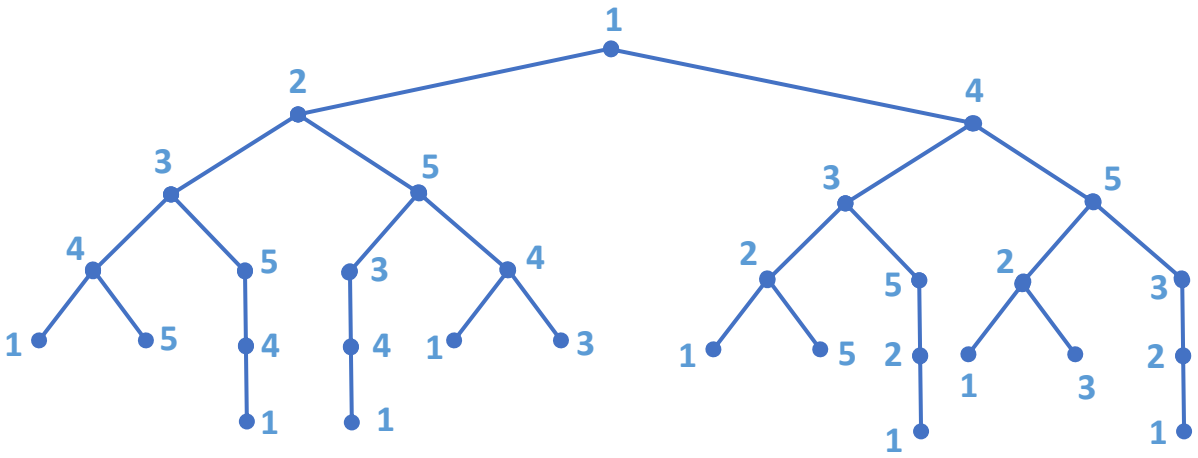
- Thuật toán:

```
Input: Đồ thị G = (V, E) lưu bằng danh sách các đỉnh kề Adjacent_Vetices
Output: Danh sách các chu trình Hamilton
//Mảng Visited để đánh dấu các đỉnh đã viếng thăm
//Mảng x chứa chu trình Hamilton, Vo là đỉnh bắt đầu
void _Hamilton(k)
{
        for (y∈ Adjacent_Vetices(x[k -1]))
                if ((k == n) && (y == vo))
                        Display(x);
                else
                        if (!Visited [y])
                        {
                                x[k] = y;
                                Visited[y] = 1;
                                _Hamilton(k + 1, vo);
                                Visited[y]==0; //Quay lui
                        }
}
void Hamilton(vo) //vo là đỉnh bắt đầu
{
        for(v ∈ V)
                Visited[v] = false;
                Visited[vo] = true;
                x[0] = vo;
                Hamilton(1);
}
```

- Ví dụ mẫu: Tìm tất cả các chu trình Hamilton của đồ thị sau:



Áp dụng thuật toán trên, thứ tự duyệt các đỉnh của đồ thị được minh họa qua hình dạng cây sau:



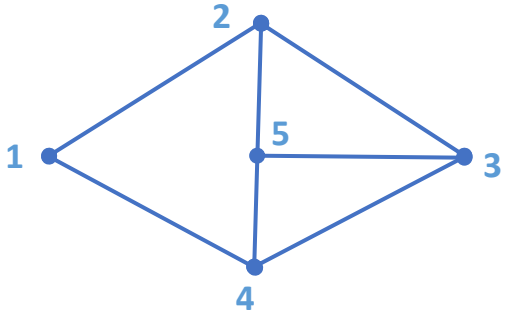
⇒ Các chu trình Hamilton: $1 \to 2 \to 3 \to 5 \to 4 \to 1$

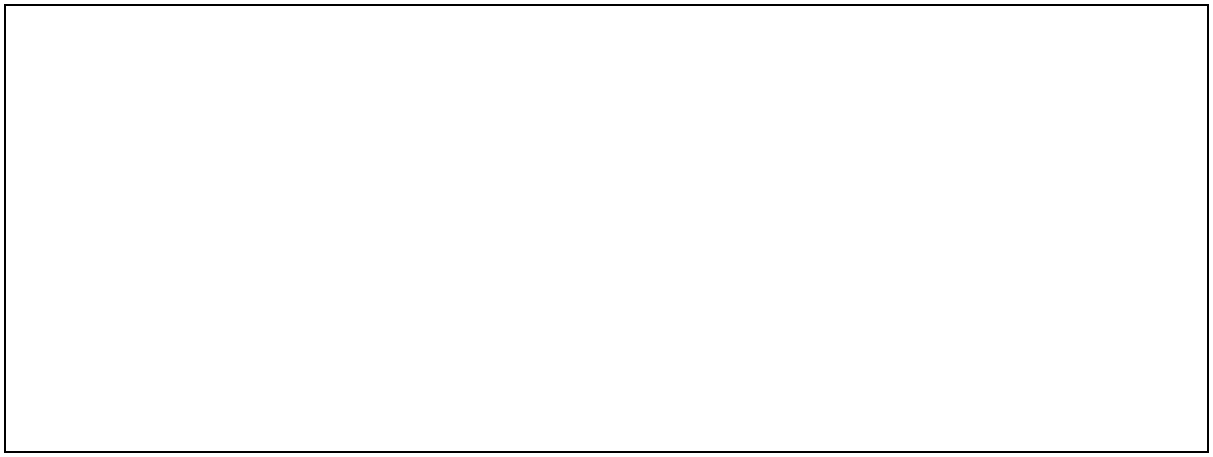
$1 \to 2 \to 5 \to 3 \to 4 \to 1$

$1 \to 4 \to 3 \to 5 \to 2 \to 1$

$1 \to 4 \to 5 \to 3 \to 2 \to 1$

- Bài tập: Tìm tất cả các chu trình Hamilton của đồ thị sau:





⇒ Các chu trình Hamilton:

.....

.....

.....

.....

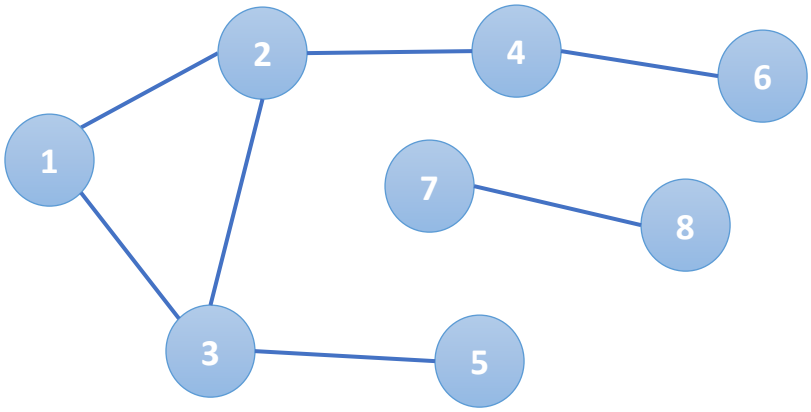
3. Các phép duyệt đồ thị:

a. Duyệt theo chiều rộng (BFS):

- Thuật toán:

```
void BFS (S)
// S: đỉnh bắt đầu duyệt
// Visited: là mảng cờ hiệu để xem các đỉnh viếng thăm chưa?
// true: đã viếng thăm, false: chưa viếng thăm
{
        for(v∈V)
                Visited[v] = false;
        Queue = ∅;
        Visited[S] = true;
        Enqueue(Queue, S); //Thêm S vào hang đợi
        while(Queue ≠ ∅)
        {
                Dequeue(Queue, u); //Lấy đỉnh u từ hàng đợi
                Visited_Vertex(u);
                for(v∈ Adjacent_Vertex(u))
                        if(!Visited[v])
                        {
                                Enqueue(Queue, v);
                                Visited[v] = true;
                        }
        }
}
```

- Ví dụ mẫu: Duyệt theo chiều rộng với đồ thị sau, đỉnh xuất phát s = 1



Hàng đợi	Đỉnh u (lấy ra từ hàng đợi)	Hàng đợi (sau khi lấy u ra)	Các đỉnh v kề u mà chưa viếng thăm	Hàng đợi sau khi đẩy những đỉnh v vào
1	1		2, 3	2, 3
2, 3	2	3	4	3, 4
3, 4	3	4	5	4, 5
4, 5	4	5	6	5, 6
5, 6	5	6		6
6	6			

⇒ Kết quả duyệt: 1, 2, 3, 4, 5, 6

Chú ý: Cách duyệt theo chiều rộng bằng thuật toán loang:

(1) Khởi tạo:
- Bắt đầu từ đỉnh s, đánh dấu đỉnh s, các đỉnh khác s đều chưa bị đánh dấu
- Old ← {s}

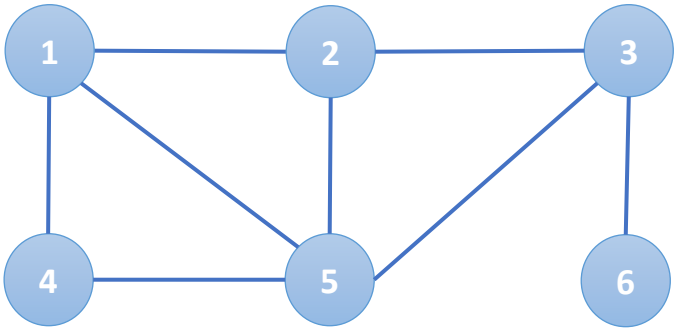
(2) Lặp các bước sau đến khi Old là tập rỗng:
- Đặt New ← ∅
- Với mọi đỉnh u ∈ Old, với mỗi đỉnh u đó:
  + Thăm u
  + Xét tất cả những đỉnh v kề với u mà chưa bị đánh dấu, với mỗi đỉnh v đó:
    . Đánh dấu v
    . Đưa v vào tập New
  + Old ← New

- Ví dụ mẫu: Tương tự đồ thị như trên nhưng ta có các duyệt bằng thuật toán loang

Bước	Tập Old	Tập New	Đỉnh u từ tập Old	Các đỉnh v kề u mà chưa viếng thăm	Tập New sau khi đưa đỉnh v vào
0	{1}	{}	1	2, 3	{2, 3}
1	{2, 3}	{}	2	4	{4}
	{2, 3}	{4}	3	5	{4, 5}
2	{4, 5}	{}	4	6	{6}
	{4, 5}	{6}	5		{6}
3	{6}	{}	6		{}
4	{}	{}	→ Dừng		

⇒ Kết quả duyệt: 1, 2, 3, 4, 5, 6

- Bài tập: Duyệt theo chiều rộng đồ thị sau bằng 2 cách, đỉnh s = 1



Cách 1:

Hàng đợi	Đỉnh u (lấy ra từ hàng đợi)	Hàng đợi (sau khi lấy u ra)	Các đỉnh v kề u mà chưa viếng thăm	Hàng đợi sau khi đẩy những đỉnh v vào

Cách 2: *(Sử dụng thuật toán Loang)*

Bước	Tập Old	Tập New	Đỉnh u từ tập Old	Các đỉnh v kề u mà chưa viếng thăm	Tập New sau khi đưa đỉnh v vào

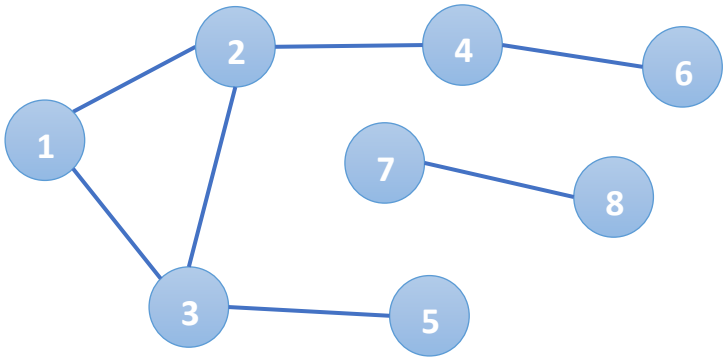
⇒ Kết quả duyệt: .....

b. Duyệt theo chiều sâu (DFS):

- Thuật toán:

```
void DFS(s)
{
        Đánh dấu s đã viếng thăm
        Đưa s vào stack
        while(stack ≠ ∅)
        {
                Lấy 1 đỉnh u từ stack
                Tìm một đỉnh v chưa được viếng thăm và kề u
                {
                        Đánh dấu v đã viếng thăm
                        Đẩy u vào lại stack
                        Đẩy v vào stack
                }
        }
}
```

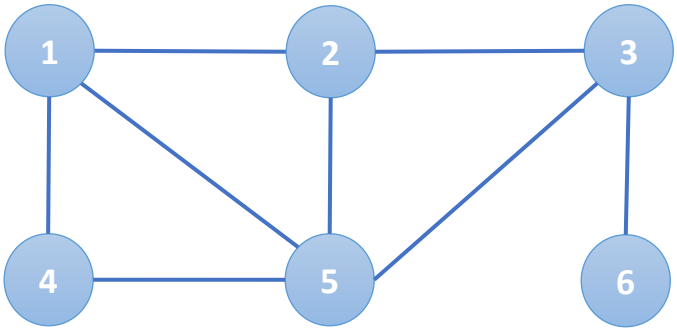
- Ví dụ mẫu: Duyệt theo chiều sâu với đồ thị sau, đỉnh xuất phát s = 1



Bước lặp	Stack	u	v	Stack sau mỗi bước	Giải thích
1	1	1	**2**	1, 2	Tiến sâu xuống thăm 2
2	1, 2	2	**3**	1, 2, 3	Tiến sâu xuống thăm 3
3	1, 2, 3	3	**5**	1, 2, 3, 5	Tiến sâu xuống thăm 5
4	1, 2, 3, 5	5		1, 2, 3	Lùi lại
5	1, 2, 3	3		1, 2	Lùi lại
6	1, 2	2	**4**	1, 2, 4	Tiến sâu xuống thăm 4
7	1, 2, 4	4	**6**	1, 2, 4, 6	Tiến sâu xuống thăm 6
8	1, 2, 4, 6	6		1, 2, 4	Lùi lại
9	1, 2, 4	4		1, 2	Lùi lại
10	1, 2	2		1	Lùi lại
11	1	1			Lùi hết dây chuyền, xong

⇒ Kết quả duyệt: 1, 2, 3, 5, 4, 6

- Bài tập: Duyệt theo chiều sâu đồ thị sau bằng 2 cách, đỉnh s = 1



Bước lặp	Stack	u	v	Stack sau mỗi bước	Giải thích
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					

⇒ Kết quả duyệt: .....

4. Thuật toán tìm đường đi ngắn nhất:

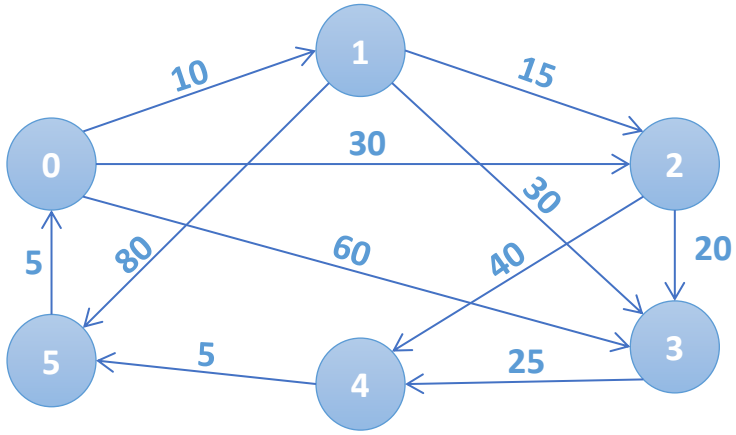
a. Thuật toán Dijkstra:

- Thuật toán:

(1) Khởi tạo:
T = V;
dist[u] = 0; // Lưu khoảng cách ngắn nhất từ đỉnh bắt đầu đến đỉnh đang xét
d [k] = +∞ ∀ k ∈ V\{u};
prev[k] = -1 ∀ k ∈ V; // Lưu đỉnh đi liền trước để lưu lại đường đi ngắn nhất
(2) Lặp cho đến khi V ∉ T
- Chọn min: chọn đỉnh i ∈ T sao cho dist[i] nhỏ nhất
- Cập nhật: Với mọi đỉnh k kề i
dist[k] = min {dist[k],dist[i] + w(i,k)};
prev[k] = i; // Nếu đường đi ngắn hơn khi đi qua đỉnh i;
- T = T\{i};
(3) Giá trị dist[v] chính là đồ dài đường đi ngắn nhất từ u đến v và prev[v] là đỉnh nằm ngay trước v trên đường đi đó.

Lưu ý: Thuật toán có thể áp dụng tìm đường đi ngắn nhất từ một đỉnh bất kì đến tất cả các đỉnh còn lại (chỉ cần sửa lại điều kiện kết thúc vòng lặp: T = ∅)

- Ví dụ mẫu: Cho đồ thị sau, tìm đường đi ngắn nhất từ u = 0 đến v = 5



T	Đỉnh min	dist, prev					
		0	1	2	3	4	5
{0, 1, 2, 3, 4, 5}		0,-1	+∞,1	+∞,-1	+∞,-1	+∞,-1	+∞,-1
{1, 2, 3, 4, 5}	0	0,-1	10,0	30,0	60,0	+∞,-1	+∞,-1
{2, 3, 4, 5}	1	0,-1	10,0	25, 1	40,1	+∞,-1	90,1
{3, 4, 5}	2	0,-1	10,0	25, 1	40,1	65,2	90,1
{4, 5}	3	0,-1	10,0	25, 1	40,1	65,2	90,1
{5}	4	0,-1	10,0	25, 1	40,1	65,2	70,4
{}	5	0,-1	10,0	25, 1	40,1	65,2	70,4

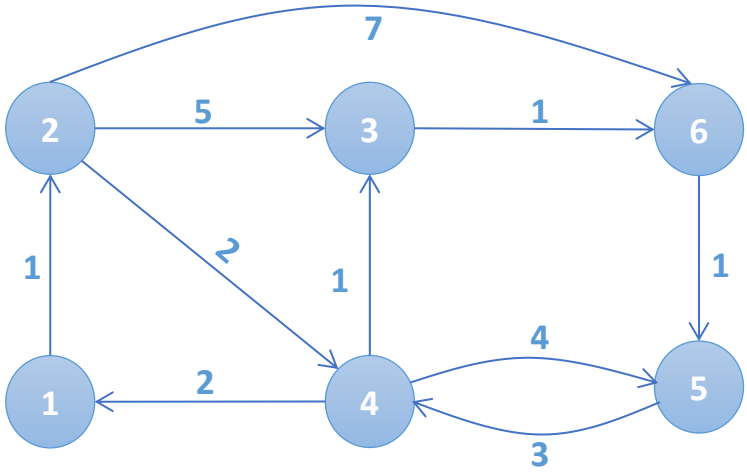
⇒ Đường đi: 5 ← 4 ← 2 ← 1 ← 0 với giá thành 70

Thêm: Tìm đường đi ngắn nhất từ đỉnh u = 0 đến tất cả các đỉnh còn lại:

T	Đỉnh min	dist, prev					
		0	1	2	3	4	5
{0, 1, 2, 3, 4, 5}		0,-1	+∞,-1	+∞,-1	+∞,-1	+∞,-1	+∞,-1
{1, 2, 3, 4, 5}	0	0,-1	10,0	30,0	60,0	+∞,-1	+∞,-1
{2, 3, 4, 5}	1	0,-1	10,0	25, 1	40,1	+∞,-1	90,1
{3, 4, 5}	2	0,-1	10,0	25, 1	40,1	65,2	90,1
{4, 5}	3	0,-1	10,0	25, 1	40,1	65,2	90,1
{5}	4	0,-1	10,0	25, 1	40,1	65,2	70,4
{}	5	0,-1	10,0	25, 1	40,1	65,2	70,4

Kết luận:

- Bài tập: Tìm đường đi ngắn nhất từ đỉnh u = 1 đến tất cả các đỉnh:



T	Đỉnh min	dist, prev					
		1	2	3	4	5	6

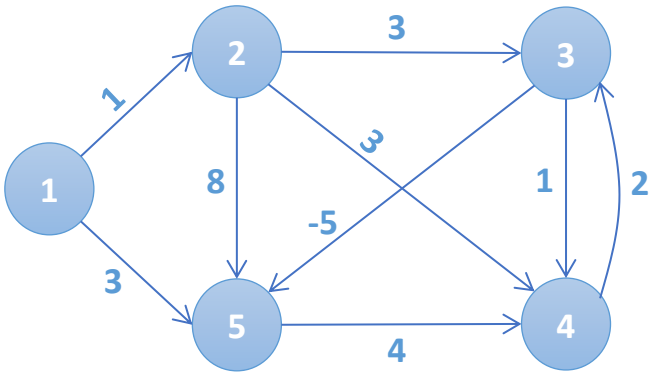
b. Thuật toán Bellman:

- Thuật toán:

```
(0) Cho trước đỉnh s ∈ V
(1) Khởi tạo:
        for(v ∈ V)
            dist[v] = +∞; // Lưu khoảng cách ngắn nhất từ đỉnh s → v ở bước 0
            prev[v] = -1; //Lưu đỉnh đi liền trước để lưu lại đường đi ngắn nhất
        dist[v] = 0;
(2) Kết nạp cạnh:
        for i = 1 to [V] – 1
                for(e(u,v) ∈ E)
                        if(dist[v] > dist[u] + Duv)
                                dist[v] = dist[u] + Duv;
                                prev[v] = u;
(3) Kiểm tra chu trình âm
        for(e(u,v) ∈ E)
                if(dist[v] > dist[u] + Duv)
```

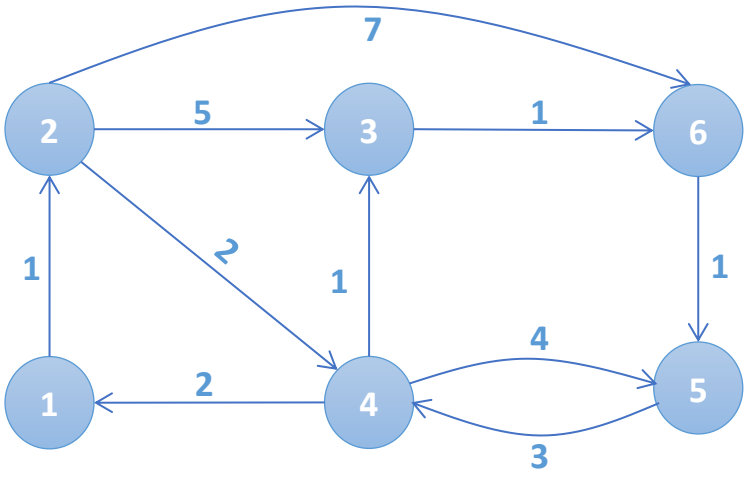
```
                return NEGATIVE CYCLE;
return dist[], prev[];
```

- Ví dụ mẫu: Sử dụng thuật toán Bellman tìm đường đi ngắn nhất của đồ thị sau (với đỉnh bắt đầu là 1)



i	dist, prev				
	1	2	3	4	5
0	0,-1	+∞,-1	+∞,-1	+∞,-1	+∞,-1
1	0,-1	1,1	+∞,-1	+∞,-1	3, 1
2	0,-1	1,1	4,2	4,2	3, 1
3	0,-1	1,1	4,2	4,2	-1,3
4	0,-1	1,1	4,2	3,5	-1,3

- Bài tập thêm: Sử dụng thuật toán Bellman tìm đường đi ngắn nhất của đồ thị sau (với đỉnh bắt đầu là 1)



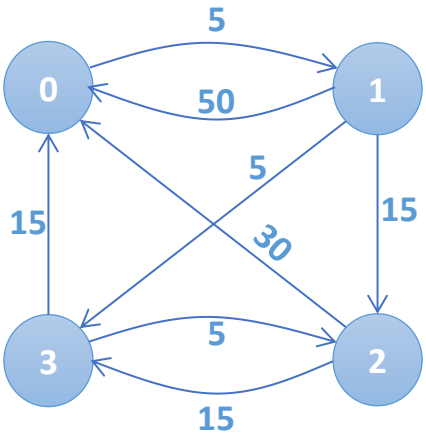
i	dist, prev					
	1	2	3	4	5	6
0						
1						
2						
3						
4						
5						

c. Thuật toán Floyd:

- Thuật toán:

(1) Khởi tạo:
$dist_{uv}^{(0)} = D_{uv}$;
$path_{uv}^{(0)} = -1$; // Đường đi ngắn nhất từ u đến v không có đỉnh trung gian nào cả
(2) Lặp:
For k = 1 to n
/* Kiểm tra với mỗi cặp đỉnh u và v có (không) tồn tại một đường đi qua trung gian đỉnh thứ k mà có khoảng cách ngắn hơn với kết quả ở bước (k-1) */
$dist_{uv}^{(0)} = \min(dist_{uv}^{(k-1)}, dist_{uk}^{(k-1)} + dist_{kv}^{(k-1)}$;
$path_{uv}^{(0)}$ = k nếu đường đi ngắn hơn qua đỉnh trung gian k

- Ví dụ mẫu: Cho đồ thị sau, sử dụng thuật toán Floyd tìm ra đường đi ngắn nhất giữa tất cả cặp đỉnh bất kỳ của đồ thị.



- Bước 0:

$$dist^{(0)} = \begin{bmatrix} 0 & 5 & +\infty & +\infty \\ 50 & 0 & 15 & 5 \\ 30 & +\infty & 0 & 15 \\ 15 & +\infty & 5 & 0 \end{bmatrix} \qquad path^{(0)} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix}$$

- Bước 1:

$$dist^{(1)} = \begin{bmatrix} 0 & 5 & +\infty & +\infty \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix} \qquad path^{(1)} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

- Bước 2:

$$dist^{(2)} = \begin{bmatrix} 0 & 5 & 20 & 10 \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix} \qquad path^{(2)} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

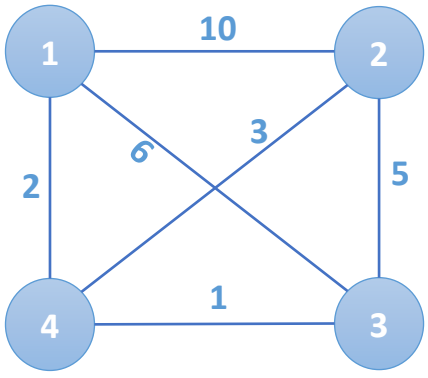
- Bước 3:

$$dist^{(3)} = \begin{bmatrix} 0 & 5 & 20 & 10 \\ 45 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix} \qquad path^{(3)} = \begin{bmatrix} -1 & -1 & 1 & 1 \\ 2 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

- Bước 4:

$$dist^{(4)} = \begin{bmatrix} 0 & 5 & 15 & 10 \\ 20 & 0 & 10 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix} \qquad path^{(4)} = \begin{bmatrix} -1 & -1 & 3 & 1 \\ 3 & -1 & 3 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

- **Bài tập: Cho đồ thị sau, sử dụng thuật toán Floyd tìm ra đường đi ngắn nhất giữa tất cả cặp đỉnh bất kỳ của đồ thị.**



- Bước 0:

$dist^{(0)} = \begin{bmatrix} & \end{bmatrix}$ $path^{(0)} = \begin{bmatrix} & \end{bmatrix}$

- Bước 1:

$dist^{(1)} = \begin{bmatrix} & \end{bmatrix}$ $path^{(1)} = \begin{bmatrix} & \end{bmatrix}$

- Bước 2:

$dist^{(2)} = \begin{bmatrix} & \end{bmatrix}$ $path^{(2)} = \begin{bmatrix} & \end{bmatrix}$

- Bước 3:

$dist^{(3)} = \begin{bmatrix} & \end{bmatrix}$ $path^{(3)} = \begin{bmatrix} & \end{bmatrix}$

- Bước 4:

$dist^{(4)} = \begin{bmatrix} & \end{bmatrix}$ $path^{(4)} = \begin{bmatrix} & \end{bmatrix}$

5. Thuật toán tìm cây khung nhỏ nhất:

a. Thuật toán Prim:

- Thuật toán:

Cho G = (V,E) là một đồ thị liên thông có trọng số gồm n đỉnh:

(1) Chọn tuỳ ý một đỉnh bất kỳ v ∈ V và khởi tạo Y = {v} và T = ∅

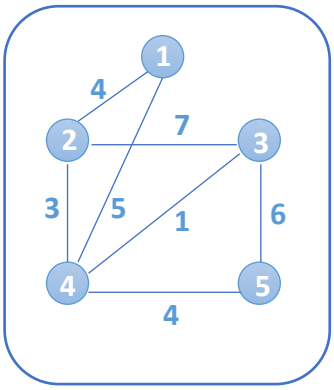
(2) Trong số những cạnh e = (v,w), trong đó v ∈ Y và w ∈ V\Y, ta chọn cạnh có độ dài nhỏ nhất

(3) Gán Y = Y ∪ {w} và T = T ∪ {e}

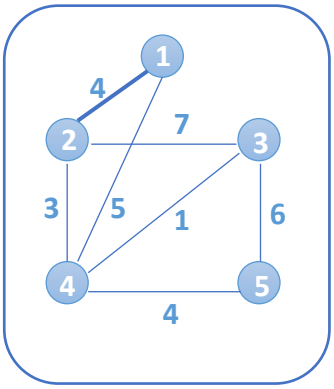
(4) Nếu T đủ (n-1) phần tử thì dừng, ngược lại làm tiếp (2)

⇒ T chỉnh là cây khung nhỏ nhất

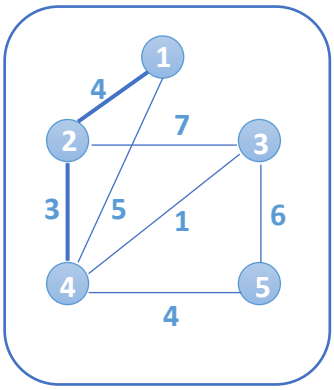
- Ví dụ mẫu: Sử dụng thuật toán Prim tìm cây khung nhỏ nhất của đồ thị sau:



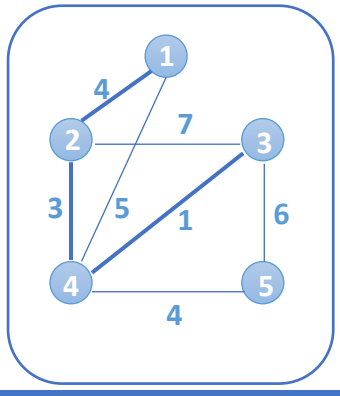
- Giả sử chọn đỉnh 1 là đỉnh đầu
- Khởi tạo:
  Y = {1}
  T = ∅



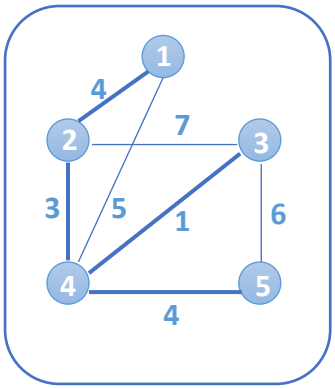
- Chọn cạnh (1,2) vì là cạnh có trọng số nhỏ nhất trong số các cạnh thoả yêu cầu (1,2) , (1,4)
- Cập nhật:
  Y = {1, 2}
  T = {(1,2)}



- Chọn cạnh (2,4) vì là cạnh có trọng số nhỏ nhất trong số các cạnh thoả yêu cầu (1,4) , (2,3) , (2,4)
- Cập nhật:
  Y = {1, 2, 4}
  T = {(1,2) , (2,4)}



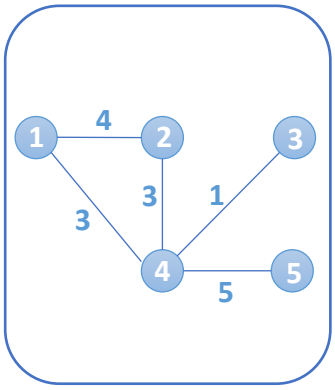
- Chọn cạnh (4,3) vì là cạnh có trọng số nhỏ nhất trong số các cạnh thoả yêu cầu (2,3) , (4,3) , (4,5)
- Cập nhật:
  Y = {1, 2, 4, 3}
  T = {(1,2) , (2,4) , (4,3)}

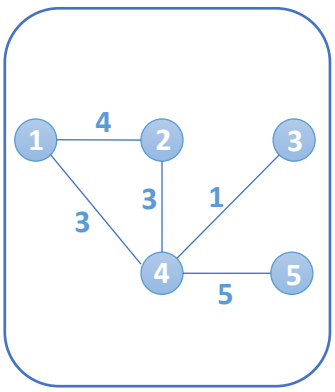


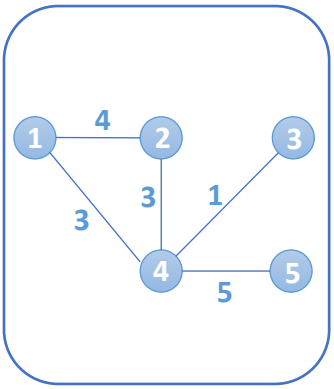
- Chọn cạnh (4,5) vì là cạnh có trọng số nhỏ nhất trong số các cạnh thoả yêu cầu (3,5) , (4,5)
- Cập nhật:
  Y = {1, 2, 4, 3, 5}
  T = {(1,2) , (2,4) , (4,3) , (4,5)}

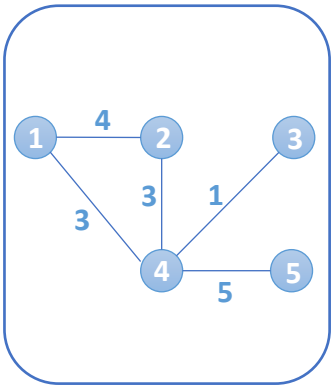
- Kết thúc thuật toán vì |T| = 4
- Cây khung nhỏ nhất tìm được có trọng số bằng 12
  T = {(1,2) , (2,4) , (4,3) , (4,5)}

- **Bài tập: Sử dụng thuật toán Prim tìm cây khung nhỏ nhất của đồ thị sau:**







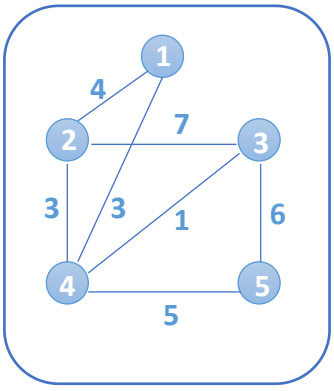


b. Thuật toán Kruskal:

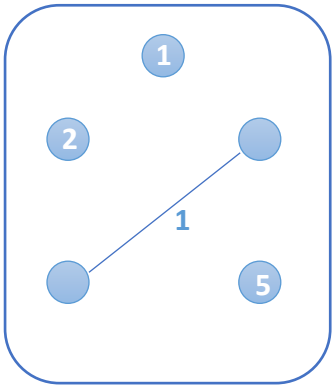
- Thuật toán:

Cho G = (V,E) là một đồ thị liên thông có trọng số gồm n đỉnh: (1) Sắp xếp các cạnh theo thứ tự đồ dài không giảm dần và khởi tạo: T = ∅ (2) Lần lượt lấy từng cạnh e trong danh sách đã sắp xếp. Nếu T ∪ {e} không tạo thành chu trình thì T = T ∪ {e} (3) Nếu T đủ (n-1) cạnh thì dừng, ngược lại thì làm tiếp (2)

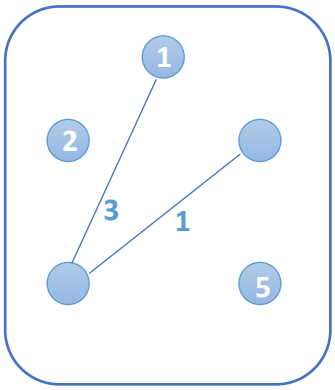
- Ví dụ mẫu: Sử dụng thuật toán Kruskal tìm cây khung nhỏ nhất của đồ thị sau:



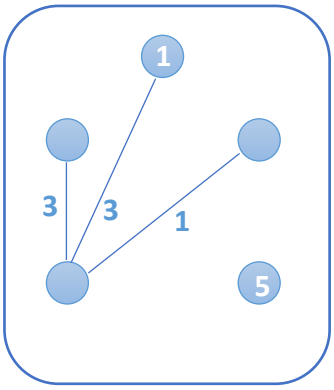
- Sắp xếp các cạnh theo thứ tự không giảm dần:
  (3,4) = 1, (1,4) = 3, (2,4) = 3, (1,2) = 4, (4,5) = 5, (3,5) = 6, (2,3) = 7
- T = ∅
- Chọn cạnh (3,4): T = {(3,4)}



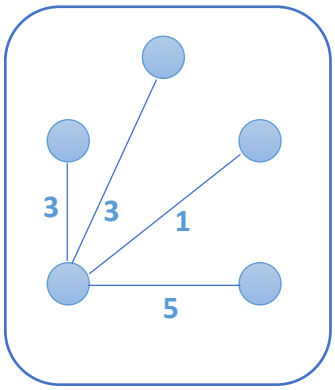
- Chọn cạnh (1,4): T = {(3,4) , (1,4)}



- Chọn cạnh (2,4): T = {(3,4) , (1,4) , (2,4)}

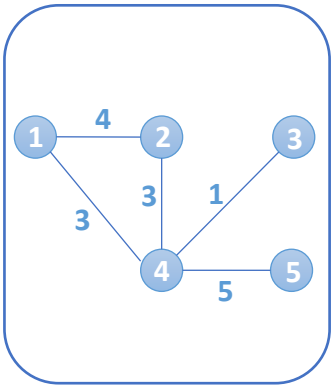


- Không chọn cạnh (1,2) vì sẽ tạo thành chu trình
- Chọn cạnh (4,5): T = {(3,4) , (1,4) , (2,4) , (4,5)}

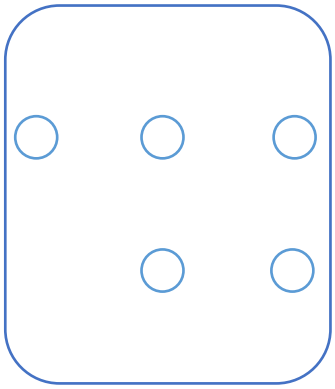


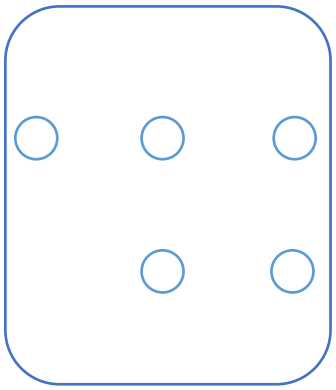
- Do |T| = 4. Kết thúc thuật toán
- Cây khung tìm được là: T = {(3,4) , (1,4) , (2,4) , (4,5)} với độ dài là 12.

- **Bài tập: Sử dụng thuật toán Kruskal tìm cây khung nhỏ nhất của đồ thị sau:**

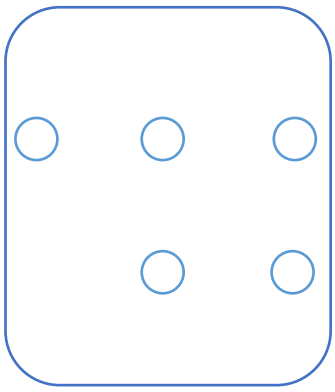


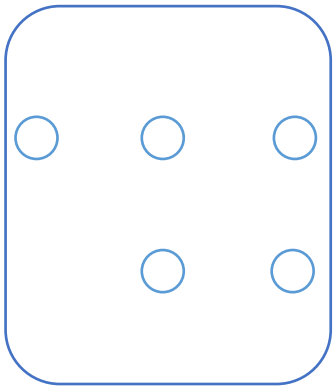
- Sắp xếp các cạnh theo thứ tự không giảm dần:
  _____
- T = ∅
- _____
- _____





- _____
- _____





6. Thuật toán tô màu đồ thị:

- Thuật toán: (Thuật toán tham lam)

(1) Khởi tạo:

Sắp xếp các đỉnh theo một trật tự nào đó $\{V_1, V_2, V_3, \ldots V_n\}$

Mỗi đỉnh V_k có một tập màu có thể tô là: $C_k = \{1, 2, 3, \ldots\}$

(2) Tô màu:

Xét tuần tự đỉnh V_k theo trật tự sắp xếp:

+ Tô màu đỉnh V_k bằng màu c là màu nhỏ nhất trong C_k

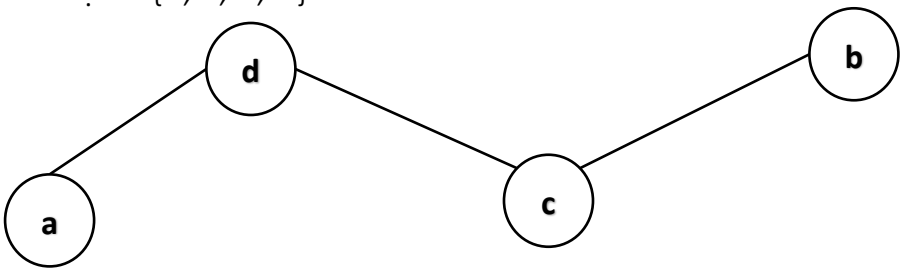
+ Loại bỏ màu c khỏi danh sách màu những đỉnh V_x kề với V_k :

$C_x = C_x \setminus \{c\}$

(3) Giải thuật kết thúc khi tất cả các đỉnh của đồ thị được tô. Số màu đã sử dụng chính là sắc số của đồ thị

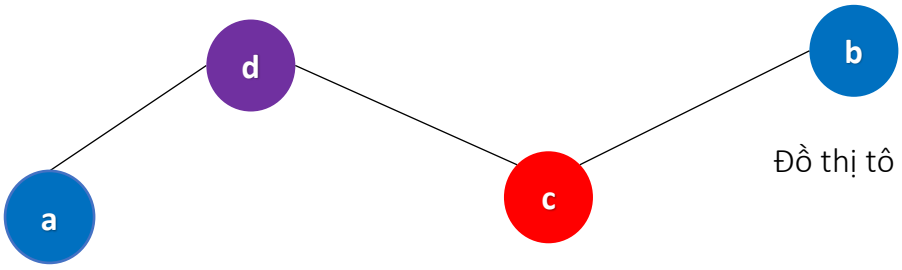
- Ví dụ mẫu: Dùng thuật toán tham lam để tô màu đồ thị sau theo 2 thứ tự sau:

- Thứ tự 1: {a, b, c, d}
- Thứ tự 2: {d, c, a, b}



- Vì đồ thị có 4 đỉnh nên số màu tối đa cần dùng là 4 màu:
- Ta lấy 4 màu: Xanh (1), Đỏ (2), Tím (3), Vàng (4) → C_k = {1, 2, 3, 4}
- Thứ tự 1:

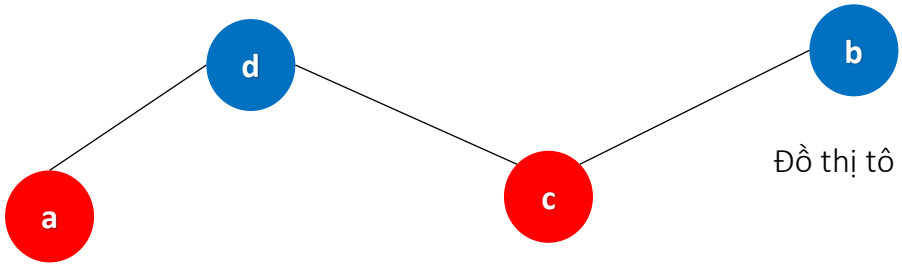
Bước	a	b	c	d	Màu tô
0	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	
1	**1**	{1, 2, 3, 4}	{1, 2, 3, 4}	{2, 3, 4}	a - Xanh
2		**1**	{2, 3, 4}	{2, 3, 4}	b - Xanh
3			**2**	{3, 4}	c - Đỏ
4				**3**	d - Tím



Đồ thị tô bởi 3 màu

- Thứ tự 2:

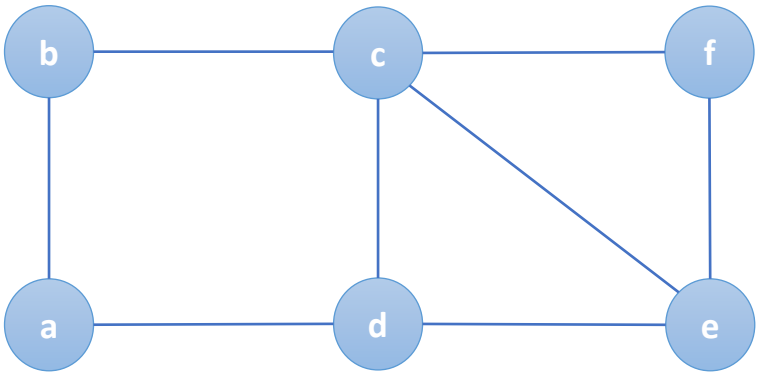
Bước	d	c	a	b	Màu tô
0	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	{1, 2, 3, 4}	
1	**1**	{2, 3, 4}	{2, 3, 4}	{1, 2, 3, 4}	d - Xanh
2		**2**	{2, 3, 4}	{1, 3, 4}	c - Đỏ
3			**2**	{1, 3, 4}	a - Đỏ
4				**1**	b - Xanh



Đồ thị tô bởi 2 màu

Lưu ý: Việc sắp xếp các đỉnh của đồ thị theo thứ tự giảm dần của bậc đỉnh sẽ cho kết quả tốt hơn (Thuật toán Welsh – Powell)

- Bài tập: Dùng thuật toán Welsh – Powell để tô màu đồ thị sau:



- Sắp xếp đỉnh theo thứ tự bậc giảm dần:

Đỉnh						
Bậc						

- Thực hiện tô màu đồ thị:

Bước							Màu tô
0							
1							
2							
3							
4							
5							
6							

Xem đáp án các phần bài tập: Liên hệ Facebook