Analysis and Design of Algorithms

Lecture 6,7

The Greedy algorithms

Nội dung

1. Lược đồ chung
2. Bài toán cái túi
3. Bài toán người du lịch
4. Đường đi ngắn nhất
5. **Cây bao trùm nhỏ nhất**
6. Bài toán tô màu
7. Bài toán các khoảng không giao nhau

Bài toán

- Cho đơn đồ thị G=(V,E)
  - V: Tập các đỉnh
  - E: Tập các cạnh
- Cây T gọi là cây bao trùm của G nếu T là đồ thị con của G và chứa tất cả các đỉnh thuộc G (có số đỉnh =V)

⇨ **Tìm cây bao trùm có trọng số nhỏ nhất (Minimal Spanning Tree) MST**

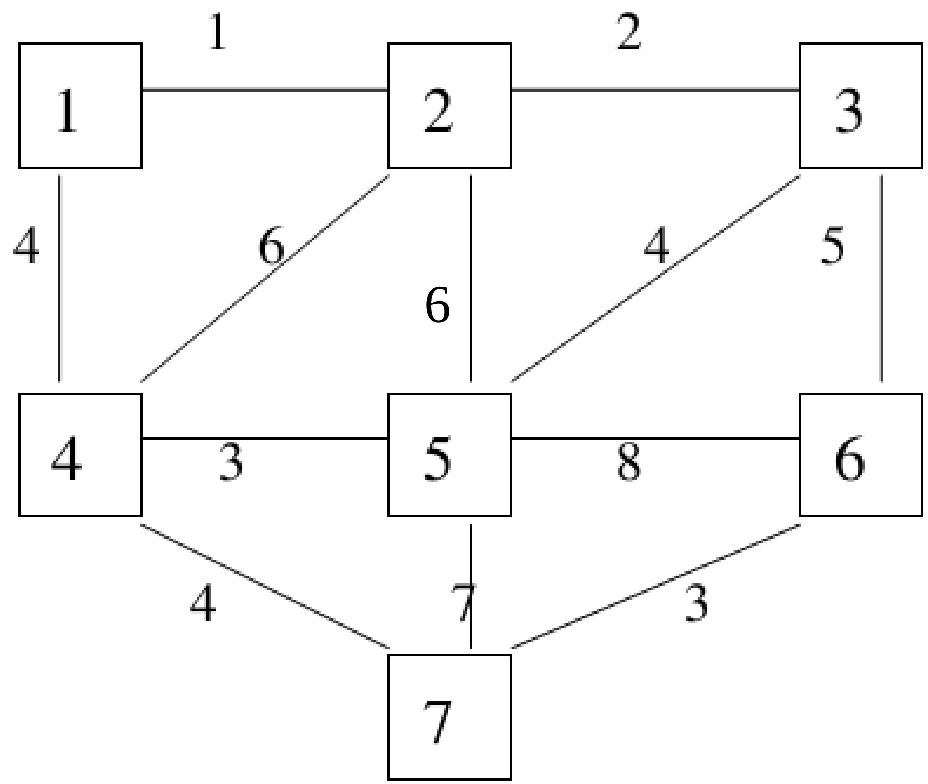
Thuật toán Prim

- $T = G_T(V_T, E_T)$ là cây khung tối thiểu cần tìm
- Ý tưởng
  - Chọn 1 đỉnh tùy ý vào V_T
  - Khi $|V_T| < |V|$
    - Tìm cạnh (s,t) $s \in V_T$, $t \in V \setminus V_T$ có trọng số nhỏ nhất (**tham lam**) nối V_T và $V \setminus V_T$
    - Thêm đỉnh t vào V_T, (s,t) vào E_T

Minh họa

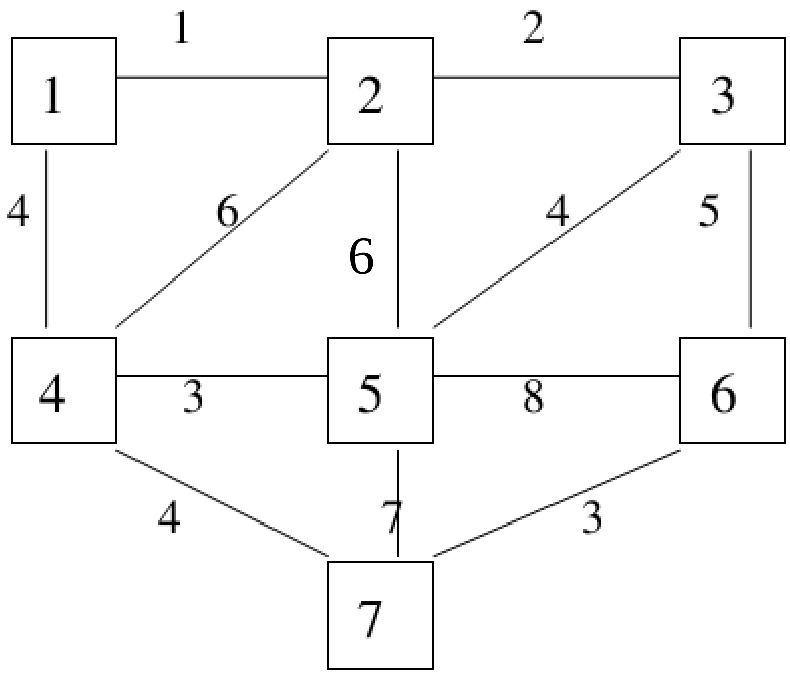
- Cho đồ thị

G = (V,E) =



Khởi tạo

- Bắt đầu từ đỉnh 1



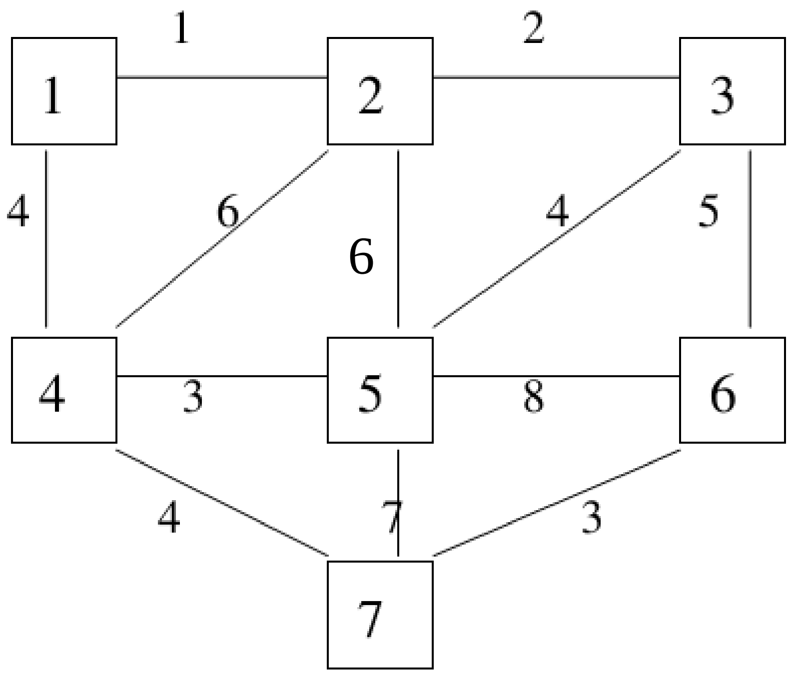
Đồ thị G

MST T

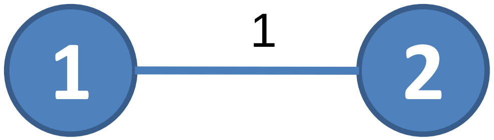
Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Bước 1

- Bắt đầu từ đỉnh 1



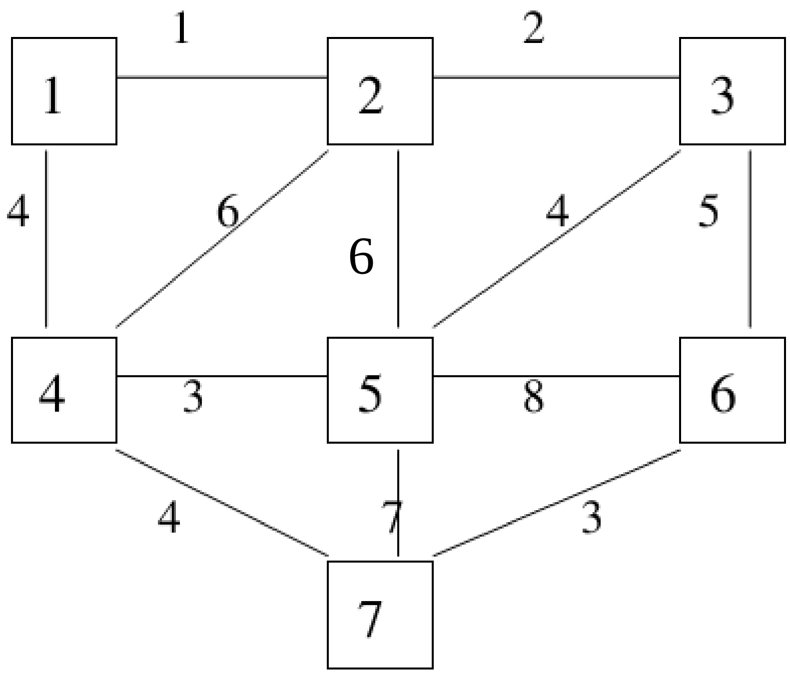
Đồ thị G



MST T

Bước 2

- Bắt đầu từ đỉnh 1



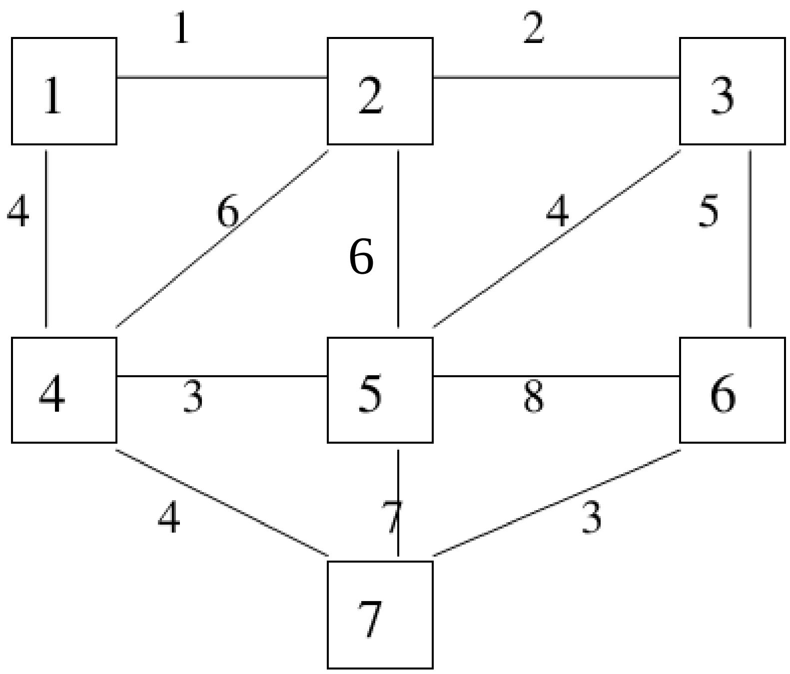
Đồ thị G

MST T

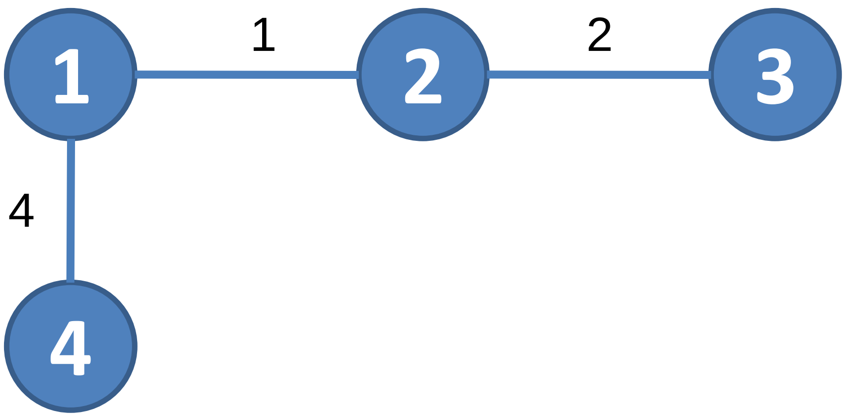
Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Bước 3

- Bắt đầu từ đỉnh 1



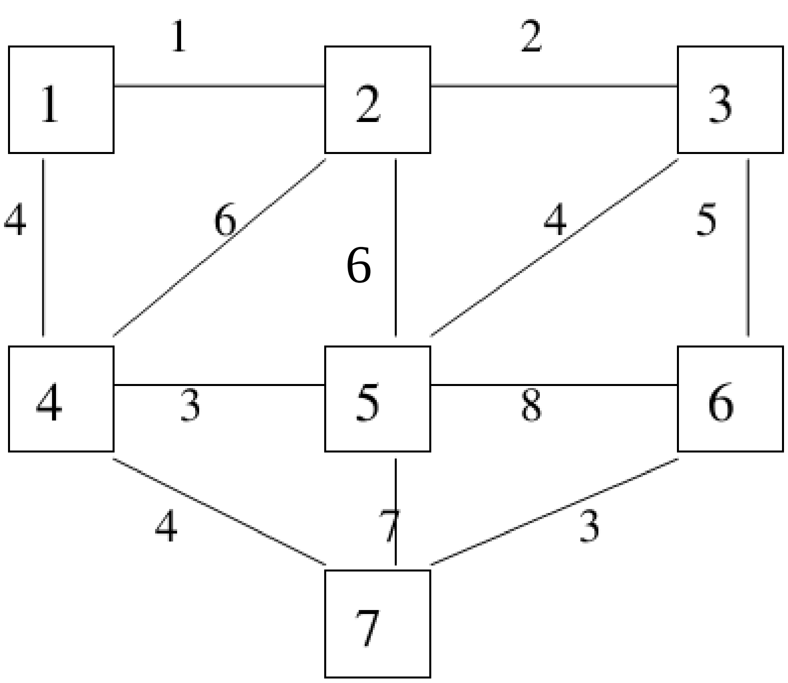
Đồ thị G



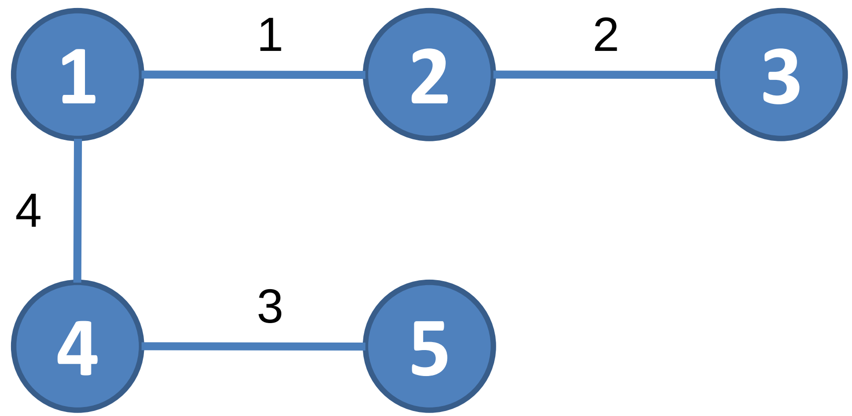
MST T

Bước 4

- Bắt đầu từ đỉnh 1



Đồ thị G

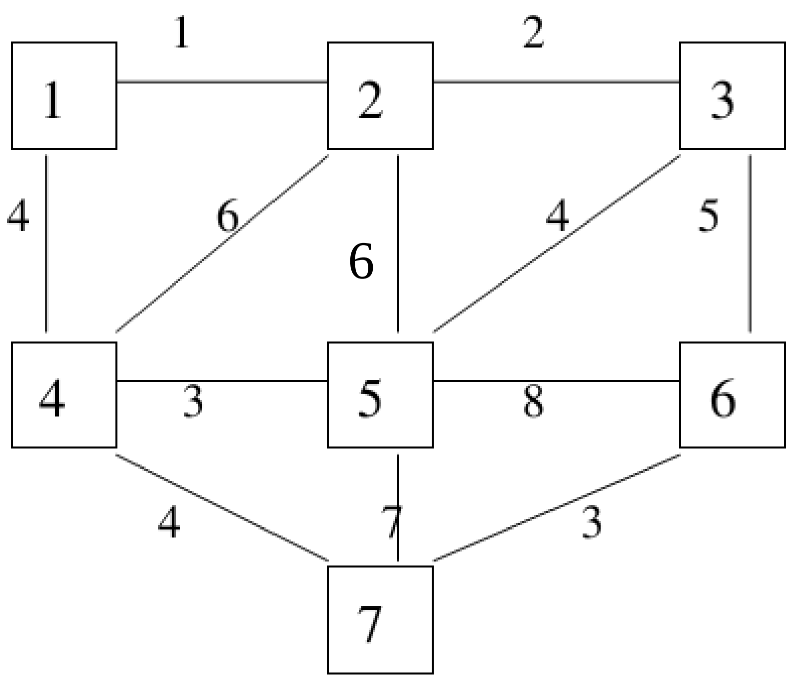


MST T

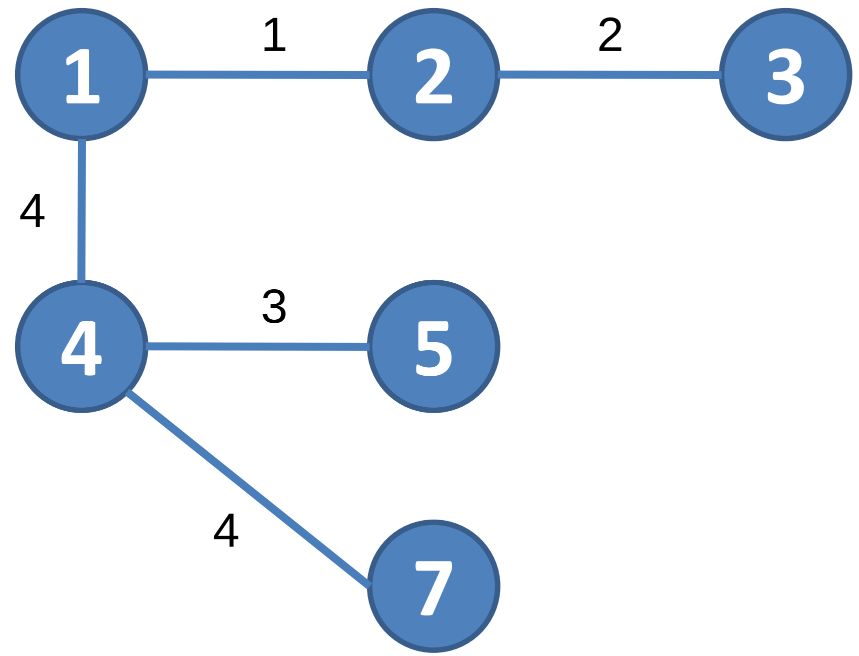
Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Bước 5

- Bắt đầu từ đỉnh 1



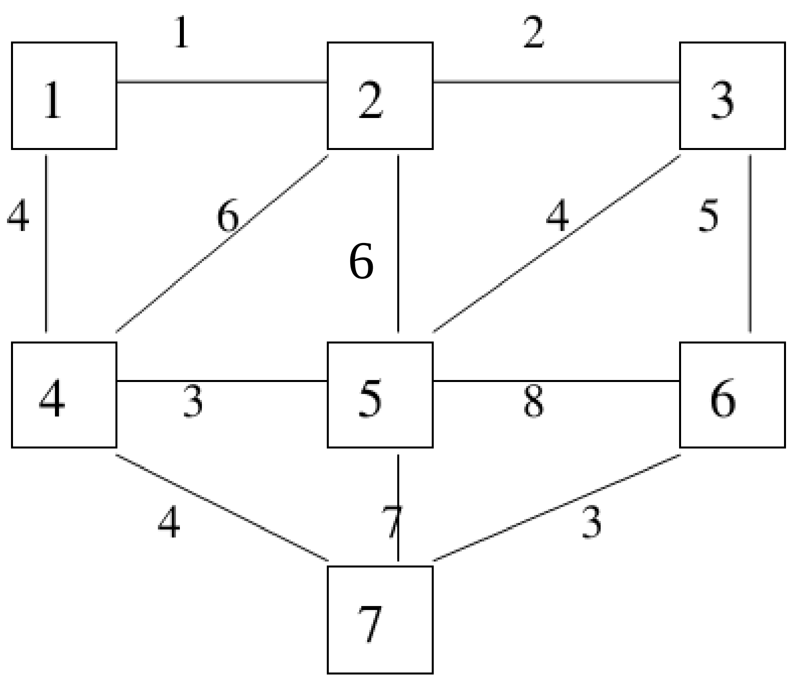
Đồ thị G



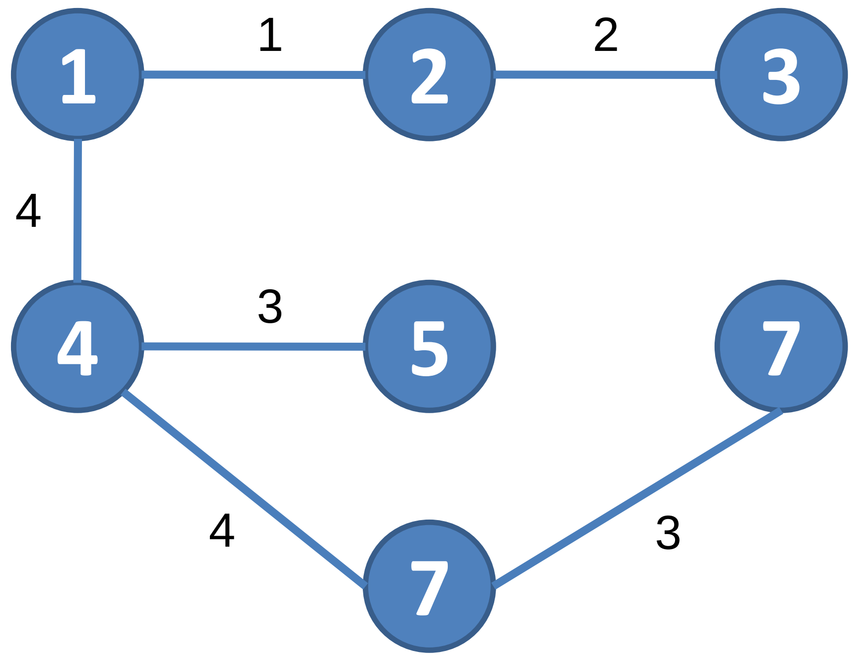
MST T

Bước 6

- Bắt đầu từ đỉnh 1



Đồ thị G



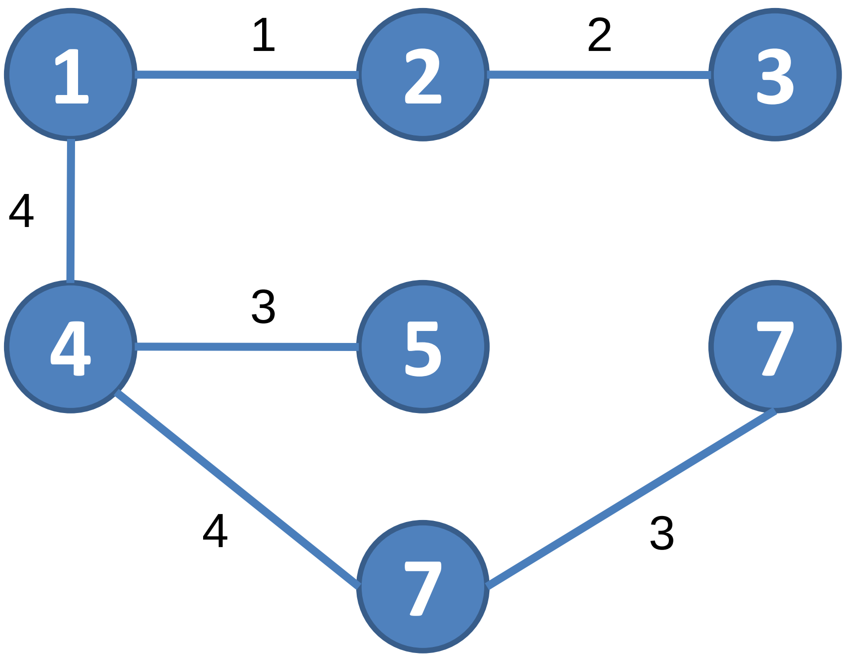
MST T

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Kết quả

- MST T= (V_T,E_T)
  - V_T=V = {1,2,3,4,5,6,7}
  - E_T={(1,2),
    (2,3),
    (1,4),
    (4,5),
    (4,7),
    (6,7),}

- W(T) = 17 (Trọng số cây T)



MST T

Cài đặt

- Biểu diễn G qua ma trận trọng số cạnh

$$c[i][j] = \begin{cases} \text{Trọng số } (i,j); \text{Nếu } (i,j) \text{ tồn tại;} \\ 0; i = j \\ \infty; \text{Nếu } (i,j) \text{ không tồn tại;} \end{cases}$$

- Mảng **Closest[i]**: Giá trị của nó đỉnh kề gần i nhất.
- Mảng **lowcost[i]**: cho trọng số của cạnh (**i,closest[i]**).

```
void Prim (Mat c)
{
        double Lowcost[MAX];
        int Closest[MAX];
        int i,j,k,Min;
        for (i=2;i<=n;i++)
        {
                Lowcost[i] = c[1][i];
                Closest[i] = 1;
        }
        for (i=2;i<=n;i++)
        {
                Min = Lowcost[2];
                k = 2;
                for ( j=3; j<=n; j++) // Chọn k
                        if (Lowcost[j] < Min )
                        {
                                Min = Lowcost[j];
                                k = j;
                        }
                cout<<endl<<k<<closest[k];
                lowcost[k] = ∞;
                // Khởi động lại Closest[], Lowcost[]
                for ( j=2; j<=n; j++)
                        if ( (c[k][ j] < lowcost[j]) && (lowcost[j] < ∞))
                        {
                                lowcost[j] = c[k][j];
                                closest[j] = k;
                        }
        }
}
```

Bài toán

- Cho đơn đồ thị G=(V,E)
  - V: Tập các đỉnh
  - E: Tập các cạnh
- Cây T gọi là cây bao trùm của G nếu T là đồ thị con của G và chứa tất cả các đỉnh thuộc G (có số đỉnh =V)

➡ **Tìm cây bao trùm có trọng số nhỏ nhất (Minimal Spanning Tree) MST**

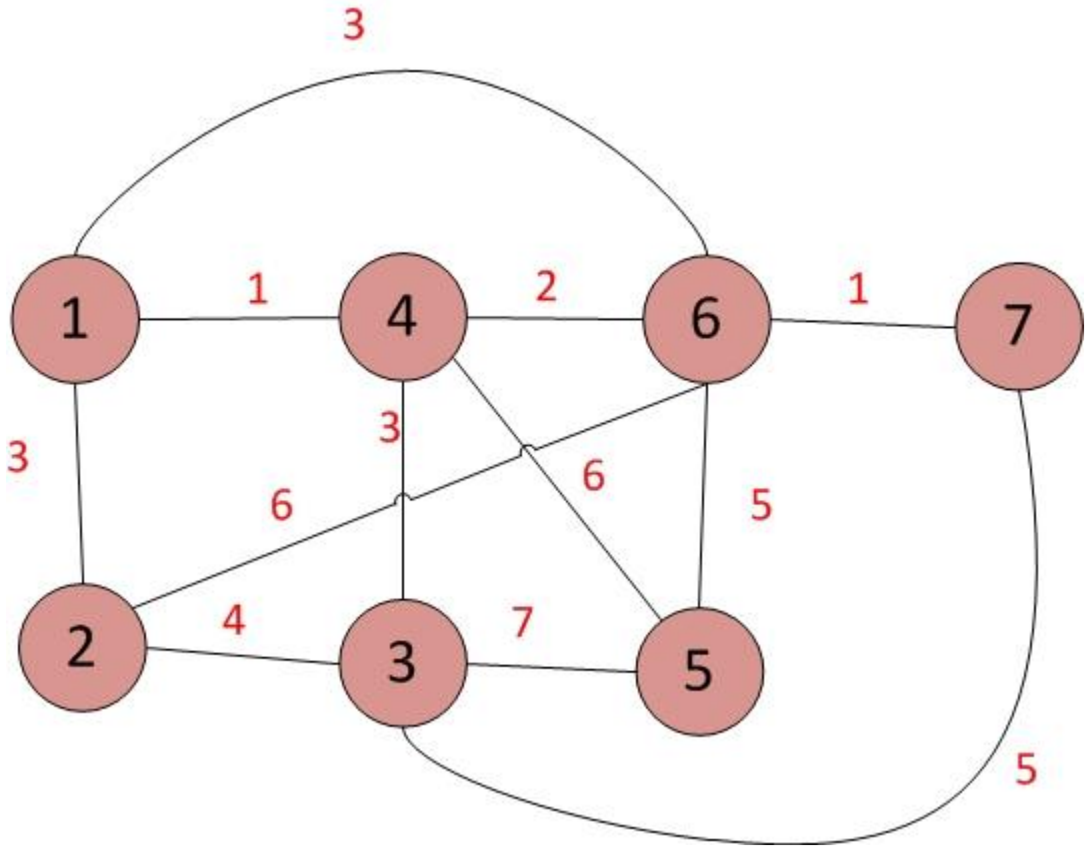
Thuật toán Kruskal

- $T = G_T(V_T, E_T)$ là cây khung tối thiểu cần tìm
- Khi G có **n** đỉnh thì T có **n-1** cạnh
- Ý tưởng (tham lam): Xây dựng tập n-1 cạnh của T theo nguyên tắc:
  - Khởi tạo $E_T=\{\}$, $V_T = V$
  - Xét lần lượt các cạnh có trọng số nhỏ đến lớn nếu không tạo thành chu trình trong T thì thêm cạnh đó vào E_T.

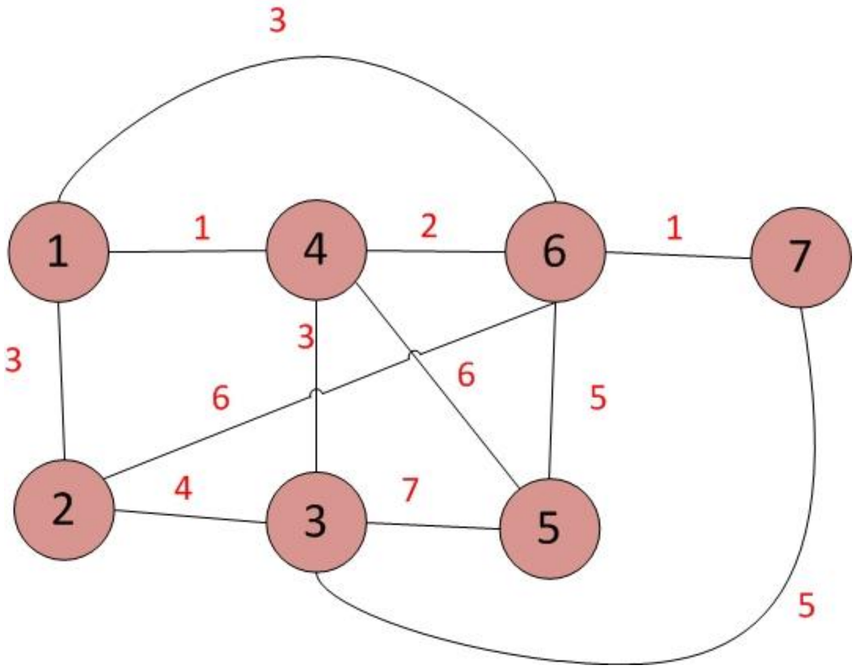
Minh họa

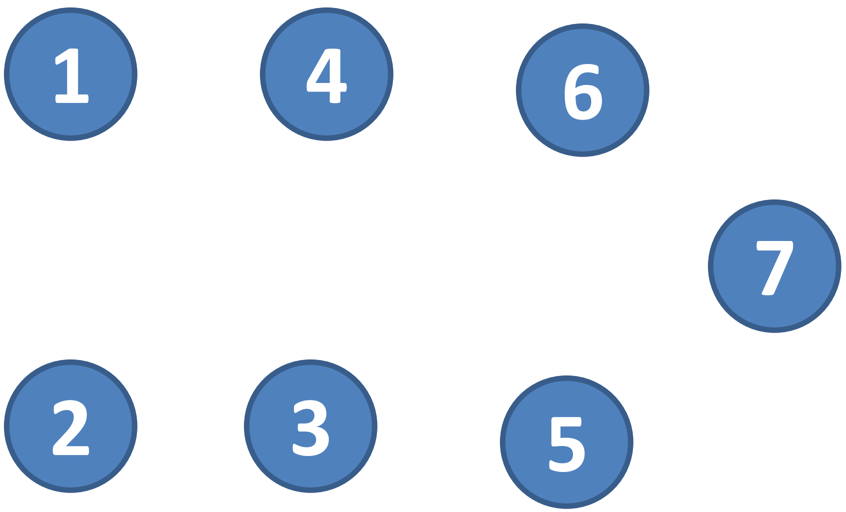
- Cho đồ thị

G = (V,E) =



Khởi tạo

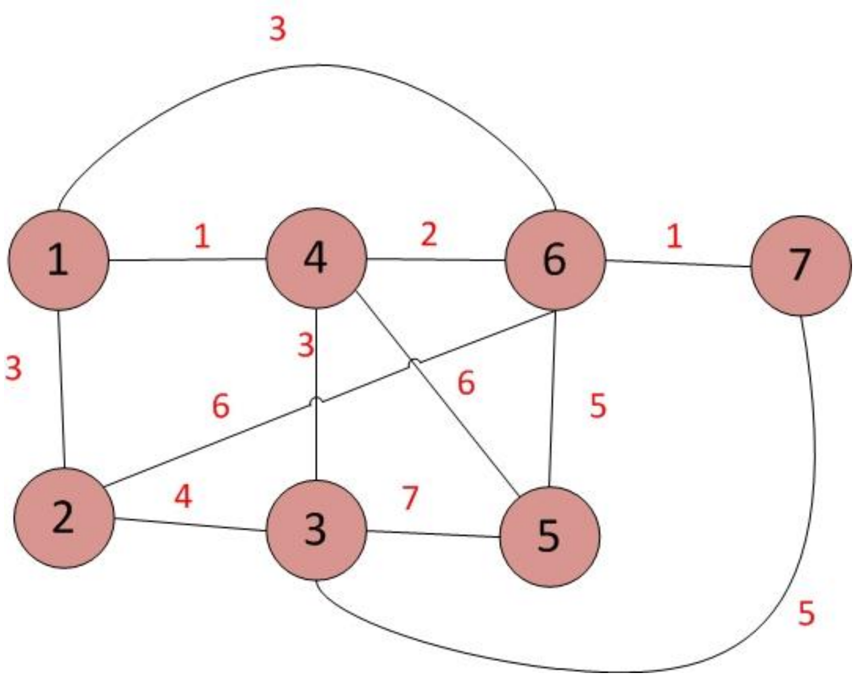


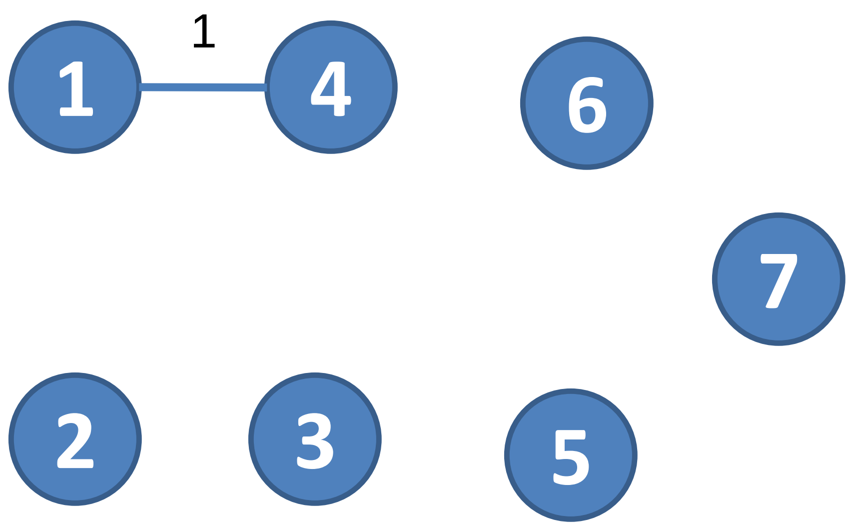


Đồ thị G

MST T

Bước 1

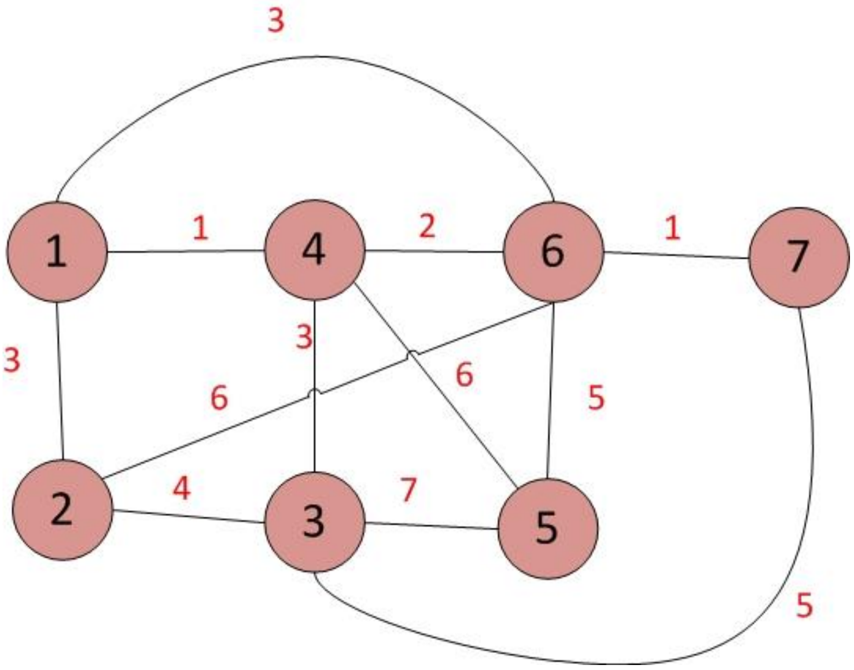


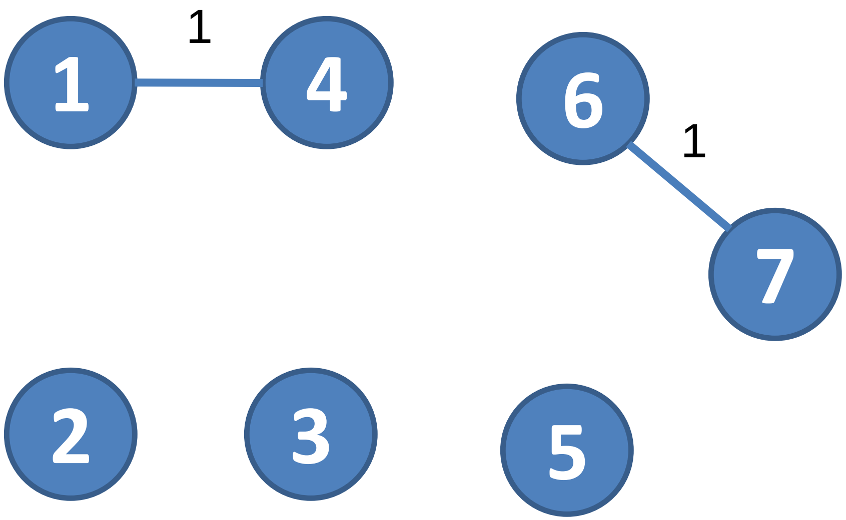


Đồ thị G

MST T

Bước 2

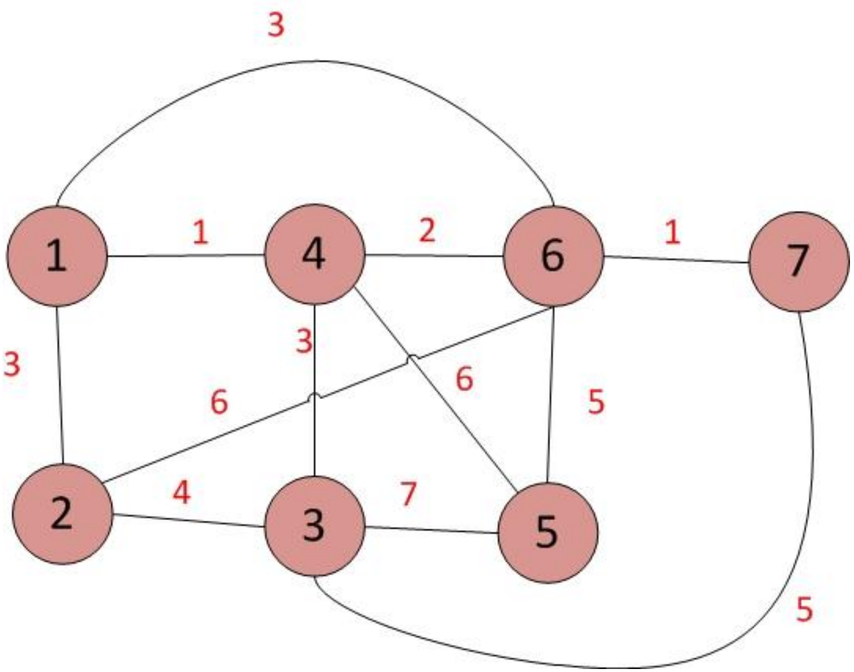




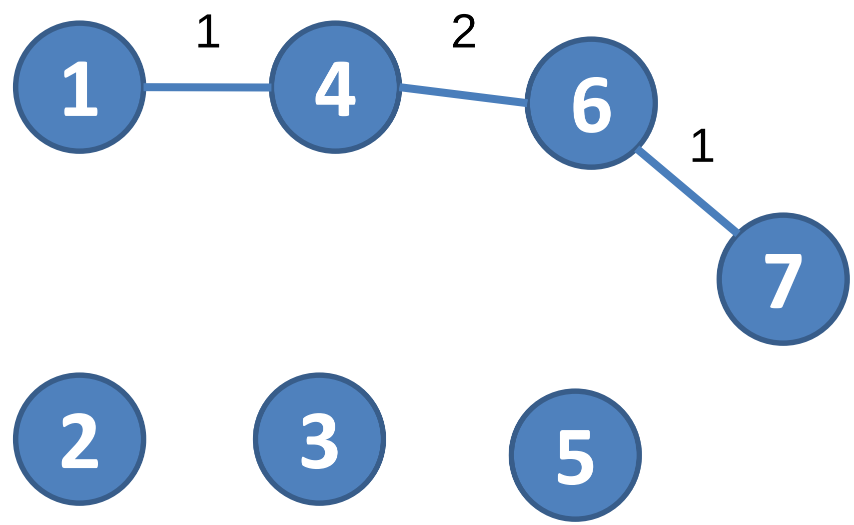
Đồ thị G

MST T

Bước 3



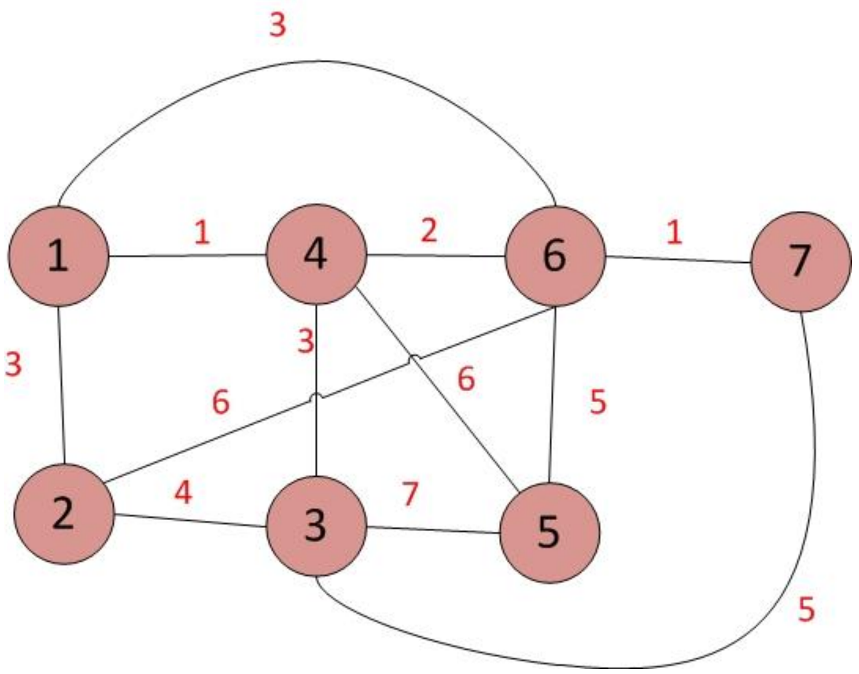
Đồ thị G

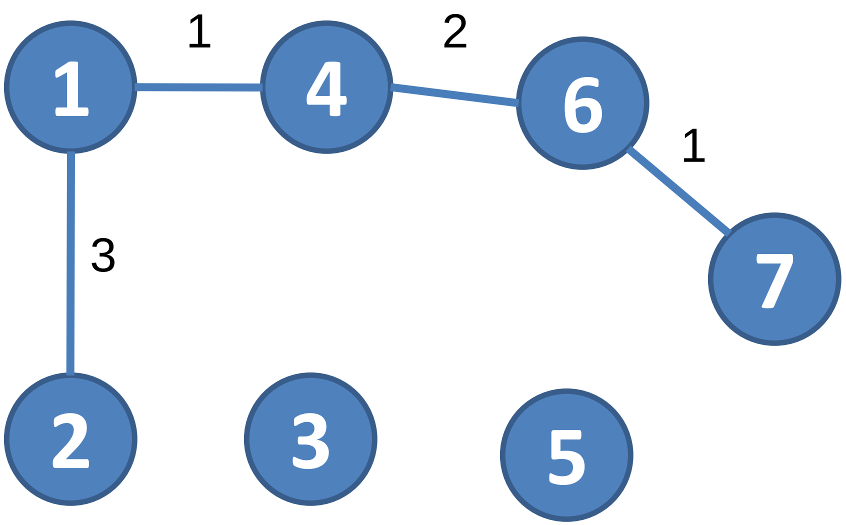


MST T

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Bước 4



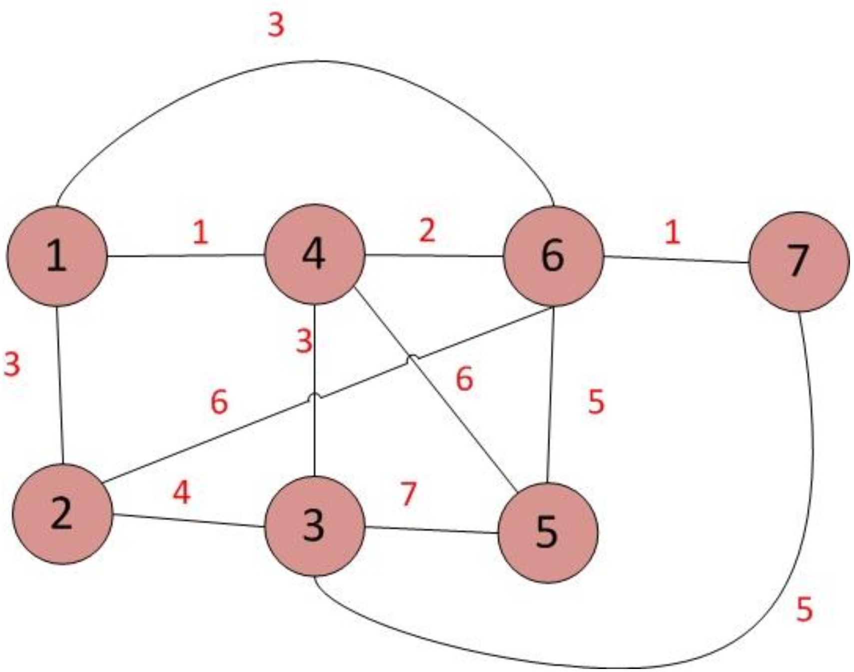


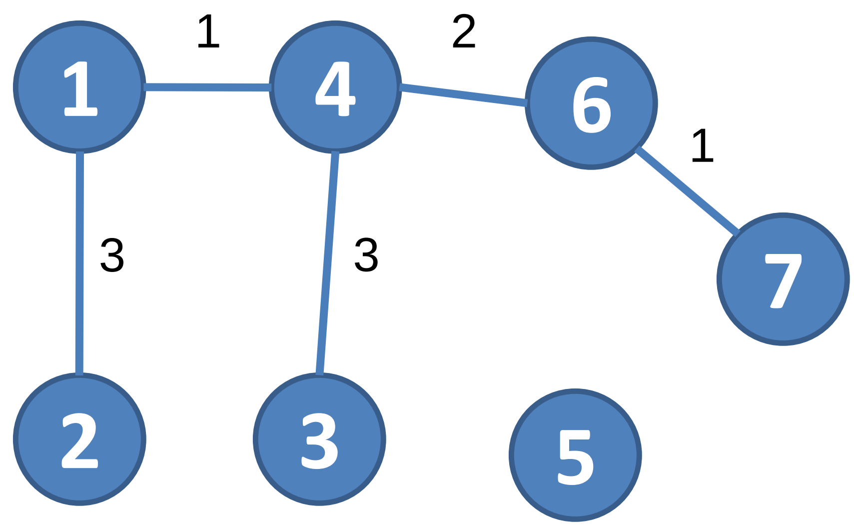
Đồ thị G

MST T

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Bước 5

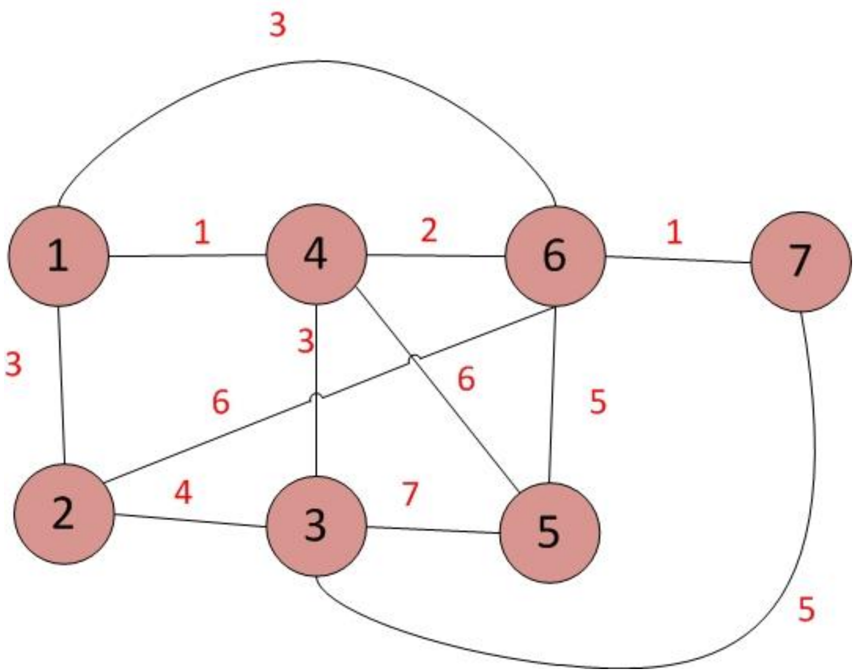


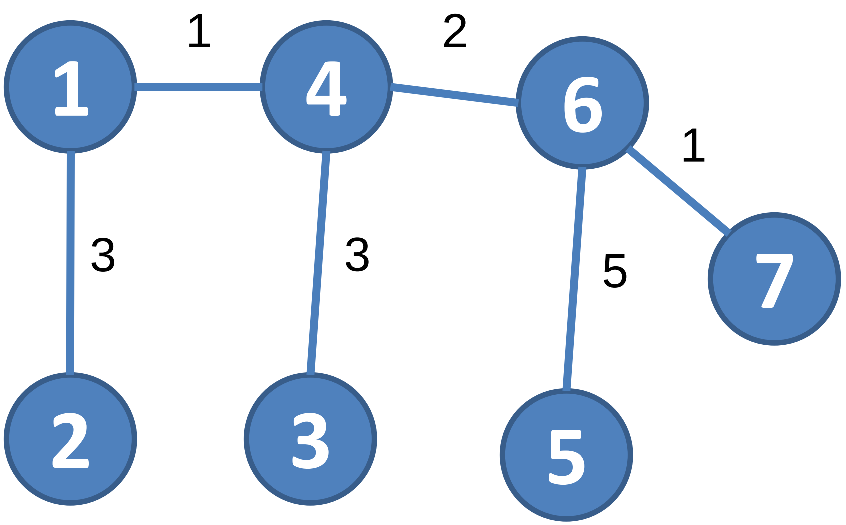


Đồ thị G

MST T

Bước 6





Đồ thị G

MST T

Kết quả

- MST T= (V_T,E_T)
  - V_T=V = {1,2,3,4,5,6,7}
  - ET={(1,4),

    (1,2),

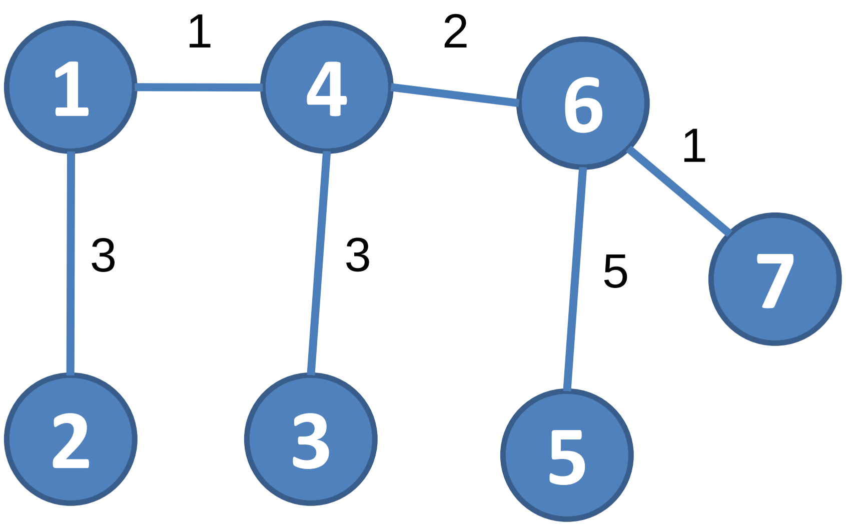
    (3,4),

    (4,6),

    (5,6),

    (6,7),}

  - W(T) = 15 (Trọng số cây T)



MST T

Cài đặt

- Mô tả G bằng ma trận trọng số cạnh A[i,j].
- D mảng 1 chiều, nếu D[i]=k thì đỉnh i thuộc vào cây thứ k, D[i] = 0 thì đỉnh i chưa thuộc vào cây.
- Tìm min {A[i][j] } j = 1..n, i =1..n trừ các cạnh (i,j) mà D[i]=D[j]<>0 (những cạnh đó tạo thành chu trình).
- Thêm cạnh vừa tìm vào cây T, lặp lại bước 2 khi T còn là rừng.

Cài đặt

- Xử lý cạnh (i,j) khi được thêm vào T:
  - Nếu D[i]=D[j]=0, cạnh (i,j) chưa thuộc vào cây nên khi lấy 2 đỉnh này vào tập cạnh ta cho chúng thuộc vào 1 cây mới. Khi đó k=k+1 và D[i]=D[j]=k.
  - Nếu D[i]=0 và D[j]<>0: i chưa thuộc vào T, j thuộc T => Ghép i vào cùng cây chứa j, D[i]=D[j].
  - Nếu D[i]<>0 và D[j]=0: i thuộc vào T, j không thuộc T => Ghép j vào cùng cây chứa i, D[j]=D[i].
  - Nếu D[i]<>D[j] và D[i]<>0, D[j]<>0: i, j thuộc 2 cây khác nhau trong T => Ghép 2 cây thành 1.

Cài đặt

- Xử lý cạnh (i,j) khi được thêm vào T:
  - Nếu D[i]=D[j]=0, cạnh (i,j) chưa thuộc vào cây nên khi lấy 2 đỉnh này vào tập cạnh ta cho chúng thuộc vào 1 cây mới. Khi đó k=k+1 và D[i]=D[j]=k.
  - Nếu D[i]=0 và D[j]<>0: i chưa thuộc vào T, j thuộc T => Ghép i vào cùng cây chứa j, D[i]=D[j].
  - Nếu D[i]<>0 và D[j]=0: i thuộc vào T, j không thuộc T => Ghép j vào cùng cây chứa i, D[j]=D[i].
  - Nếu D[i]<>D[j] và D[i]<>0, D[j]<>0: i, j thuộc 2 cây khác nhau trong T => Ghép 2 cây thành 1.

Cài đặt

- Xử lý cạnh (i,j) khi được thêm vào T:
  - Nếu D[i]=D[j]=0, cạnh (i,j) chưa thuộc vào cây nên khi lấy 2 đỉnh này vào tập cạnh ta cho chúng thuộc vào 1 cây mới. Khi đó k=k+1 và D[i]=D[j]=k.
  - Nếu D[i]=0 và D[j]<>0: i chưa thuộc vào T, j thuộc T => Ghép i vào cùng cây chứa j, D[i]=D[j].
  - Nếu D[i]<>0 và D[j]=0: i thuộc vào T, j không thuộc T => Ghép j vào cùng cây chứa i, D[j]=D[i].
  - Nếu D[i]<>D[j] và D[i]<>0, D[j]<>0: i, j thuộc 2 cây khác nhau trong T => Ghép 2 cây thành 1.

Cài đặt

- Xử lý cạnh (i,j) khi được thêm vào T:
  - Nếu D[i]=D[j]=0, cạnh (i,j) chưa thuộc vào cây nên khi lấy 2 đỉnh này vào tập cạnh ta cho chúng thuộc vào 1 cây mới. Khi đó k=k+1 và D[i]=D[j]=k.
  - Nếu D[i]=0 và D[j]<>0: i chưa thuộc vào T, j thuộc T => Ghép i vào cùng cây chứa j, D[i]=D[j].
  - Nếu D[i]<>0 và D[j]=0: i thuộc vào T, j không thuộc T => Ghép j vào cùng cây chứa i, D[j]=D[i].
  - Nếu D[i]<>D[j] và D[i]<>0, D[j]<>0: i, j thuộc 2 cây khác nhau trong T => Ghép 2 cây thành 1.

Cài đặt

```
typedef struct Egde {
  int x,y;
};
void Kruskal(int **A, int n){
  char *D = new char[n];
  Egde *L = new Egde[n-1];
  int min, Dem = 0, Sum = 0, T = 0, Temp;
  for(int i=0; i<n; i++)
    D[i] = 0;
  do{
    min = MAXINT;
    for( i=0; i<n; i++)
    for(int j=0; j<n; j++)
    if(A[i][j]>0 && min>A[i][j]&& !(D[i]!=0 && D[i]==D[j])) {
        min = A[i][j];
        L[Dem].x = i;
        L[Dem].y = j;
    }
```

```
    /*Tạo ra cây mới*/
    if(D[L[Dem].x] ==0 && D[L[Dem].y] == 0){
      T++;
      D[L[Dem].x] = D[L[Dem].y] = T;
    }
    /*Đưa đỉnh tương ứng vào cây*/
    if(D[L[Dem].x] == 0 && D[L[Dem].y] != 0)
      D[L[Dem].x] = D[L[Dem].y];
    /*Đưa đỉnh tương ứng vào cây*/
    if(D[L[Dem].x] != 0 && D[L[Dem].y] == 0)
      D[L[Dem].y] = D[L[Dem].x];
    /*Ghép 2 cây thành 1 cây mới*/
    if(D[L[Dem].x] != D[L[Dem].y] && D[L[Dem].y]!=0)  {
      Temp = D[L[Dem].x];
      for( i=0; i<n; i++)
      if(Temp==D[i])
        D[i]=D[L[Dem].y];
    }
    Sum+=min;
    Dem++;
  } while(Dem<n-1);
}
```

Nội dung

1. Lược đồ chung
2. Bài toán cái túi
3. Bài toán người du lịch
4. Đường đi ngắn nhất
5. Cây bao trùm nhỏ nhất
6. **Bài toán tô màu**
7. Bài toán các khoảng không giao nhau

Vấn đề

- Suppose that you are responsible for scheduling times for lectures in a university .
- You want to make sure that any two lectures with a common student occur at different times to avoid a conflict.
- We could put the various lectures on a chart and mark with an \X" any pair that has students in common.

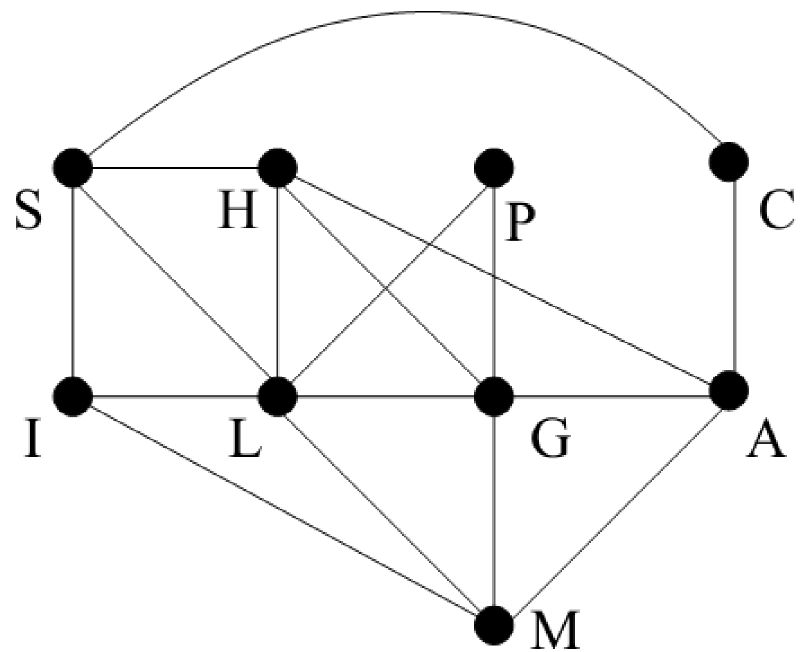
Vấn đề …

Lecture	A	C	G	H	I	L	M	P	S
Astronomy		X	X	X			X		
Chemistry	X								X
Greek	X			X		X	X	X	
History	X		X			X			X
Italian						X	X		X
Latin			X	X	X		X	X	X
Music	X		X		X	X			
Philosophy			X			X			
Spanish		X		X	X	X			

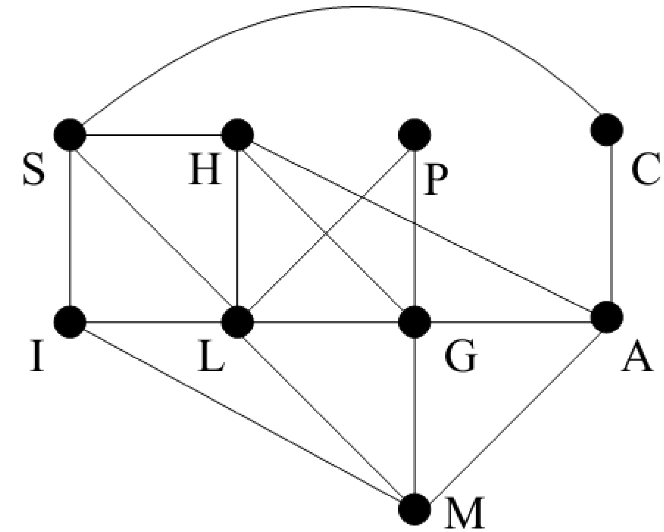
Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

- A more convenient representation of this information is a graph: One vertex for each lecture and in which two vertices are joined if there is a conflict between them

Lecture	A	C	G	H	I	L	M	P	S
Astronomy		X	X	X			X		
Chemistry	X								X
Greek	X			X		X	X	X	
History	X		X			X			X
Italian						X	X		X
Latin			X	X	X		X	X	X
Music	X		X		X	X			
Philosophy			X			X			
Spanish		X		X	X	X			



Bài toán



- Bài toán: Tô mỗi đỉnh 1 màu sao cho 2 đỉnh kề nhau có màu khác nhau. Tìm cách tô tất cả đỉnh của đồ thị với số màu ít nhất.
- Ý nghĩa: Xếp lịch thi cuối kỳ sao cho số buổi cần tổ chức là ít nhất.

Tô màu tham lam

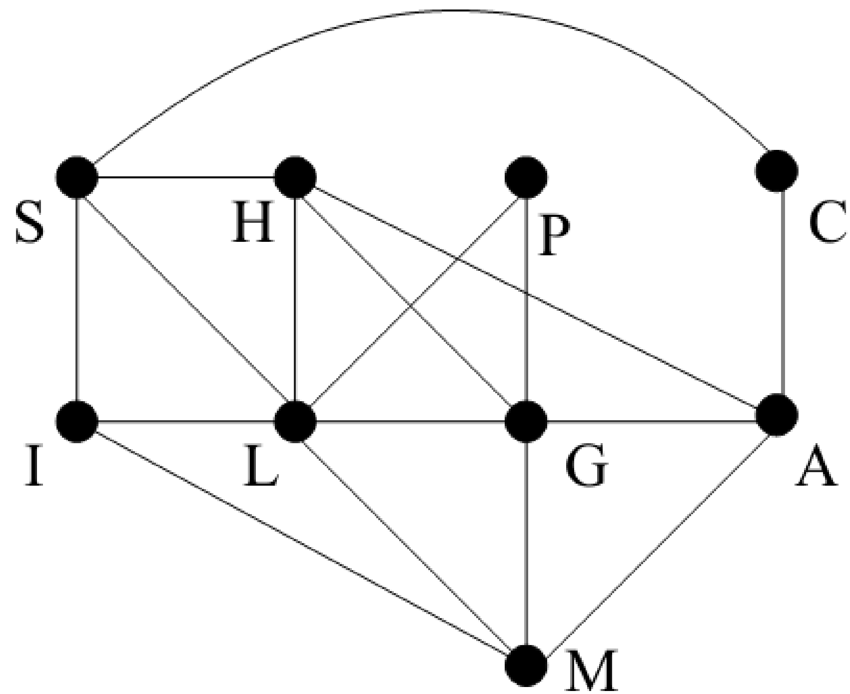
- Ý tưởng
  - Qui ước màu là các số: 1, 2, 3, ...

1. Tô màu một đỉnh bất kỳ với màu 1
2. Với đỉnh **v** chưa tô màu: Tô nó với màu là số nhỏ nhất chưa dùng với các đỉnh kề và đã được tô màu của **v**. (Nếu tất cả các đỉnh kề của **v** đã tô màu -> **v** sẽ được tô với màu mới).
3. Lặp lại bước 2 cho đến khi tất cả các đỉnh được tô màu.

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Minh họa

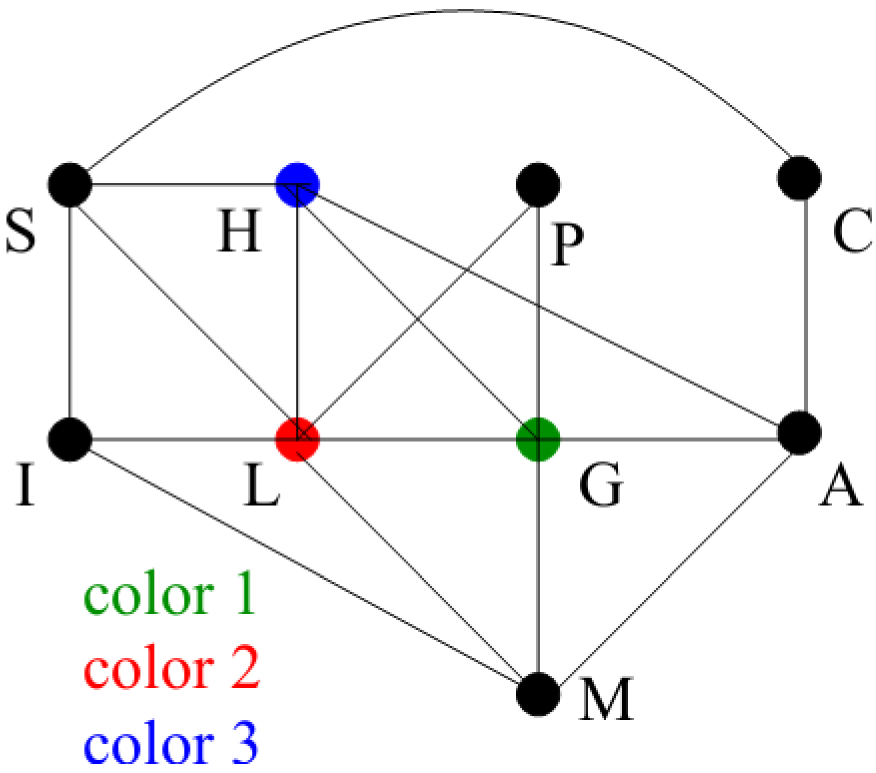
- Tô màu (tham lam) đồ thị sau



- Giả sử tô theo thứ tự: G, L, H , P , M , A, I , S , C

Minh họa …

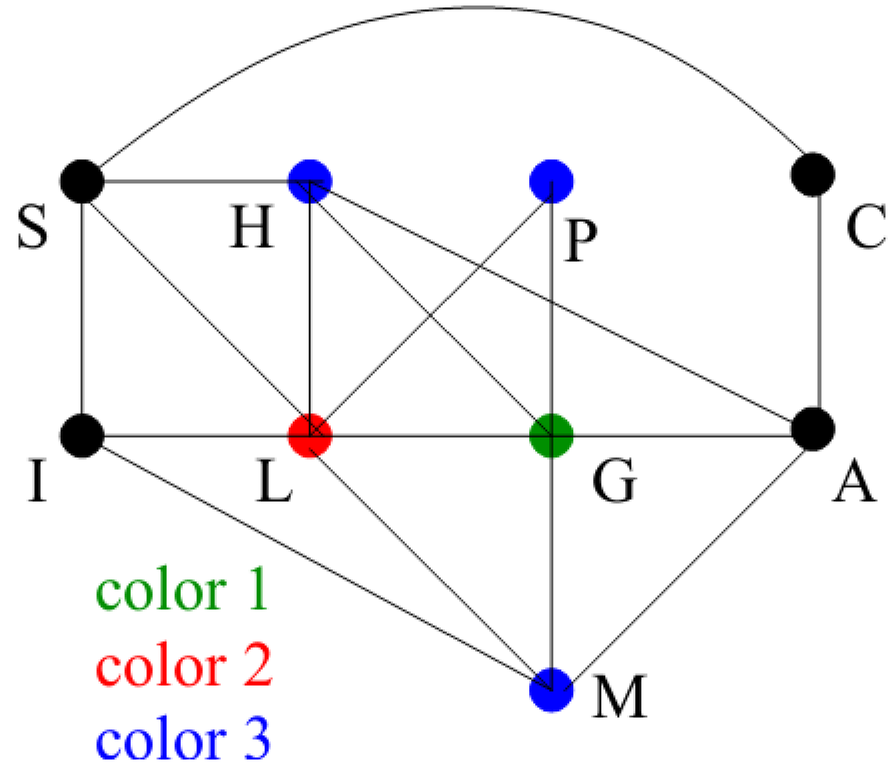
Then we would color G with color 1 (green), L with color 2 (red) since adjacency with G prevents it from receiving color 1 (green), and we color H with color 3 (blue) since adjacency with G and L prevents it from receiving colors 1 and 2 (green and red)



- Tô theo thứ tự: G, L, H , P , M , A, I , S , C

Minh họa …

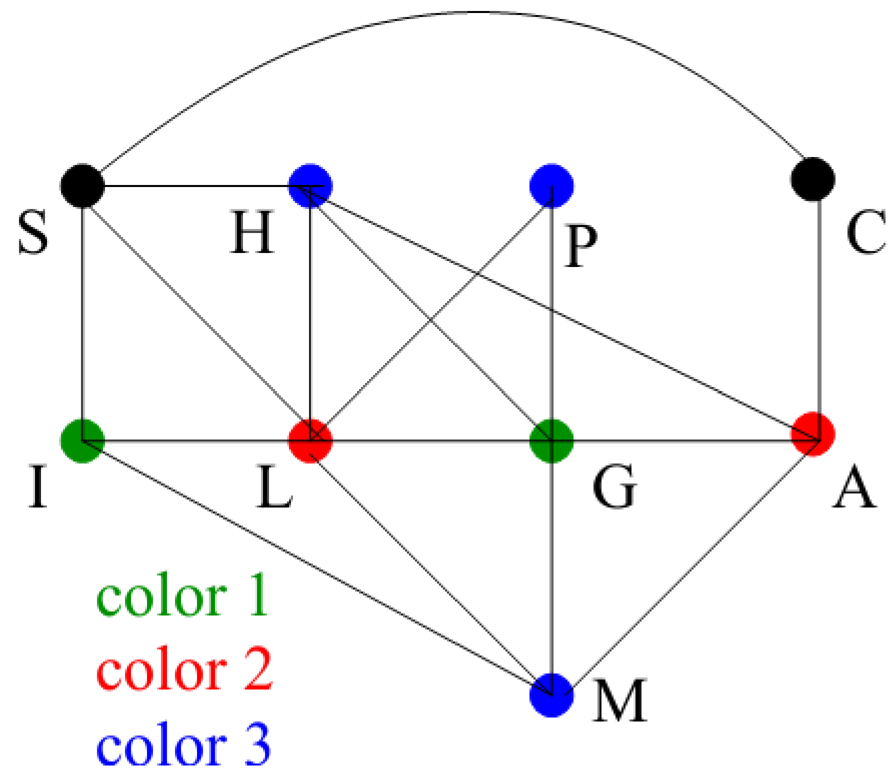
P and M also cannot receive colors 1 and 2 (green and red), so they are given color 3 (blue):



- Tô theo thứ tự: G, L, H , P , M , A, I , S , C

Minh họa …

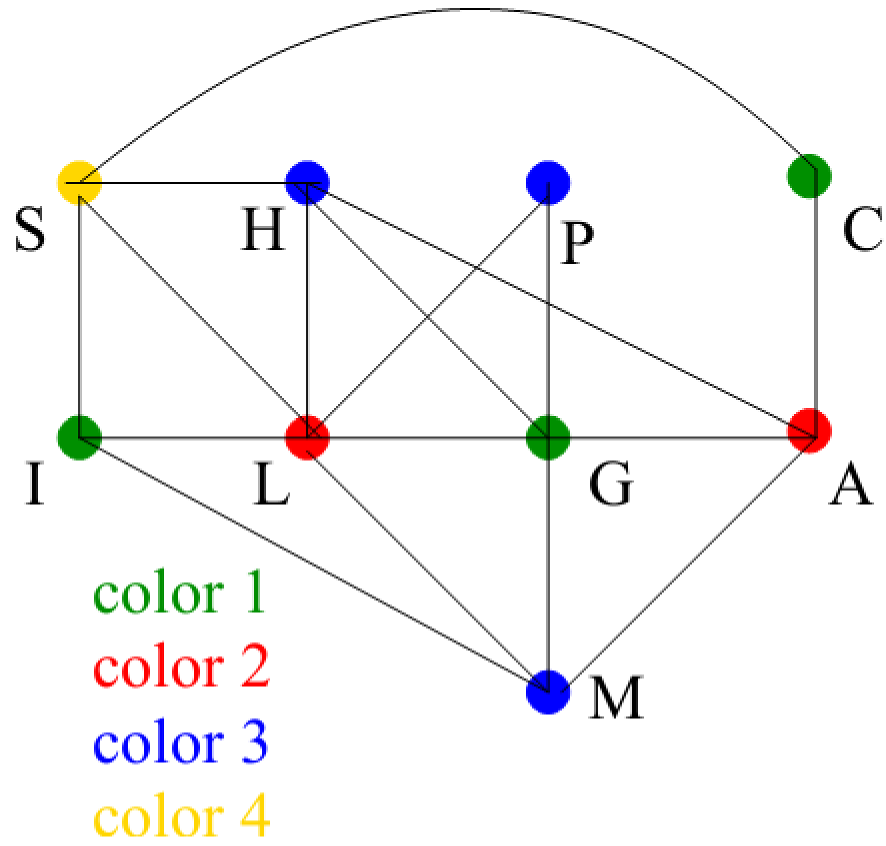
Then A cannot receive colors 1 and 3 (green and blue), so we give it color 2 (red), while I cannot receive colors 2 and 3 (red and blue), so we give it color 1 (green)



- Tô theo thứ tự: G, L, H , P , M , A, I , S , C

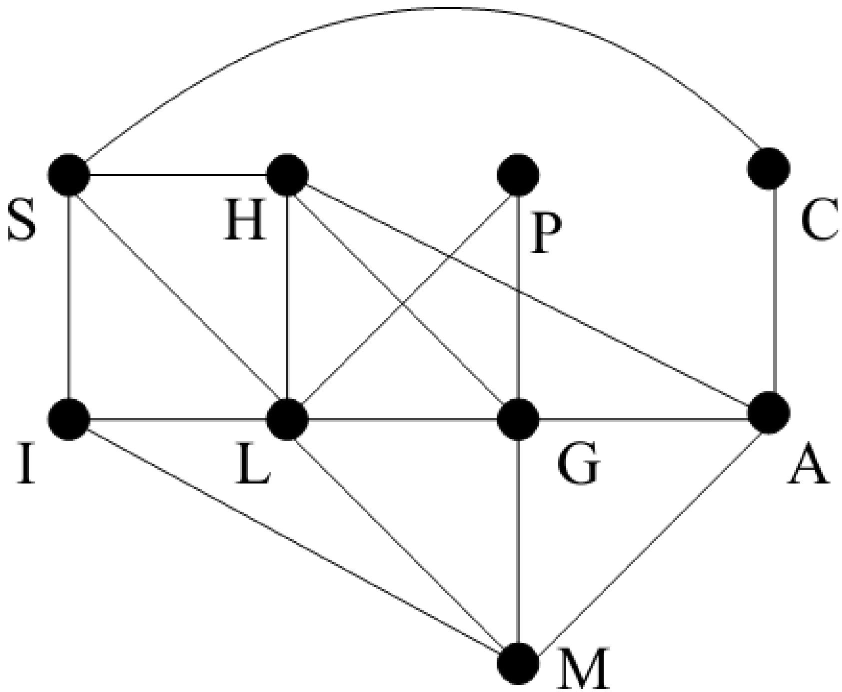
Minh họa …

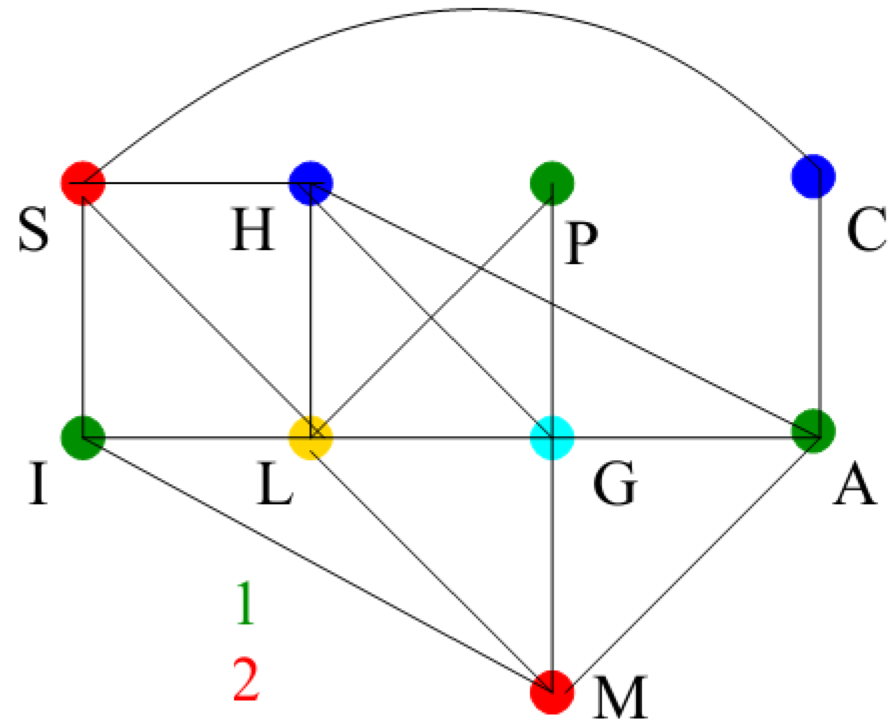
Vertex S cannot receive color 1, 2, or 3, and so we give it color 4 (say , yellow). Vertex C cannot receive color 2 or 4 (red or yellow), so we give it color 1 (green)



- Tô theo thứ tự: G, L, H , P , M , A, I , S , C

Minh họa 2

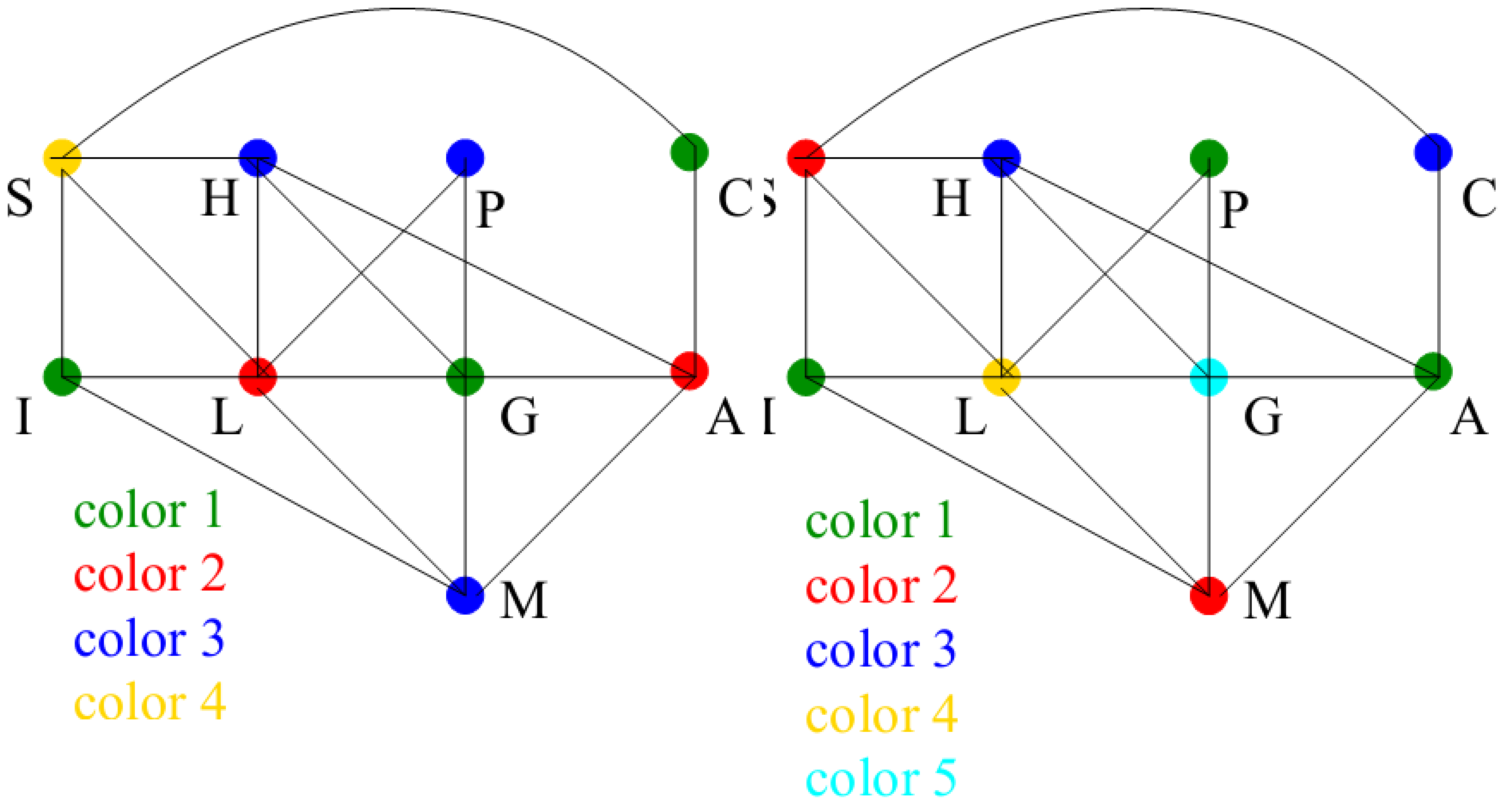




- Tô theo thứ tự: A, I , P , M , S , C , H , L, G

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

Đánh giá



Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Nội dung

1. Lược đồ chung
2. Bài toán cái túi
3. Bài toán người du lịch
4. Đường đi ngắn nhất
5. Cây bao trùm nhỏ nhất
6. Bài toán tô màu
7. **Bài toán các khoảng không giao nhau**

Bài toán

- Có **n** công việc cần thực hiện; **a_i** - thời điểm bắt đầu, **b_i** - thời điểm kết thúc công việc **i** (i=1..n)
- Hãy chọn ra các công việc để một người có thể thực hiện được nhiều việc nhất.
- Các dạng tương tự: Bài toán xếp thời gian biểu cho các hội thảo, bài toán lựa chọn hành động (Activity Selection)…

Thuật toán xếp lịch 1

- Ý tưởng (tham lam):
  - Gọi C là tập các công việc ban đầu
  - Gọi S là tập các công việc được lựa chọn
  - Sắp xếp các công việc theo thứ tự tăng dần của đầu mút trái ($\mathbf{a_i}$).
  - Lần lượt xét các đoạn trong danh sách theo thứ tự đã sắp xếp và bổ sung đoạn thẳng đang xét vào S nếu nó không có điểm chung với bất cứ đoạn nào trong S.

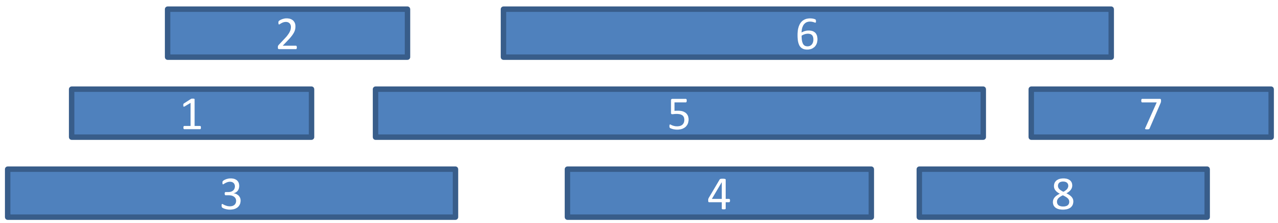
Thuật toán xếp lịch 1 …

- Chi tiết

```
procedure Greedy1;
begin
   S: = ∅; (*  S là tập các đoạn thẳng cần tìm *)
   <Sắp xếp các đoạn trong C theo thứ tự không giảm của mút trái >
   while (C ≠ ∅) do
   begin
        (a_c ,b_c ) ← đoạn đầu tiên trong C;
        C:= C \ (a_c ,b_c );
        if  < (a_c ,b_c ) không giao với bất cứ đoạn nào trong S>
          then S:= S ∪ (a_c ,b_c );
   end;
    <S là tập cần tìm>
end;
```

Minh họa

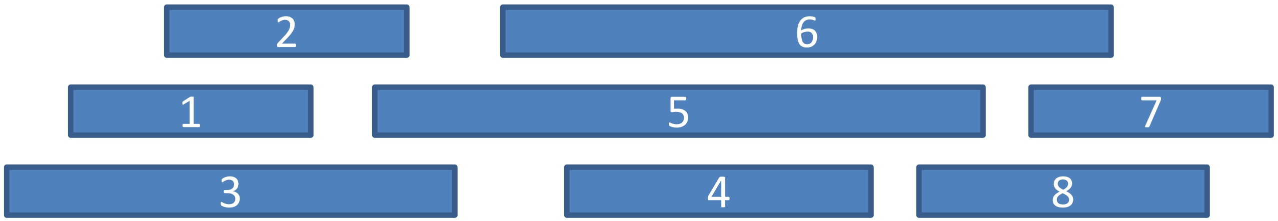
- Cho 8 công việc



- Sắp xếp công việc theo thứ tự tăng dần của nút trái ta được thứ tự các công việc

C = {3, 1, 2, 5, 6, 4, 8, 7}

Khởi tạo

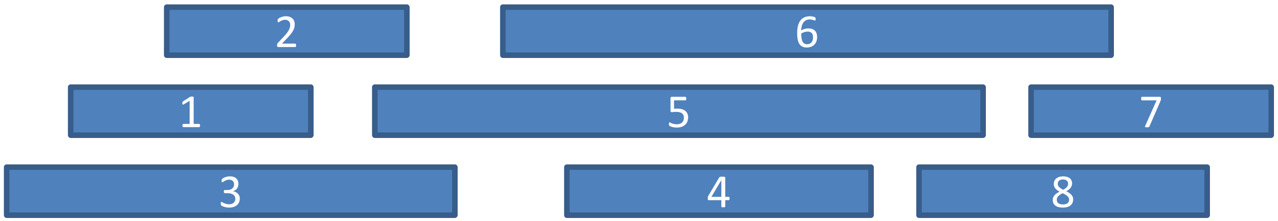


C = {3, 1, 2, 5, 6, 4, 8, 7}

S = {}

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

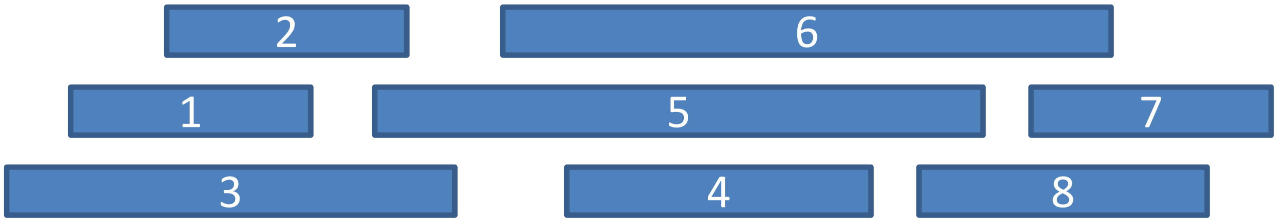
Lặp ...



C = {3, 1, 2, 5, 6, 4, 8, 7}

S = {3, 6}

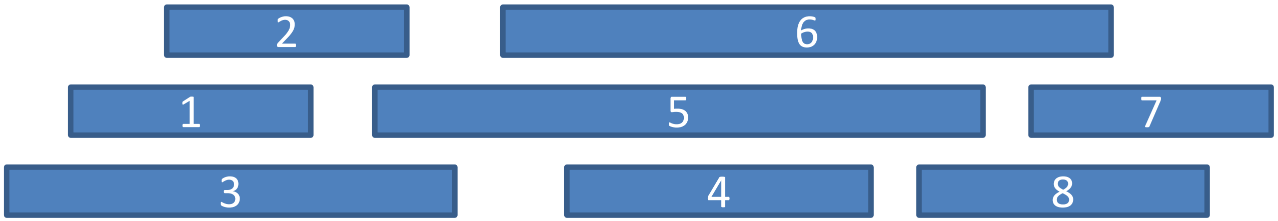
Kết quả TT1



C = {3, 1, 2, 5, 6, 4, 8, 7}

S = {3, 6}

Dễ thấy



C = {3, 1, 2, 5, 6, 4, 8, 7}

Phương án S = {1, 5, 7}

tốt hơn **S = {3, 6}**

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Thuật toán xếp lịch 2

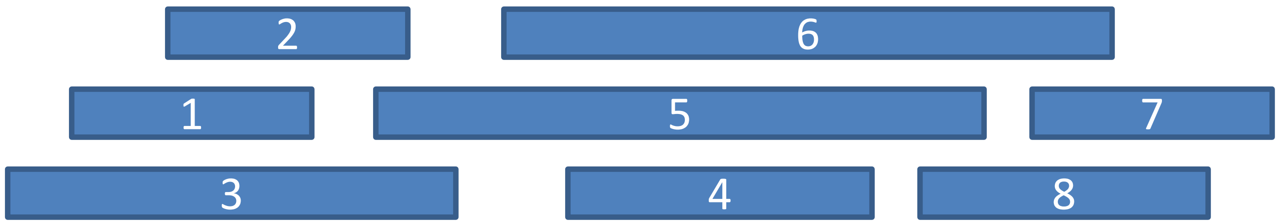
- Ý tưởng (tham lam):
  - Gọi C là tập các công việc ban đầu
  - Gọi S là tập các công việc được lựa chọn
  - Sắp xếp các công việc theo thứ tự tăng dần của thời gian thực hiện công việc (**b_i - a_i**).
  - Lần lượt xét các đoạn trong danh sách theo thứ tự đã sắp xếp và bổ sung đoạn thẳng đang xét vào S nếu nó không có điểm chung với bất cứ đoạn nào trong S.

Thuật toán xếp lịch 3

- Ý tưởng (tham lam):
  - Gọi C là tập các công việc ban đầu
  - Gọi S là tập các công việc được lựa chọn
  - Sắp xếp các công việc theo thứ tự không giảm của đầu mút phải (**b_i**).
  - Lần lượt xét các đoạn trong danh sách theo thứ tự đã sắp xếp và bổ sung đoạn thẳng đang xét vào S nếu nó không có điểm chung với bất cứ đoạn nào trong S.

Minh họa

- Cho 8 công việc

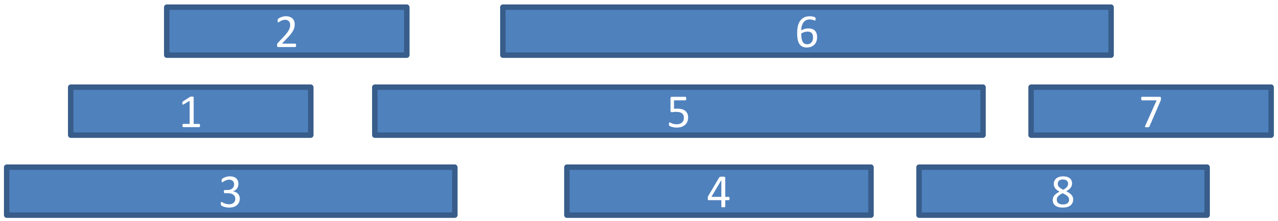


- Sắp xếp công việc theo thứ tự không giảm của mút phải ta được thứ tự các công việc

C = {1, 2, 3, 4, 5, 6, 8, 7}

Biên soạn: Hà Đại Dương, duonghd@mta.edu.vn

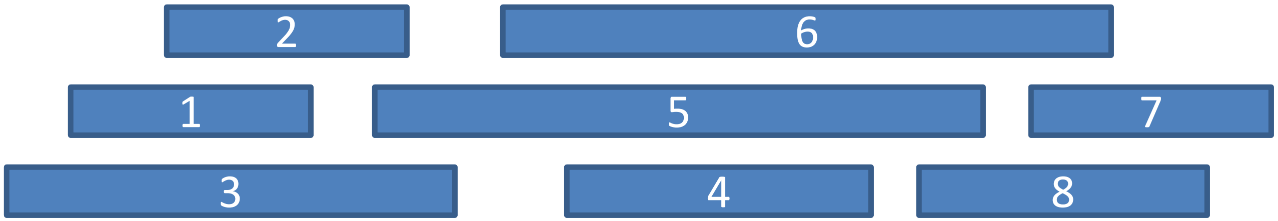
Khởi tạo



C = {1, 2, 3, 4, 5, 6, 8, 7}

S = {}

Lặp …

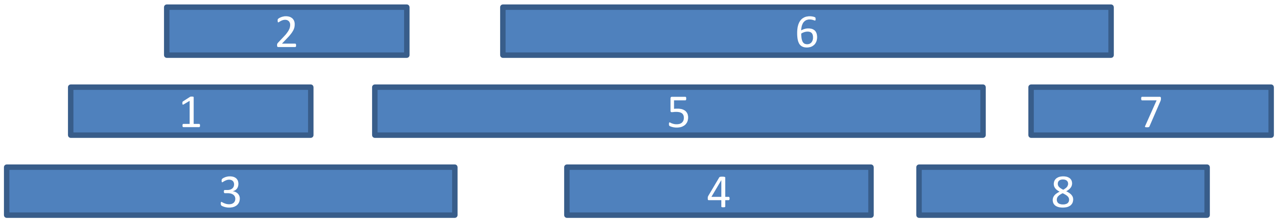


C = {1, 2, 3, 4, 5, 6, 8, 7}

S = {1, 4, 8}

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Kết quả TT3



C = {1, 2, 3, 4, 5, 6, 8, 7}

S = {1, 4, 8}

Biên soạn: Hà Đại Dương,
duonghd@mta.edu.vn

Cài đặt

- ACTIONSELECTION3(a[i], b[i]):
  - Sort (a[i],b[i]) in increasing order by b[i]
  - S = {1}
  - t = 1
  - **for** i = 2 to n

    if b[t]≤a[i] //C[i] *does not confict with* C[t]

    t = i

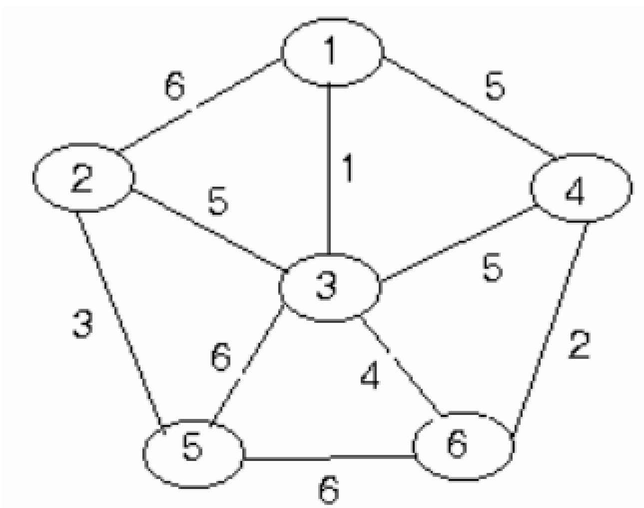
    S = S ∪ {i}
  - return S

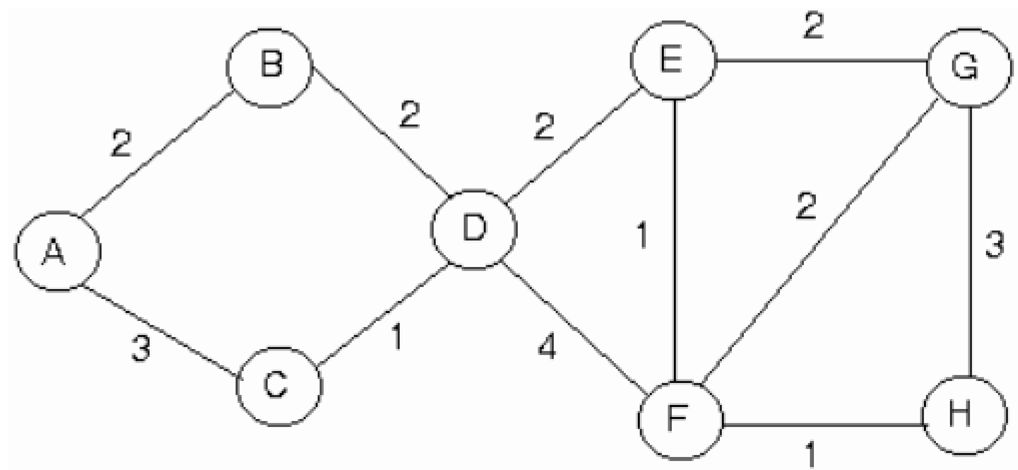
Đánh giá

- Độ phức tạp T(n) = ?
- **Mệnh đề**: Thuật toán xếp lịch 3 cho lời giải tối ưu của bài toán

Bài tập

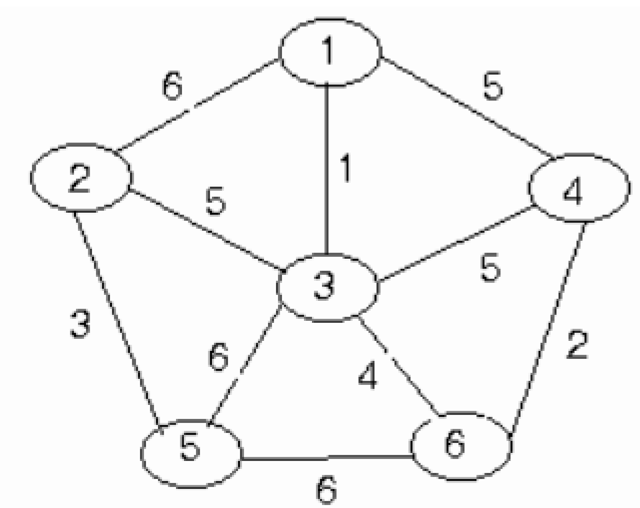
1. Thực hiện từng bước giải thuật Prim trên các đồ thị sau:

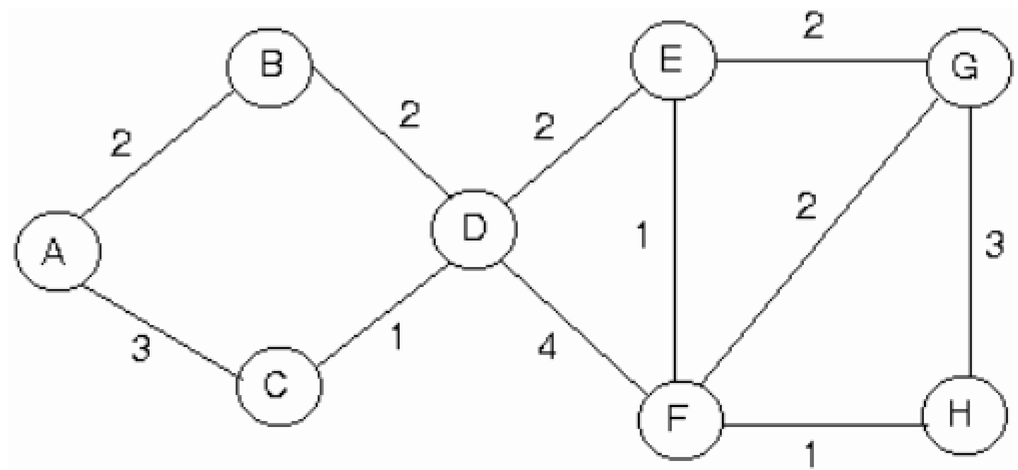




Bài tập

2. Mô tả chi tiết thuật toán Kruskal và thực hiện từng bước giải thuật đó trên các đồ thị sau và so sánh kết quả với bài 1





Bài tập

3. Cài đặt thuật toán Prim. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
4. Cài đặt thuật toán Kruskal. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
5. Cài đặt thuật toán xếp lịch theo ý tưởng tham lam. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.
6. Cài đặt thuật toán tô màu đồ thị. Đánh giá độ phức tạp bằng thực nghiệm và so sánh với lý thuyết.

Nội dung đã học

1. Lược đồ chung
2. Bài toán cái túi
3. Bài toán người du lịch
4. Đường đi ngắn nhất
5. Cây bao trùm nhỏ nhất
6. Bài toán tô màu
7. Bài toán các khoảng không giao nhau