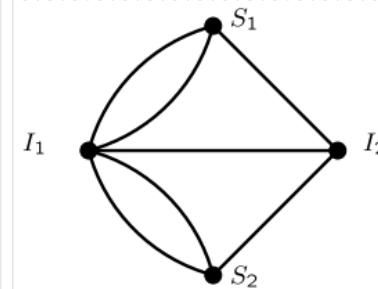
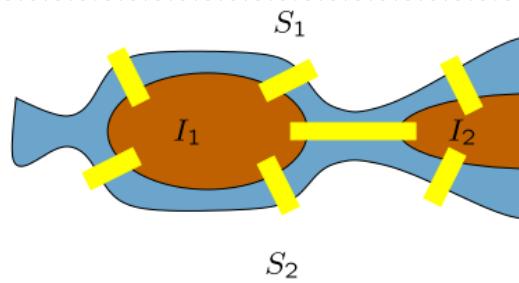


LÝ THUYẾT ĐỒ THỊ

Cây & Cây khung tối thiểu

Phạm Nguyên Khang
BM. Khoa học máy tính, CNTT
pnkhang@cit.ctu.edu.vn

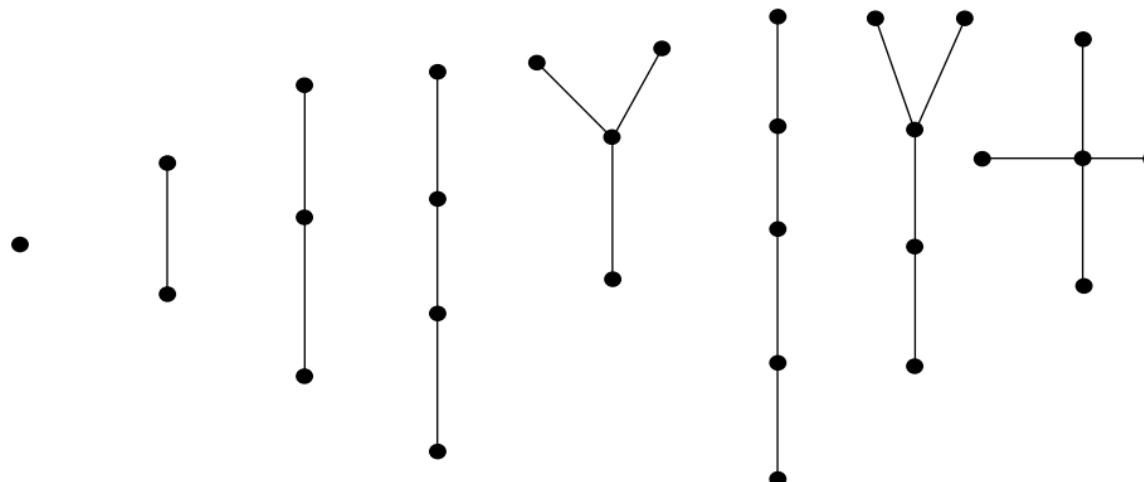


Cần Thơ, 2017

Cây vô hướng

- **Định nghĩa:**

- Cây (tree): đồ thị **vô hướng** **liên thông** và **không có chu trình**.
 - T là cây $\Rightarrow T$ là đồ thị đơn (simple graph)
- Rừng (forest): đồ thị **vô hướng** **không có chu trình**.



Cây vô hướng

- Định lý 1
 - Đồ thị G là cây \Leftrightarrow giữa mỗi cặp đỉnh của G có đúng 1 đường đi (path).
- C/M:
 - \leq
 - mọi cặp đỉnh của G đều có đường đi \Rightarrow G **liên thông**
 - G **không có chu trình** vì nếu G có chu trình, ví dụ: $u \dots \rightarrow v \rightarrow \dots u$ thì sẽ có ít nhất 2 đường đi từ u đến v (trái với giả thiết).
 - \geq
 - G là cây \Rightarrow G liên thông \Rightarrow giữa mỗi cặp đỉnh của G có ít nhất 1 đường đi.
 - Giả có 2 đường đi khác nhau từ $u \rightarrow v \Rightarrow$ G chứa chu trình (trái với GT G là 1 cây) \Rightarrow Giữa mỗi cặp đỉnh có đúng 1 đường đi.

Cây vô hướng

- Định lý 2
 - Cây có n đỉnh sẽ có $n - 1$ cung.
- C/M:
 - Quy nạp theo n
 - Định lý hiển nhiên đúng với $n = 1, 2, 3$
 - G/S ĐL đúng với tất cả các cây có nhiều nhất là k đỉnh
 - C/M ĐL đúng với cây có $k+1$ đỉnh
 - Xét cây T có $k+1$ đỉnh và $e = (u,v)$ là 1 cung của T .
 - Theo định lý 1, đường đi từ u đến v là cung e .
 - Nếu xoá e , cây T sẽ trở nên không liên thông và có 2 BPLT T_1 có n_1 đỉnh, m_1 cung và T_2 có n_2 đỉnh, m_2 cung.
 - $n_1 \leq k$ và $n_2 \leq k$. Theo GTQN: $m_1 = n_1 - 1$ và $m_2 = n_2 - 1$
 - $\Rightarrow m = m_1 + m_2 + 1 = n_1 - 1 + n_2 - 1 + 1 = (k+1) - 1$

Cây vô hướng

- Định lý 3
 - Đồ thị liên thông n đỉnh, $n - 1$ cung là 1 cây
- C/M:
 - Gọi G là đồ thị liên thông n đỉnh, $n-1$ cung
 - Ta C/M G không chứa chu trình (bằng phản chứng)
 - G/S G chứa chu trình c. Xoá 1 cung trong chu trình này thì G vẫn còn liên thông. Tiếp tục như thế ta sẽ thu được cây H có n đỉnh.
 - Theo định lý 2 thì H có $n - 1 \Rightarrow$ số cung của $G >$ số cung của H hay $n-1 > n-1$ (vô lý)
 - Vậy G không có chu trình $\Rightarrow G$ là 1 cây.

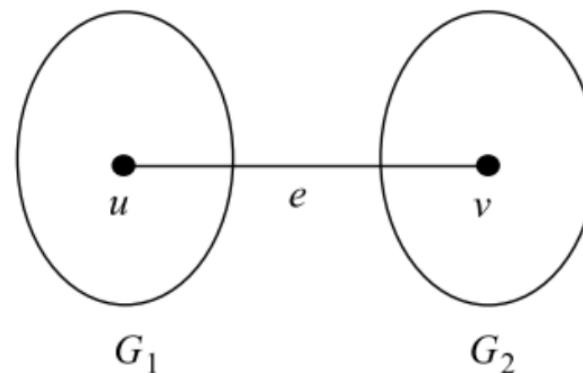
Cây vô hướng

- **Định nghĩa**
 - Đồ thị liên thông tối thiểu (minimally connected)
 - Đồ thị liên thông và nếu xoá một cung bất kỳ thì nó không còn liên thông nữa.
 - Đồ thị liên thông tối thiểu không chứa chu trình
- **Định lý 4:**
 - Đồ thị G là cây $\Leftrightarrow G$ liên thông tối thiểu
 - \Leftarrow
 - G liên thông tối thiểu $\Rightarrow G$ không chứa chu trình $\Rightarrow G$ là cây
 - \Rightarrow
 - G là cây $\Rightarrow G$ không chứa chu trình \Rightarrow xoá 1 cung bất kỳ sẽ làm G không còn liên thông nữa $\Rightarrow G$ liên thông tối thiểu.

Cây vô hướng

- Định lý 5:
 - Đồ thị G có n đỉnh, $n-1$ cung, không chứa chu trình thì liên thông.
- C/M: phản chứng
 - G/S G không liên thông $\Rightarrow G$ có ít nhất 2 BPLT không chứa chu trình G_1 và G_2 . Ta có thể G/S G chỉ có 2 BPLT.
 - Thêm cung e nối 2 đỉnh u (thuộc G_1) và v (thuộc G_2) $\Rightarrow G + \{e\}$ liên thông và không chứa chu trình $\Rightarrow G + \{e\}$ là 1 cây n đỉnh và n cung (trái với ĐL2)

- Vì thế, G liên thông



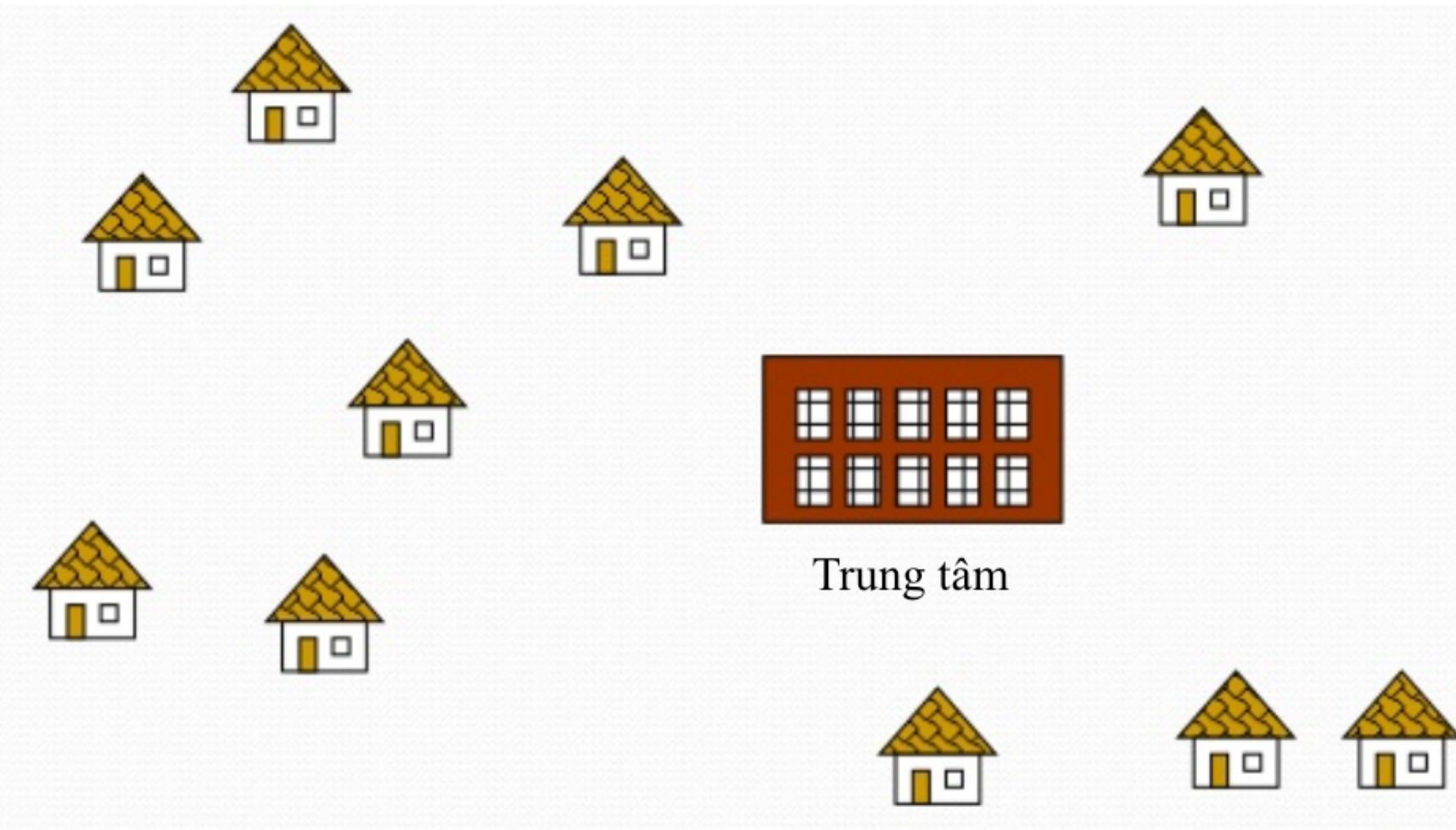
Cây vô hướng

- Định lý 6:
 - Cây có ít nhất 2 đỉnh thì sẽ có ít nhất 2 đỉnh treo (đỉnh bậc 1)
- C/M1:
 - Gọi n là số đỉnh của cây T ($n > 1$) \Rightarrow số cung = $n-1$.
 - \Rightarrow Tổng bậc các đỉnh = $2(n-1)$
 - T liên thông \Rightarrow không có đỉnh bậc 0 $\Rightarrow \deg(u) \geq 1$
 - Bài tập: Hãy tiếp tục C/M T có ít nhất 2 đỉnh bậc 1.
- C/M2:
 - Quy nạp theo n (xem như bài tập)

Cây vô hướng

- Định lý 7:
 - 1 rừng n đỉnh, k cây sẽ có $n - k$ cung.
- C/M:
 - Gọi T_1, T_2, \dots, T_k là các cây trong rừng F.
 - T_1 có n_1 đỉnh, n_1-1 cung, T_2 có n_2 đỉnh n_2-1 cung, ...
 - \Rightarrow Số cung của F = $n_1-1 + n_2-1 + \dots = n - k$.

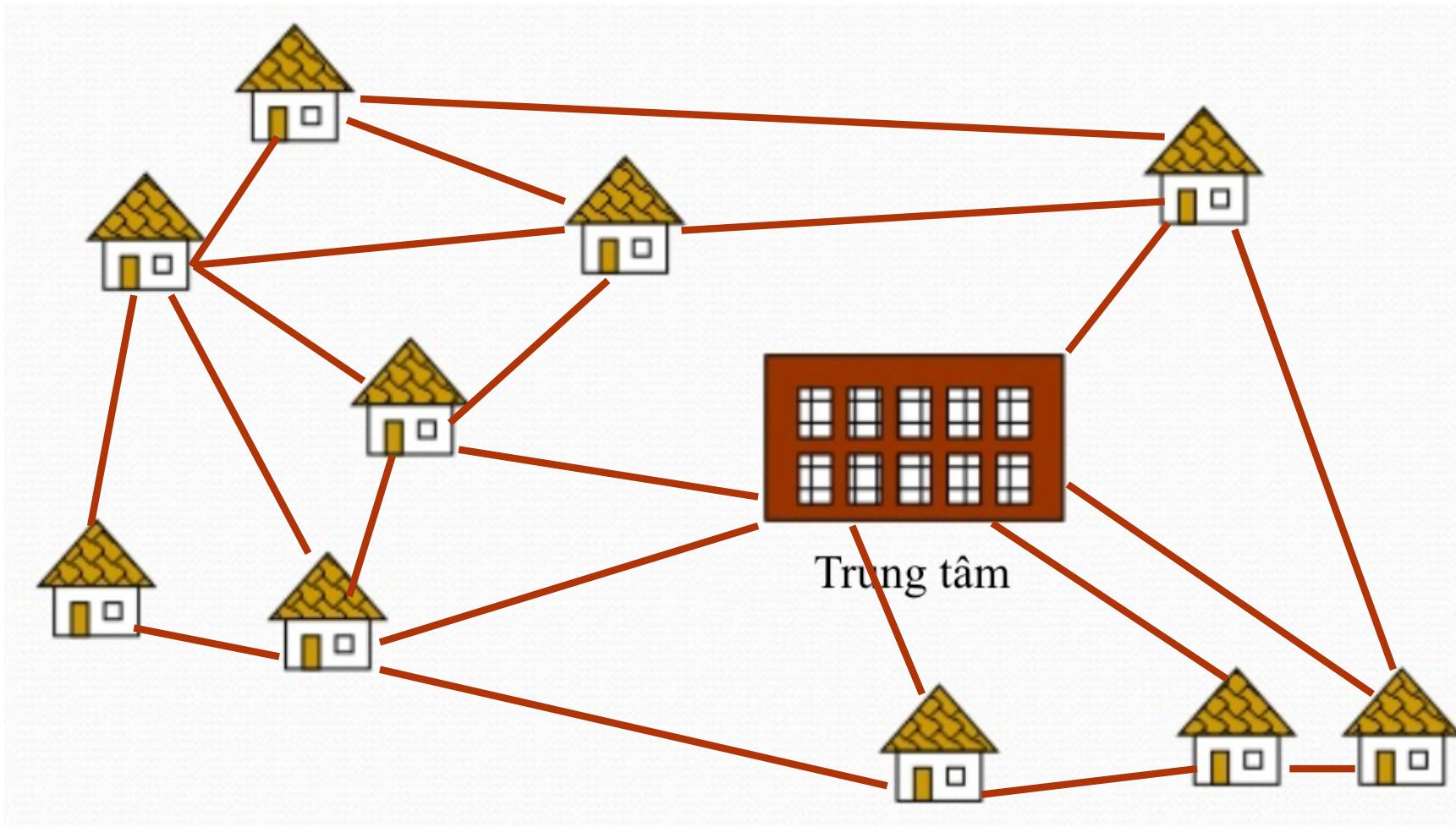
Bài toán xây dựng đường



Bài toán xây dựng đường



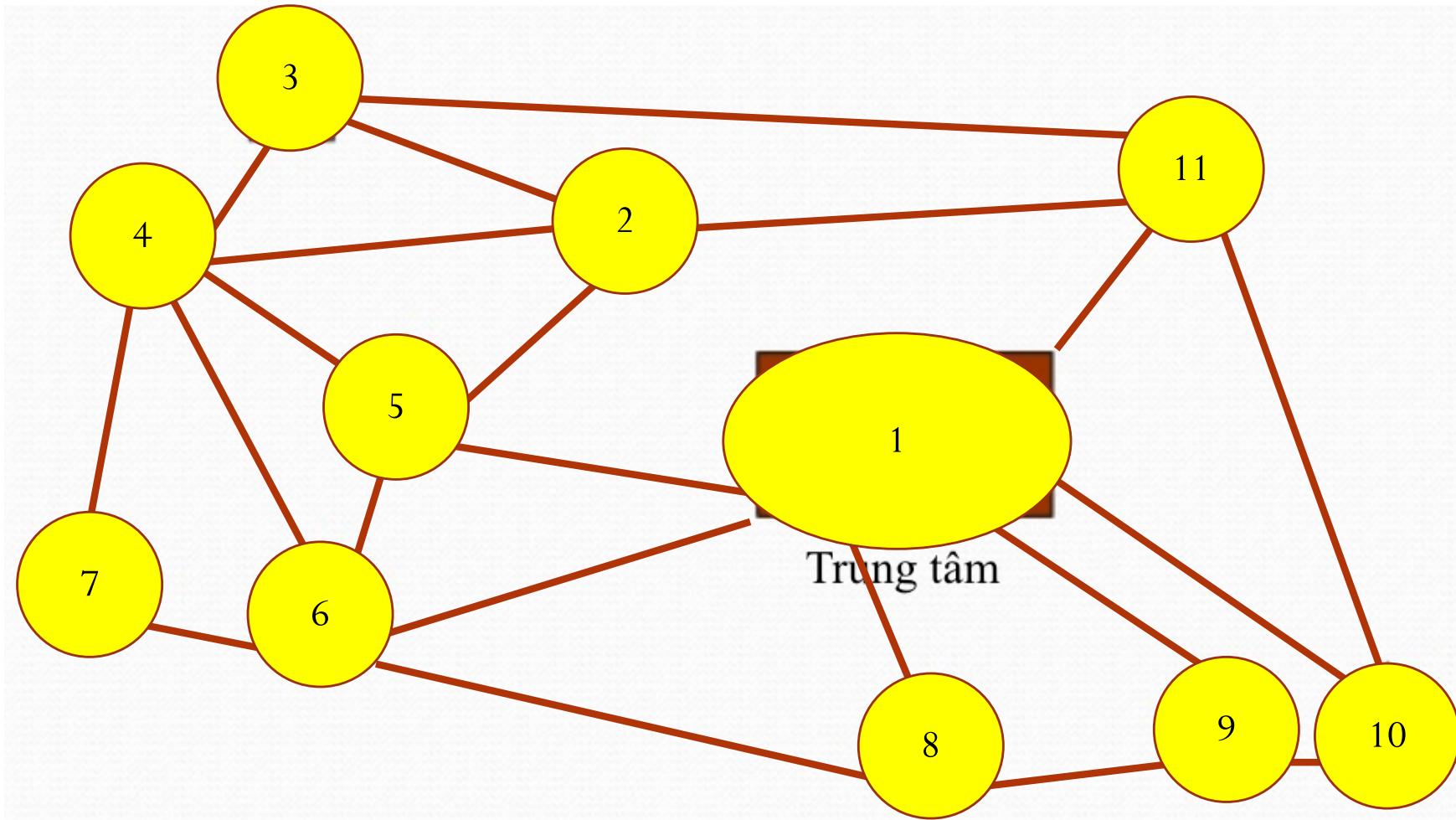
Bài toán xây dựng đường



Bài toán xây dựng đường

- Mô hình về đồ thị
 - Đỉnh: nhà/trung tâm, gọi chung là địa điểm
 - Cung: đường (dự kiến) nối 2 địa điểm

Bài toán xây dựng đường



Bài toán xây dựng đường



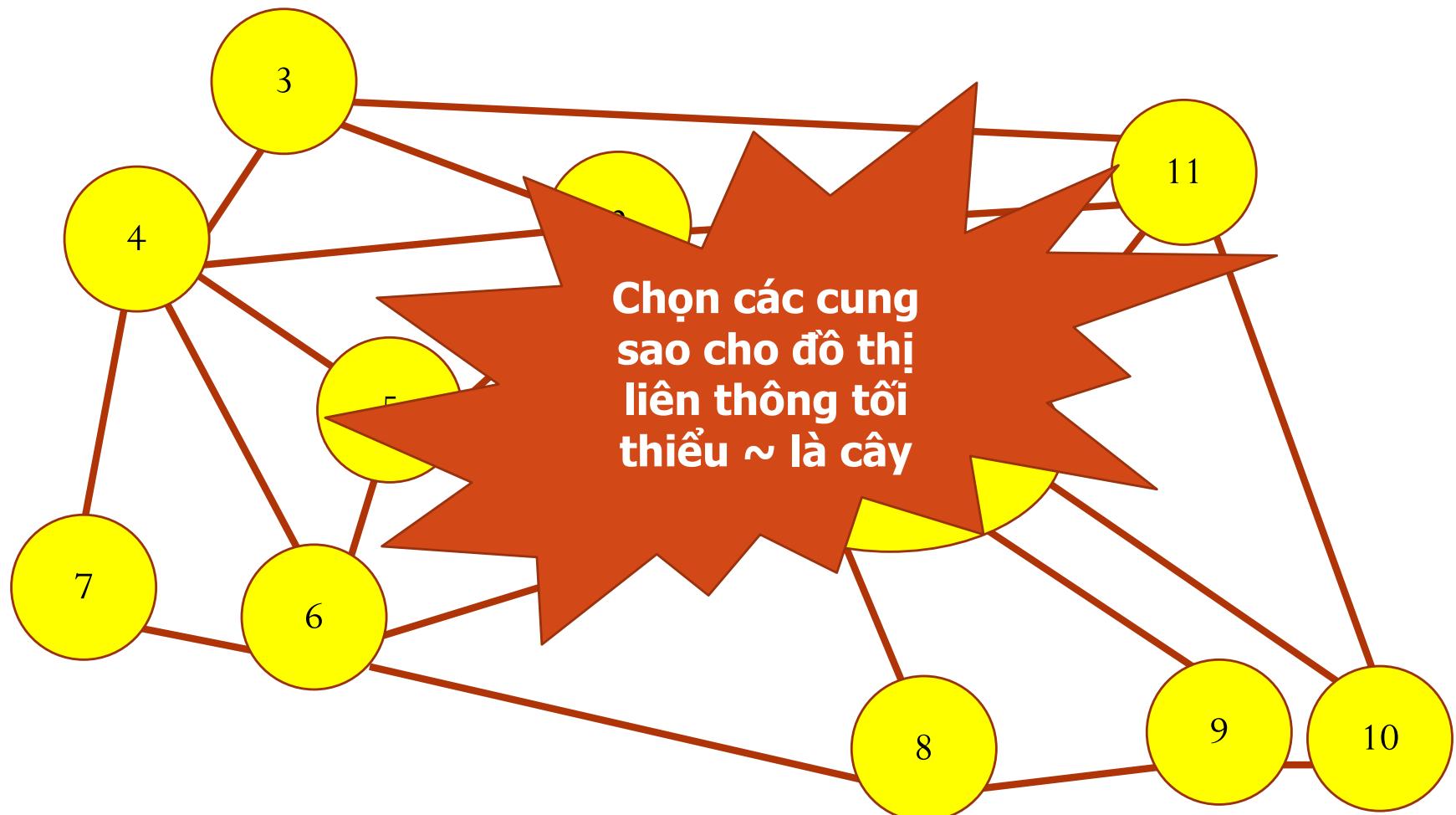
Bài toán xây dựng đường



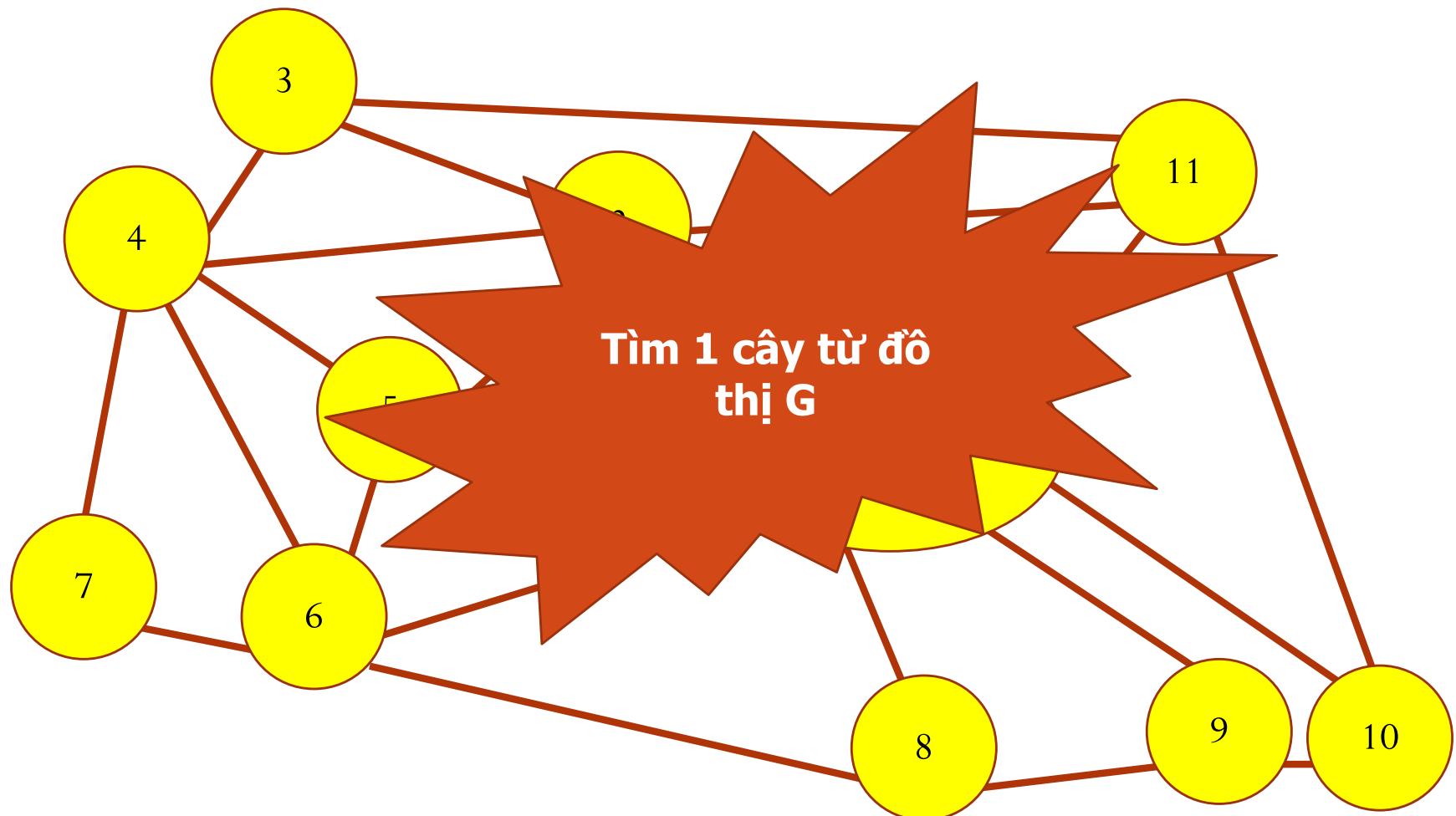
Bài toán xây dựng đường



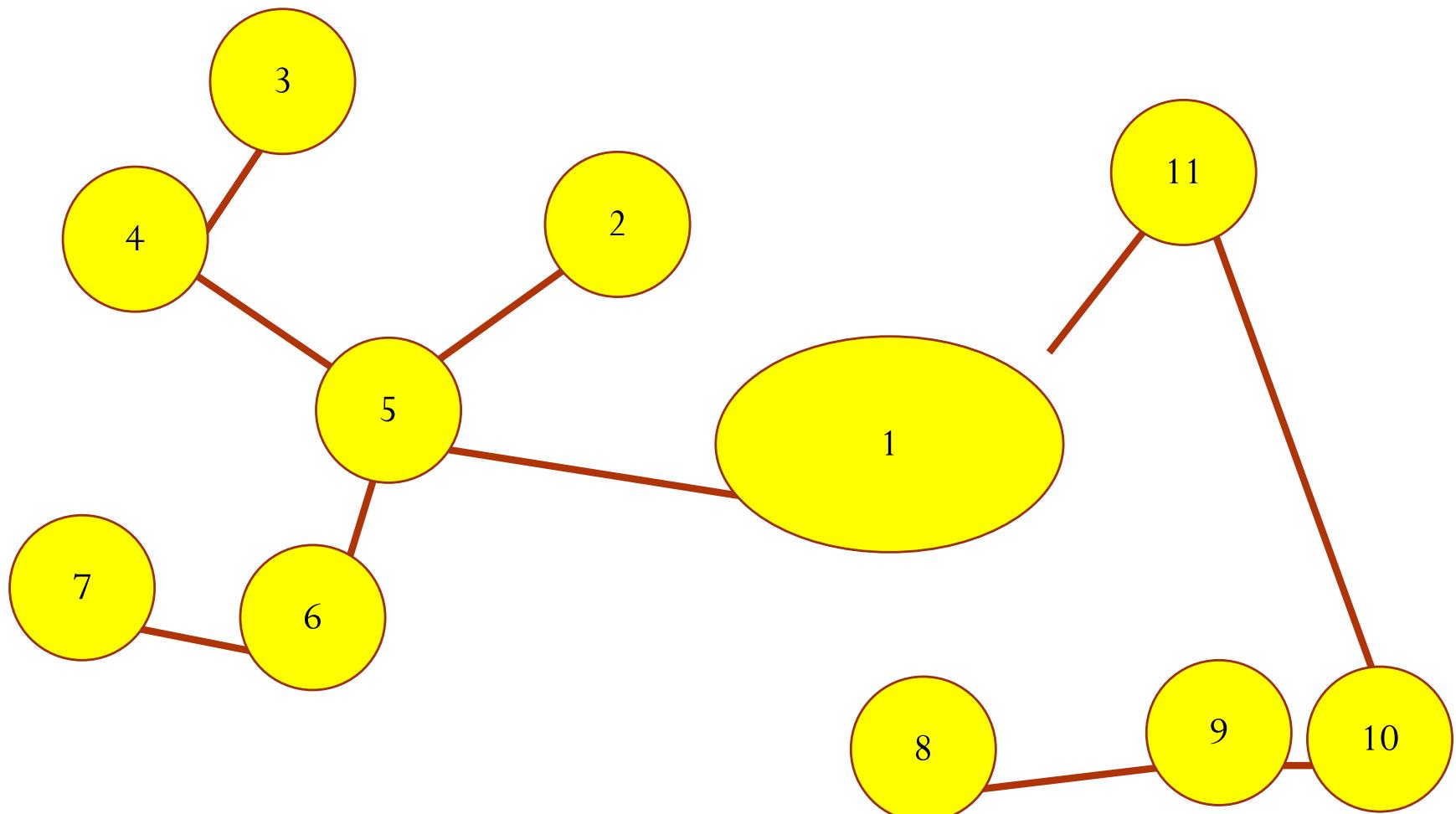
Bài toán xây dựng đường



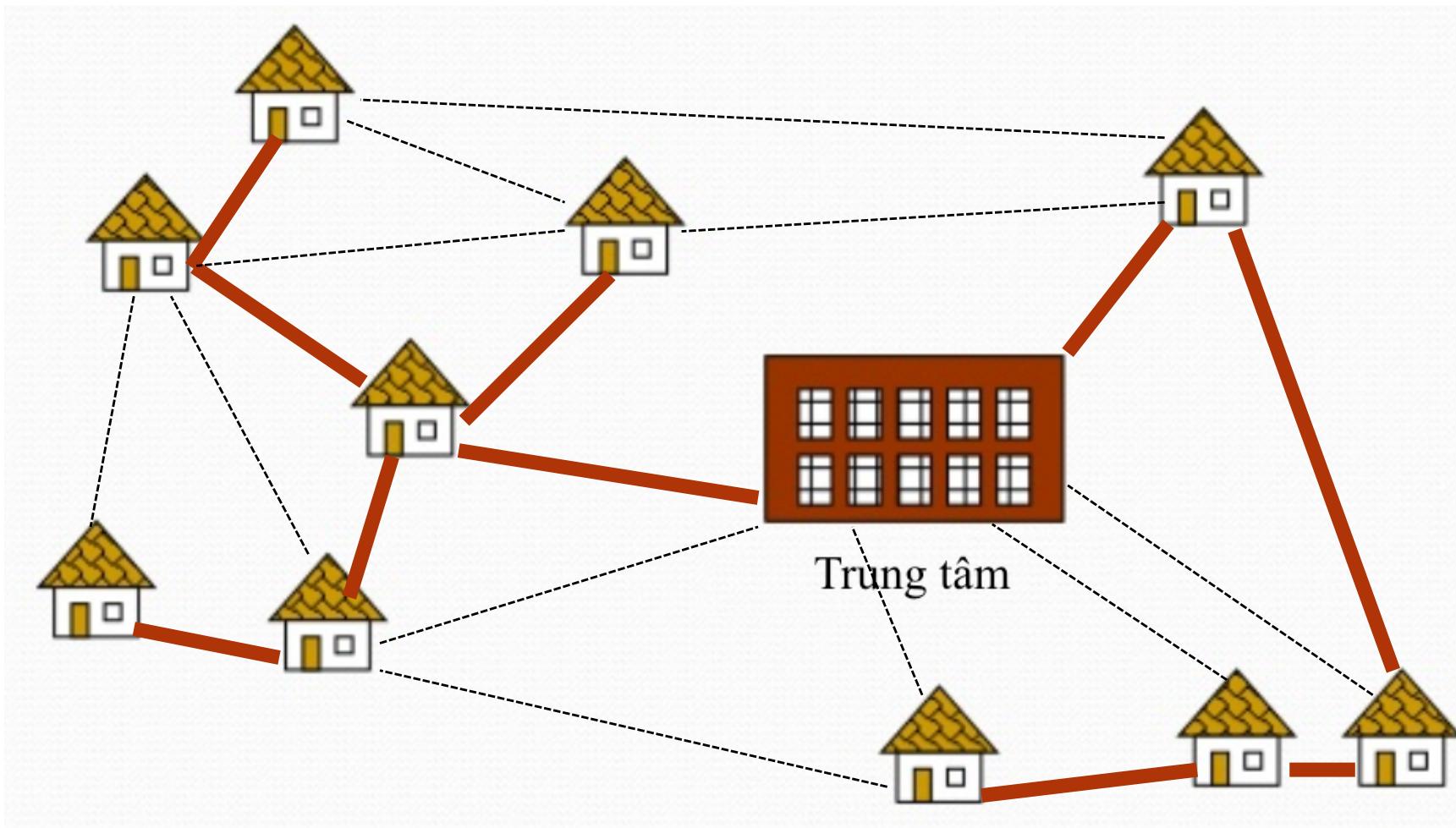
Bài toán xây dựng đường



Bài toán xây dựng đường

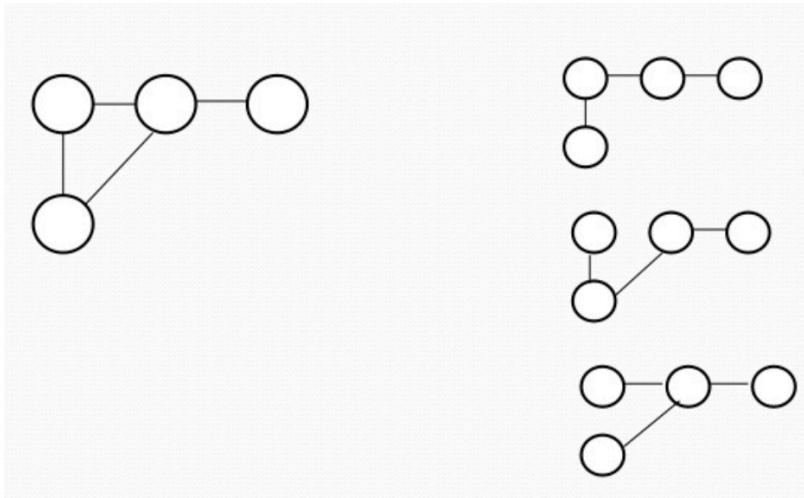


Bài toán xây dựng đường

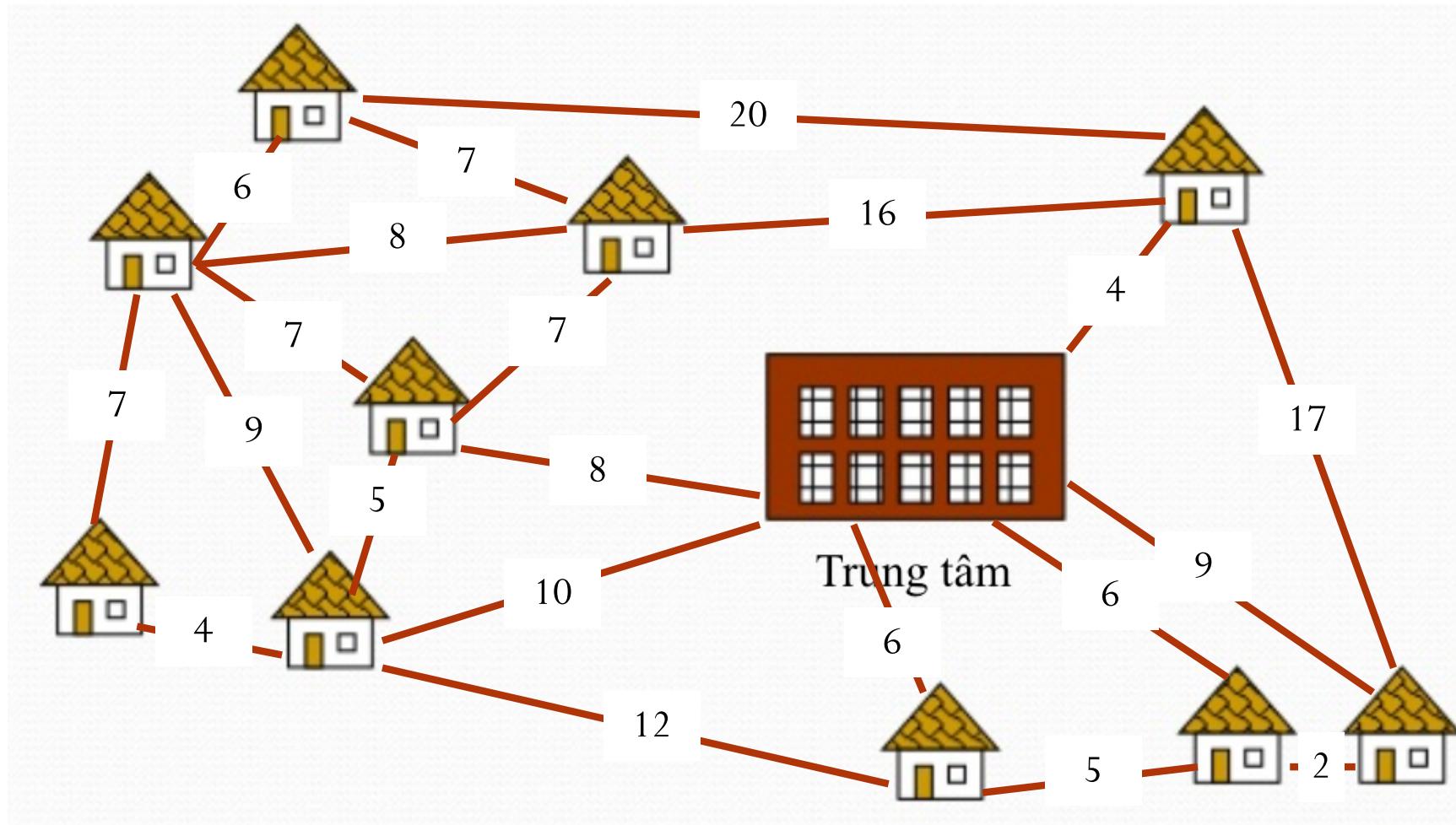


Cây khung

- Cây khung (spanning tree): cây bao trùm/cây phủ
 - $G = \langle V, E \rangle$ là đồ thị vô hướng liên thông
 - Đồ thị con $T = \langle V, E' \rangle$ là cây khung của G :
 - T là cây
 - Chứa tất cả các đỉnh của G
 - Có $|V| - 1$ cung
- Đồ thị (liên thông) có thể có nhiều cây khung



Bài toán xây dựng đường



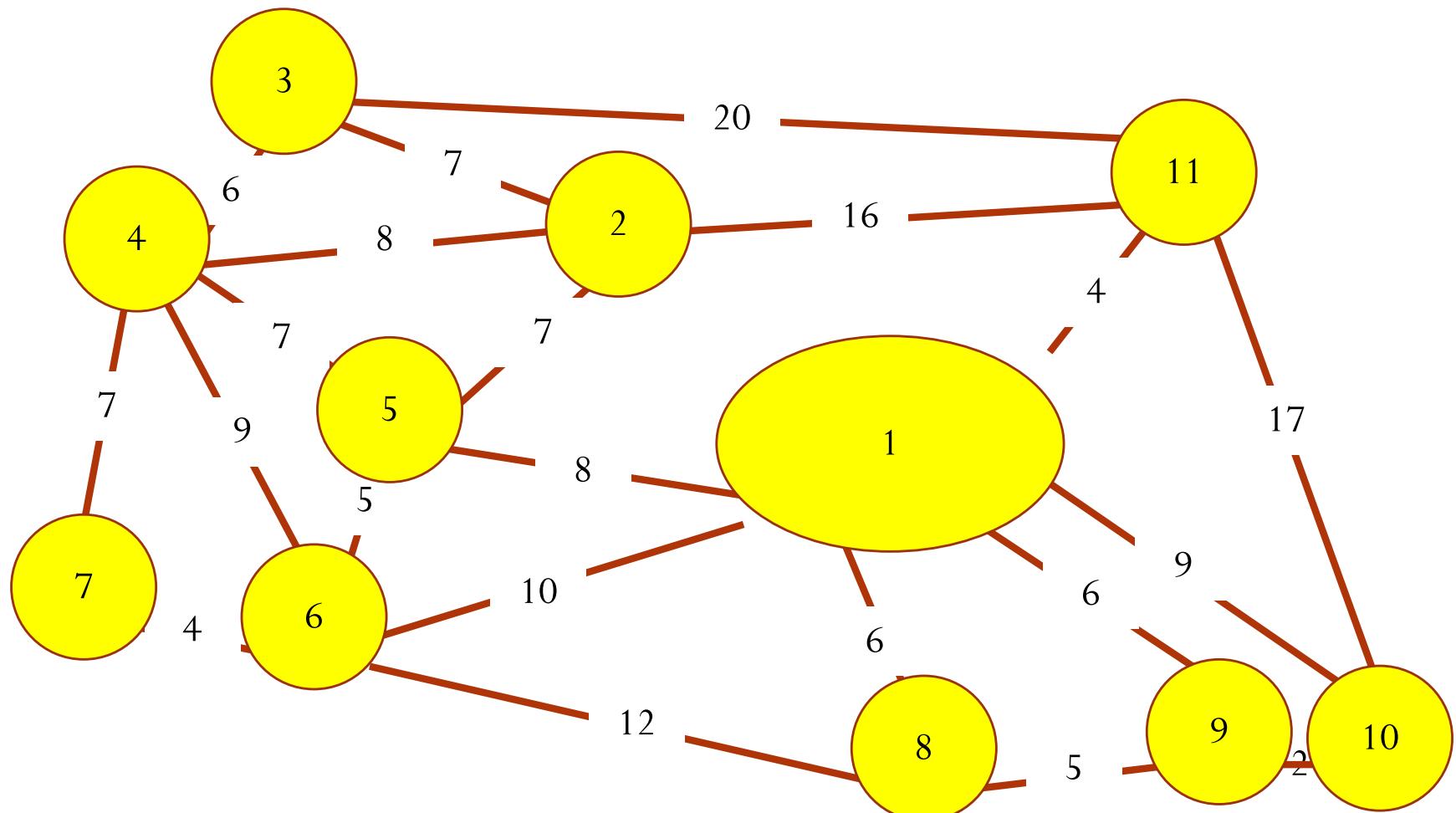
Bài toán xây dựng đường



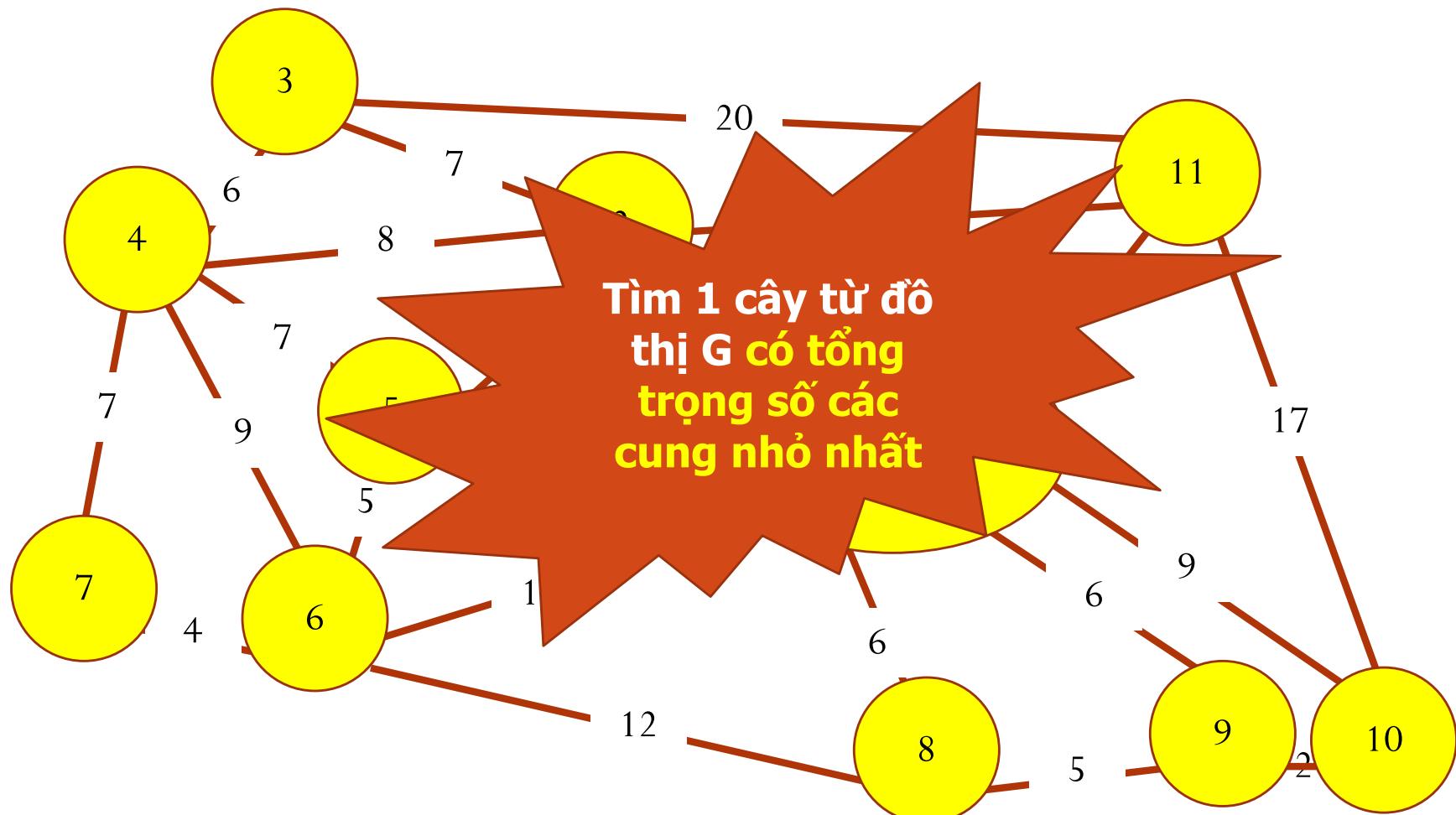
Bài toán xây dựng đường

- Mô hình về đồ thị
 - Đỉnh: nhà/trung tâm, gọi chung là địa điểm
 - Cung: đường (dự kiến) nối 2 địa điểm
 - Trọng số cung: chi phí xây dựng con đường nối 2 địa điểm tương ứng

Bài toán xây dựng đường



Bài toán xây dựng đường



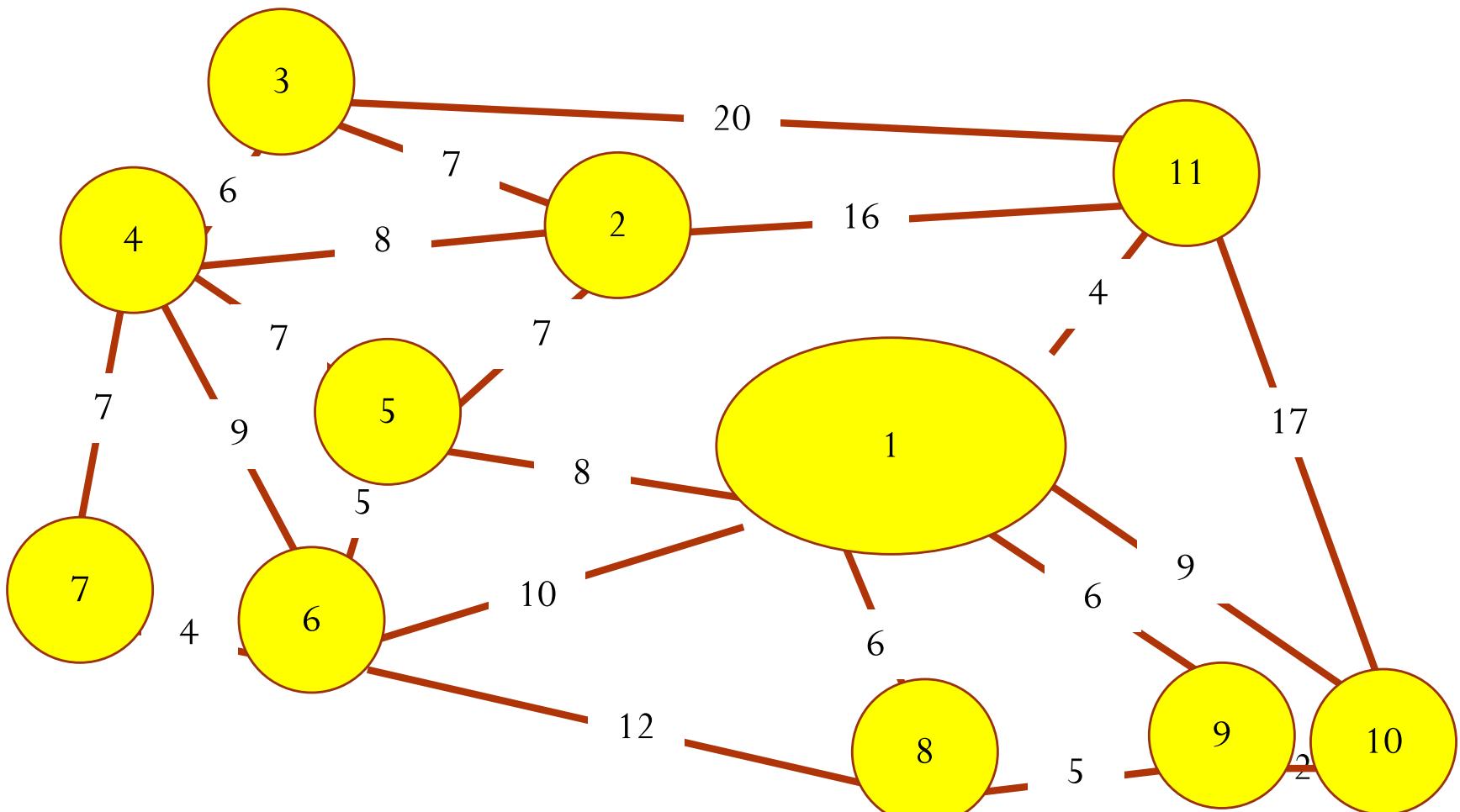
Cây khung tối thiểu

- Minimum spanning tree (MST)
 - Cây khung tối thiểu/cây khung nhỏ nhất
 - Cây khung có trọng lượng nhỏ nhất
 - ...
- $G = \langle V, E \rangle$ là đồ thị vô hướng liên thông. Cung (u, v) có trọng số $w(u, v)$
- Cây T là cây khung tối thiểu của G
 - T là cây khung của G
 - T có tổng trọng số nhỏ nhất trong tất cả các cây khung của G

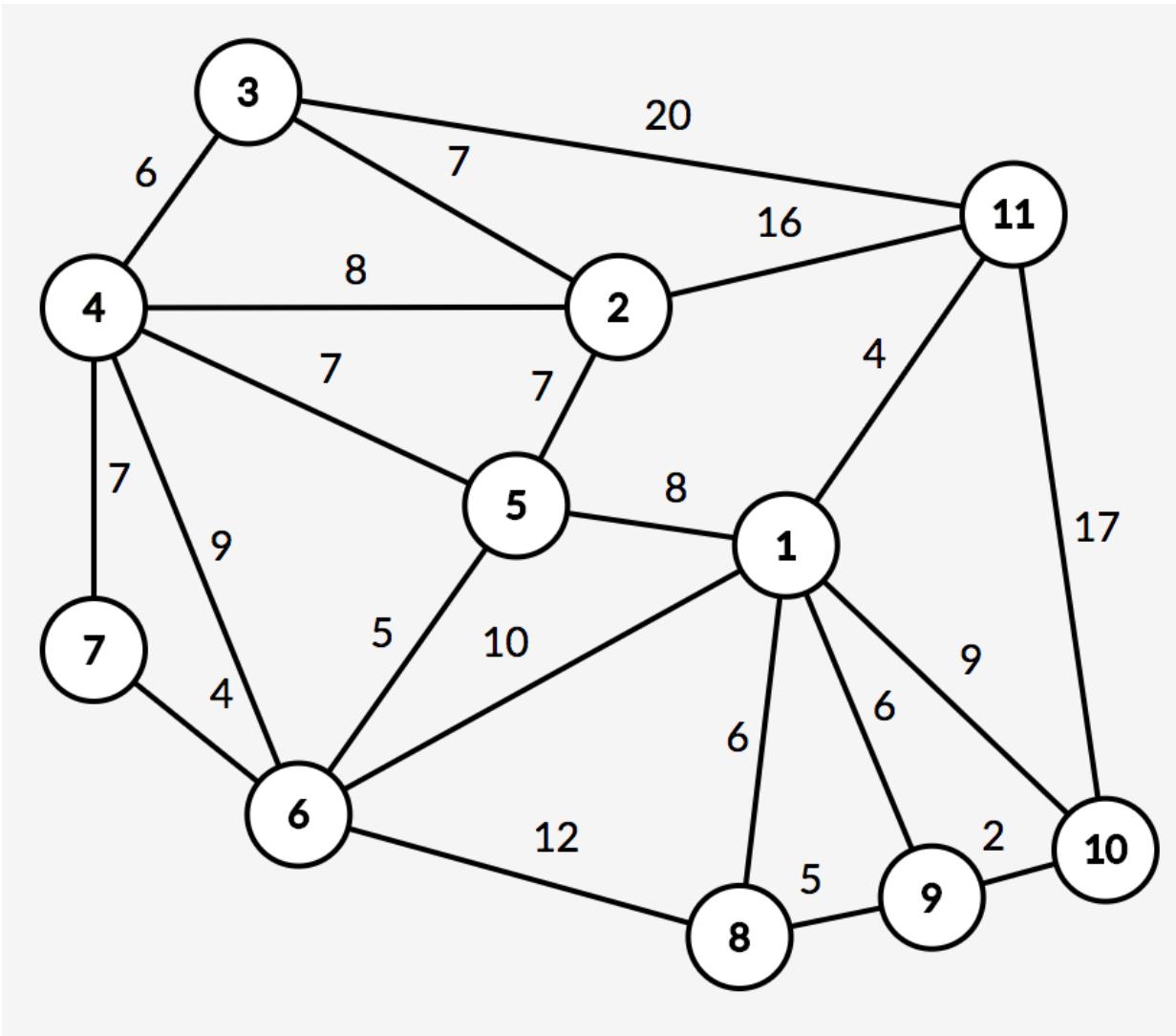
Giải thuật Kruskal

- Sắp xếp các cung của G theo thứ tự trọng số tăng dần
- Khởi tạo cây T rỗng (không chứa cung nào cả)
- Lần lượt xét từng cung của G (theo thứ tự đã sắp xếp)
 - Nếu thêm cung $e = (u, v)$ vào T mà không tạo thành chu trình thì thêm e vào T
 - Ngược lại, bỏ qua cung e
- Điều kiện dừng:
 - T có $n - 1$ cung hoặc tất cả các cung của G đều đã được xét

Ví dụ



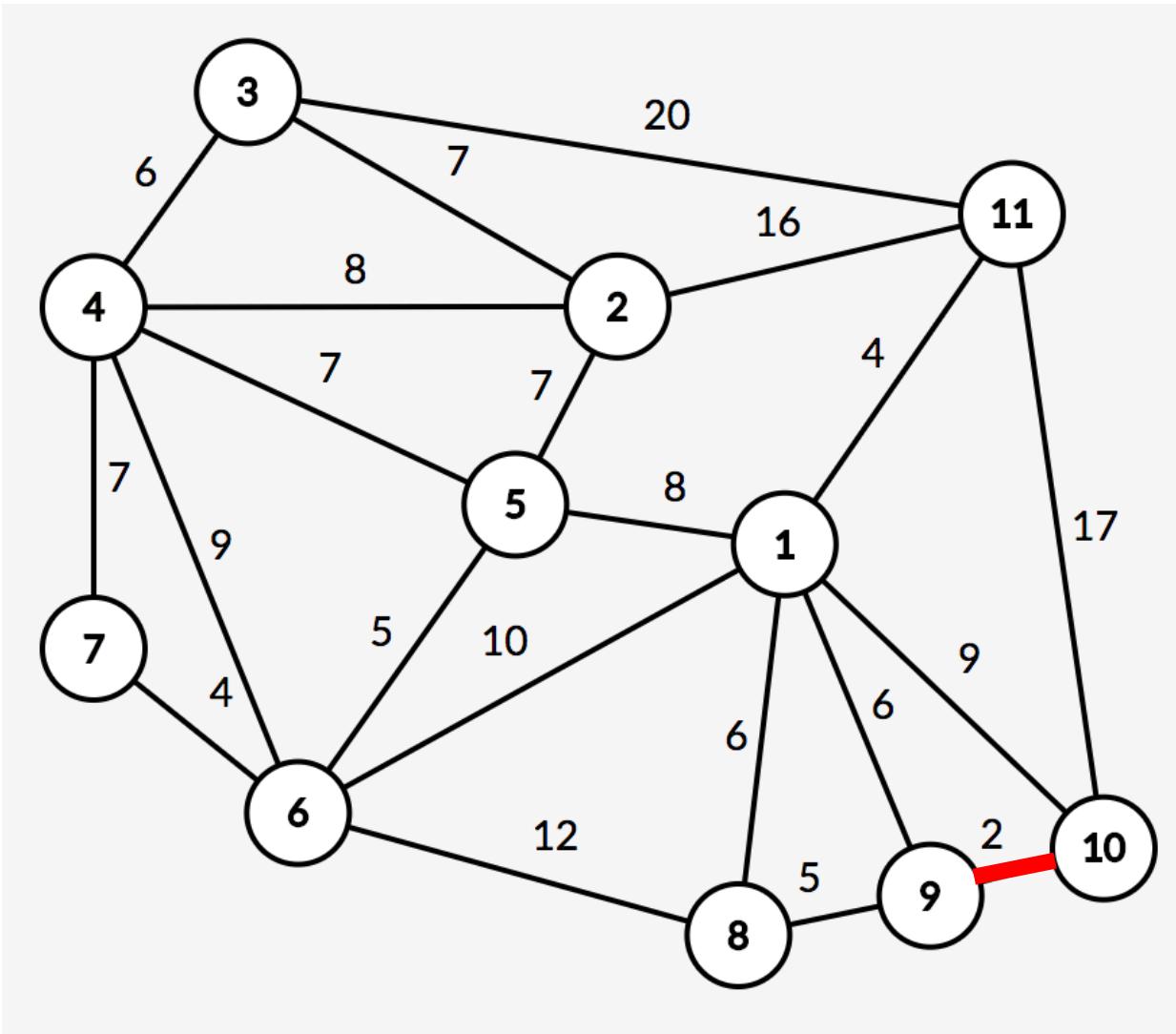
Ví dụ



u	v	w
1	5	8
1	6	10
1	8	6
1	9	6
1	10	9
1	11	4
2	3	7
2	4	8
2	5	7
2	11	16
3	4	6
3	11	20
4	5	7
4	6	9
4	7	7
5	6	5
6	7	4
6	8	12
8	9	5
9	10	2
10	11	17

Ví dụ

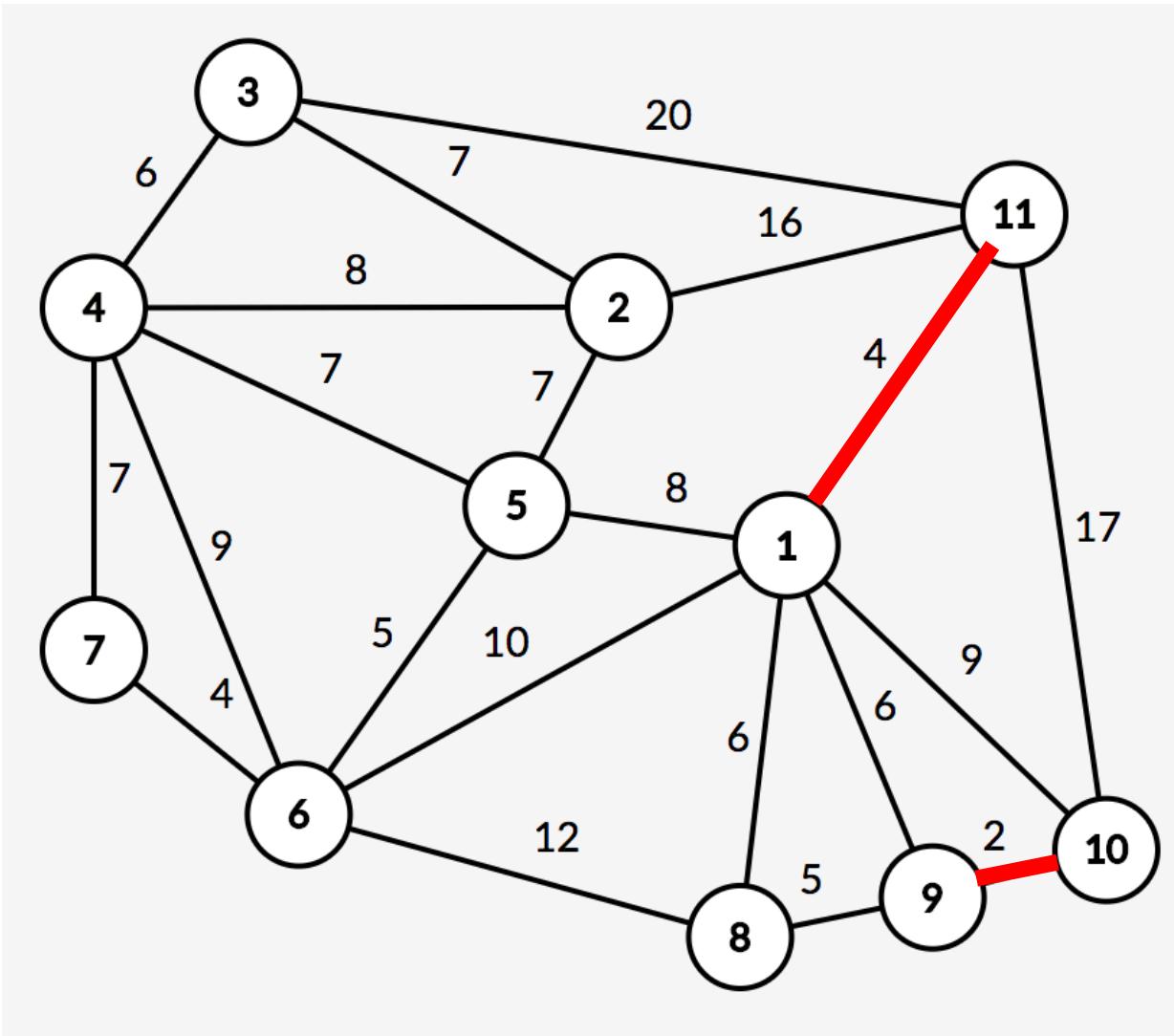
Xét (9, 10)
thêm (9, 10)



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

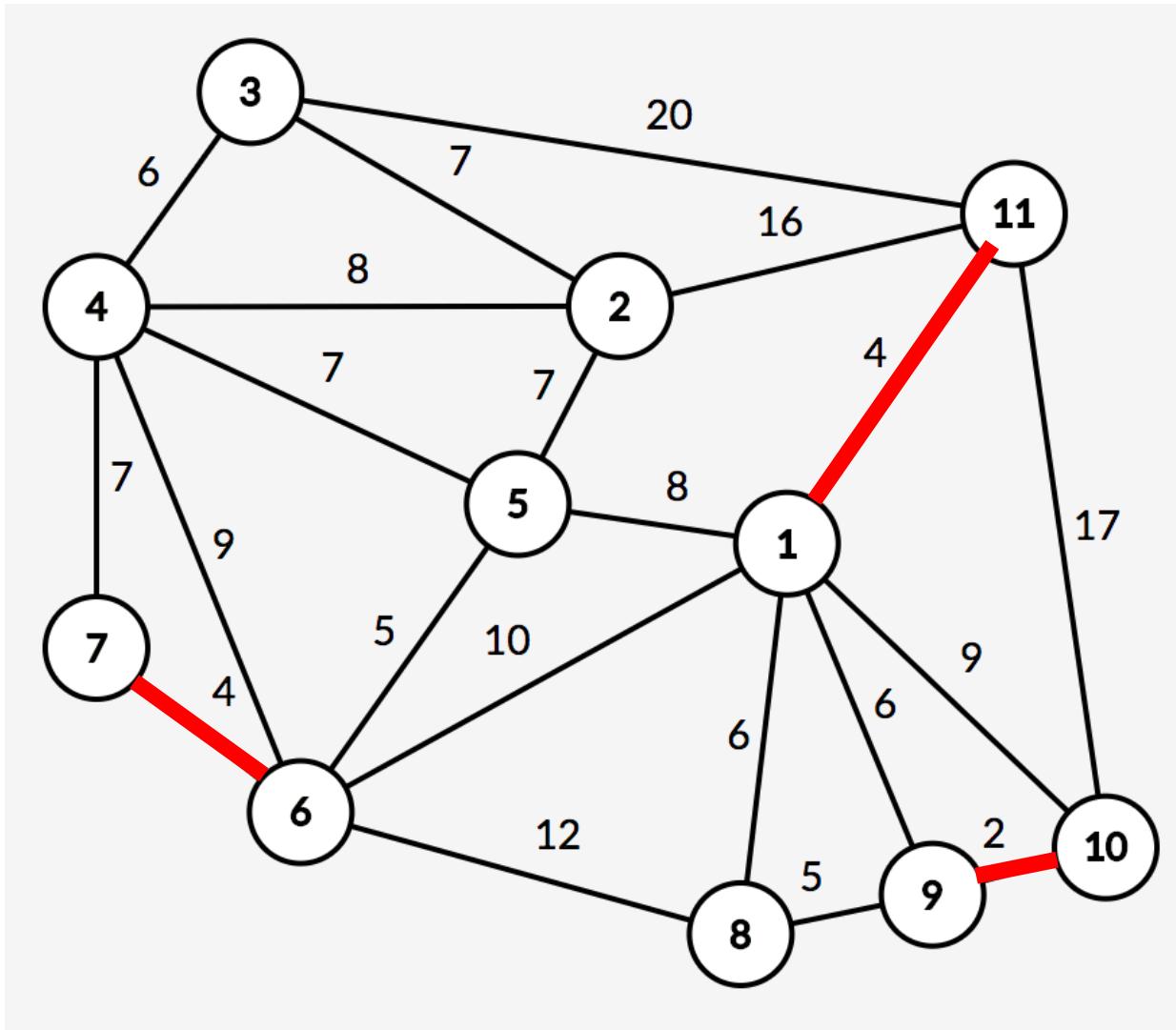
Xét (1, 11)
thêm (1, 11)



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

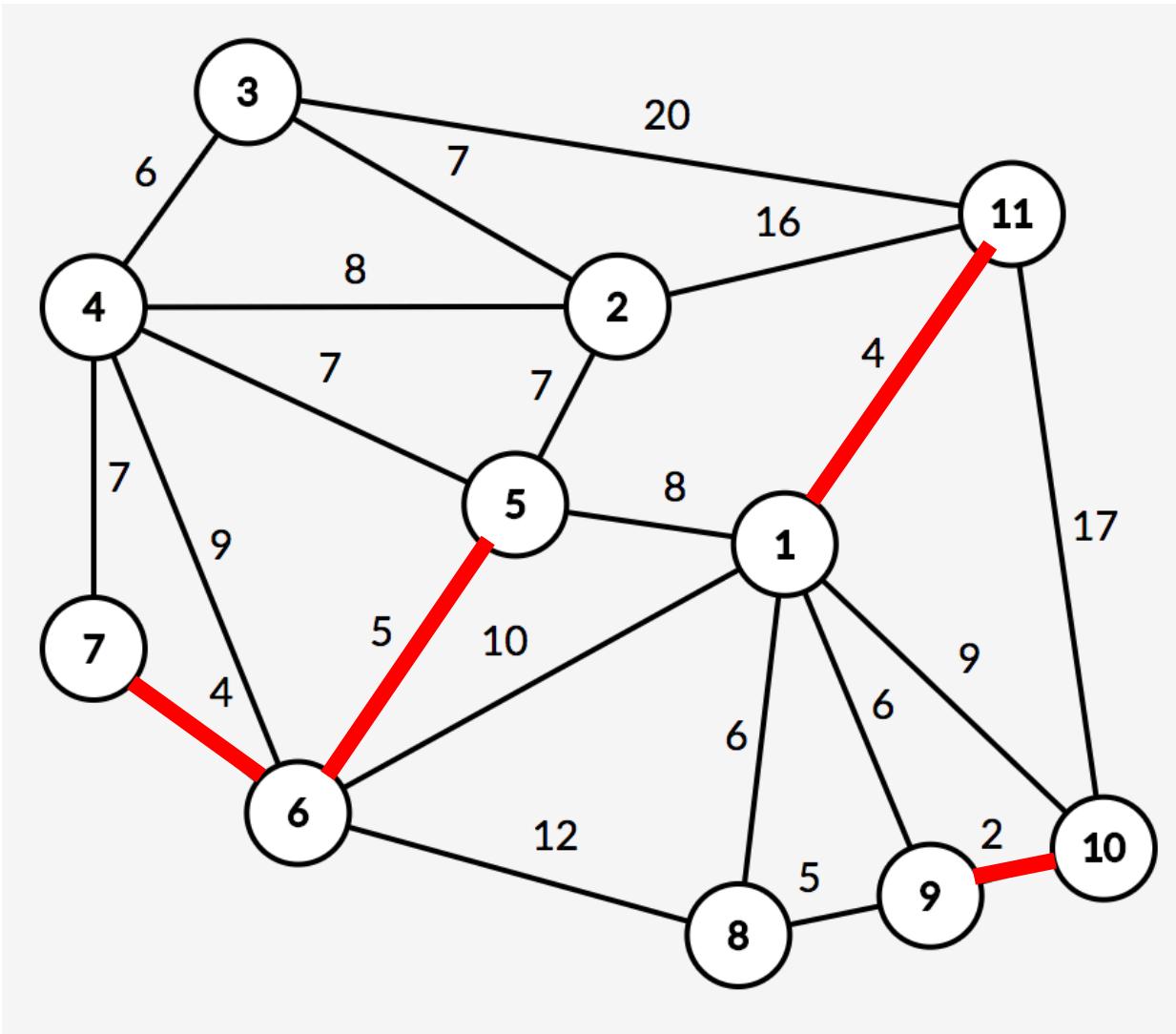
Xét $(6, 7)$
thêm $(6, 7)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

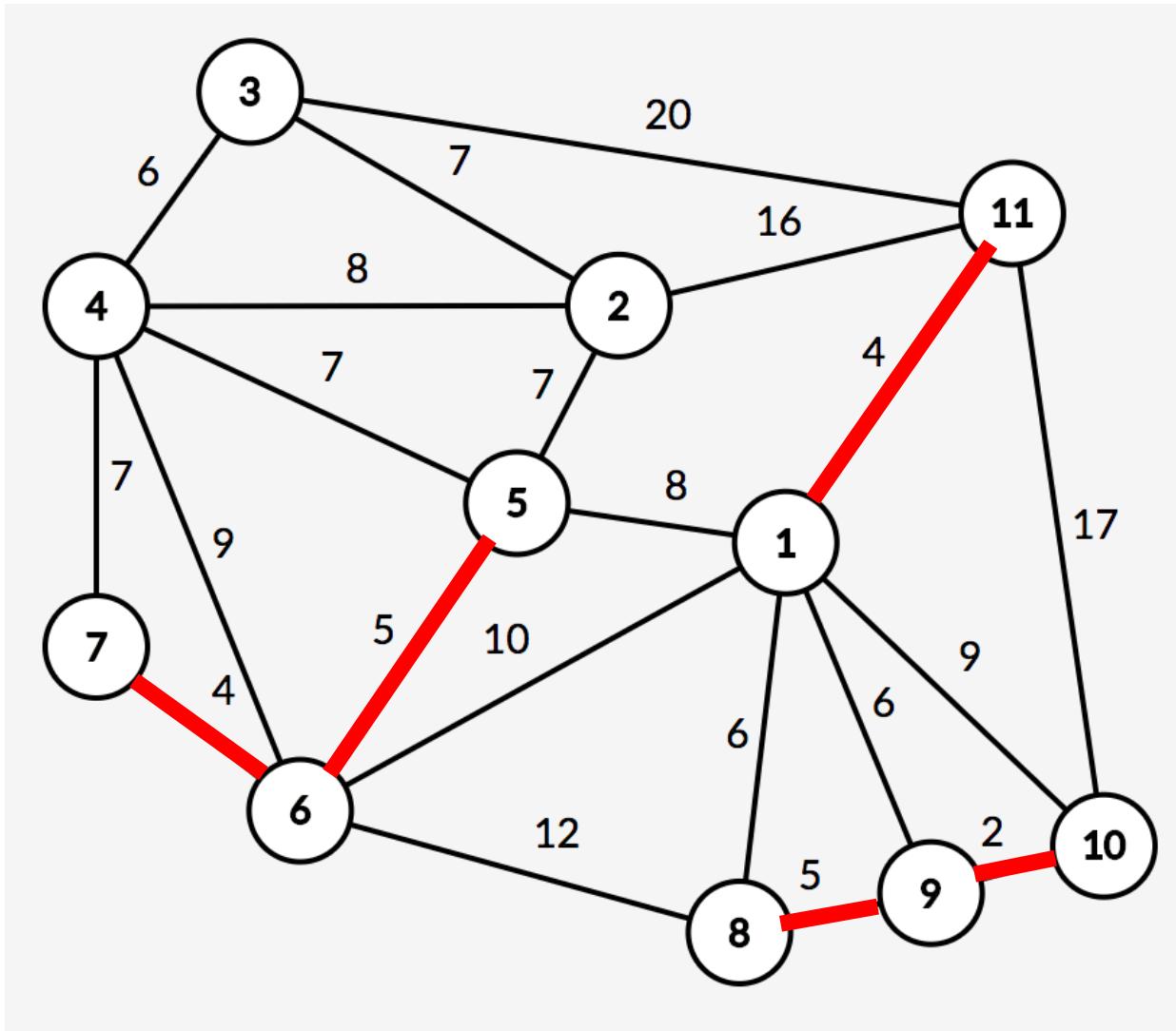
Xét $(5, 6)$
thêm $(5, 6)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

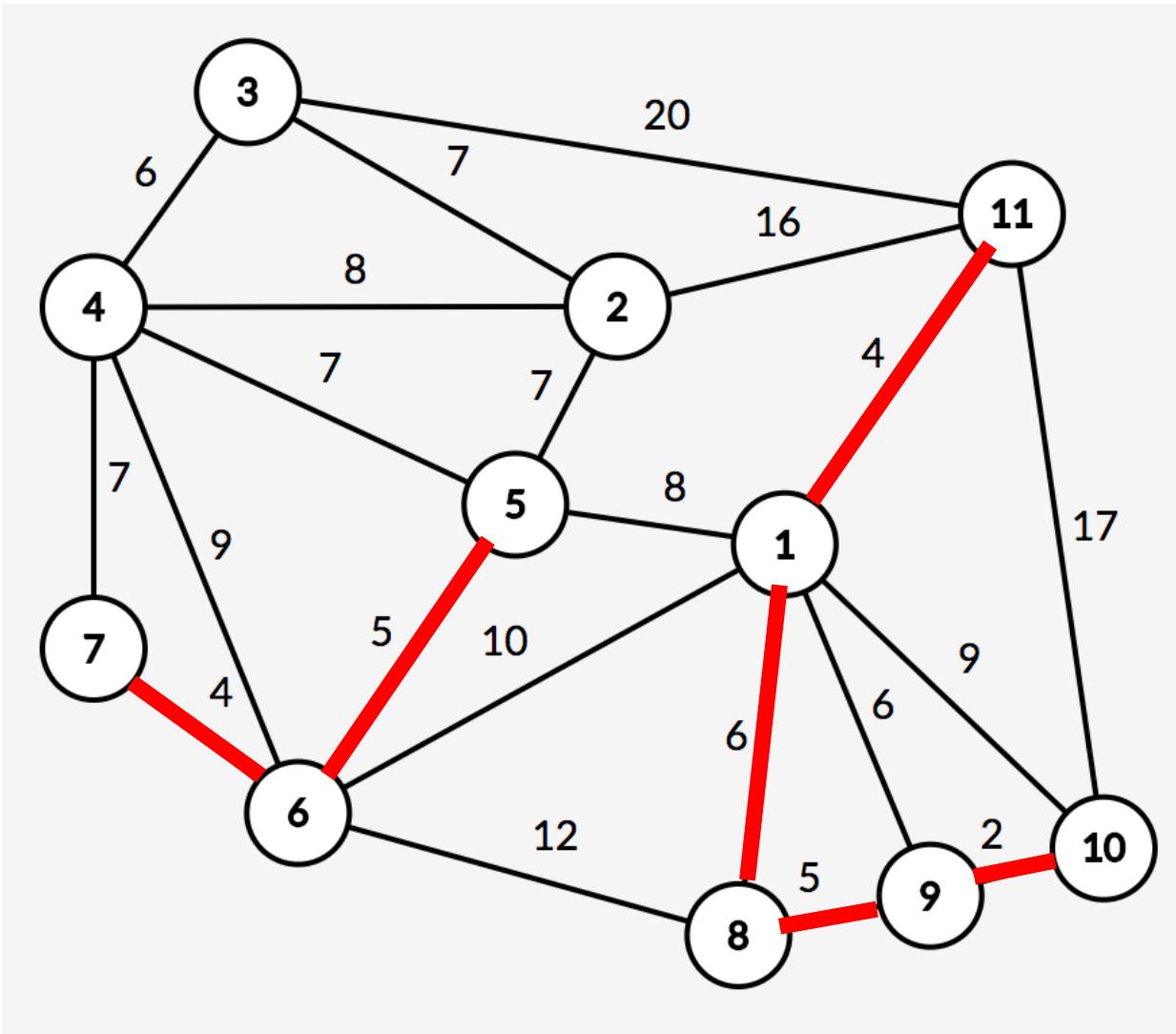
Xét $(8, 9)$
thêm $(8, 9)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

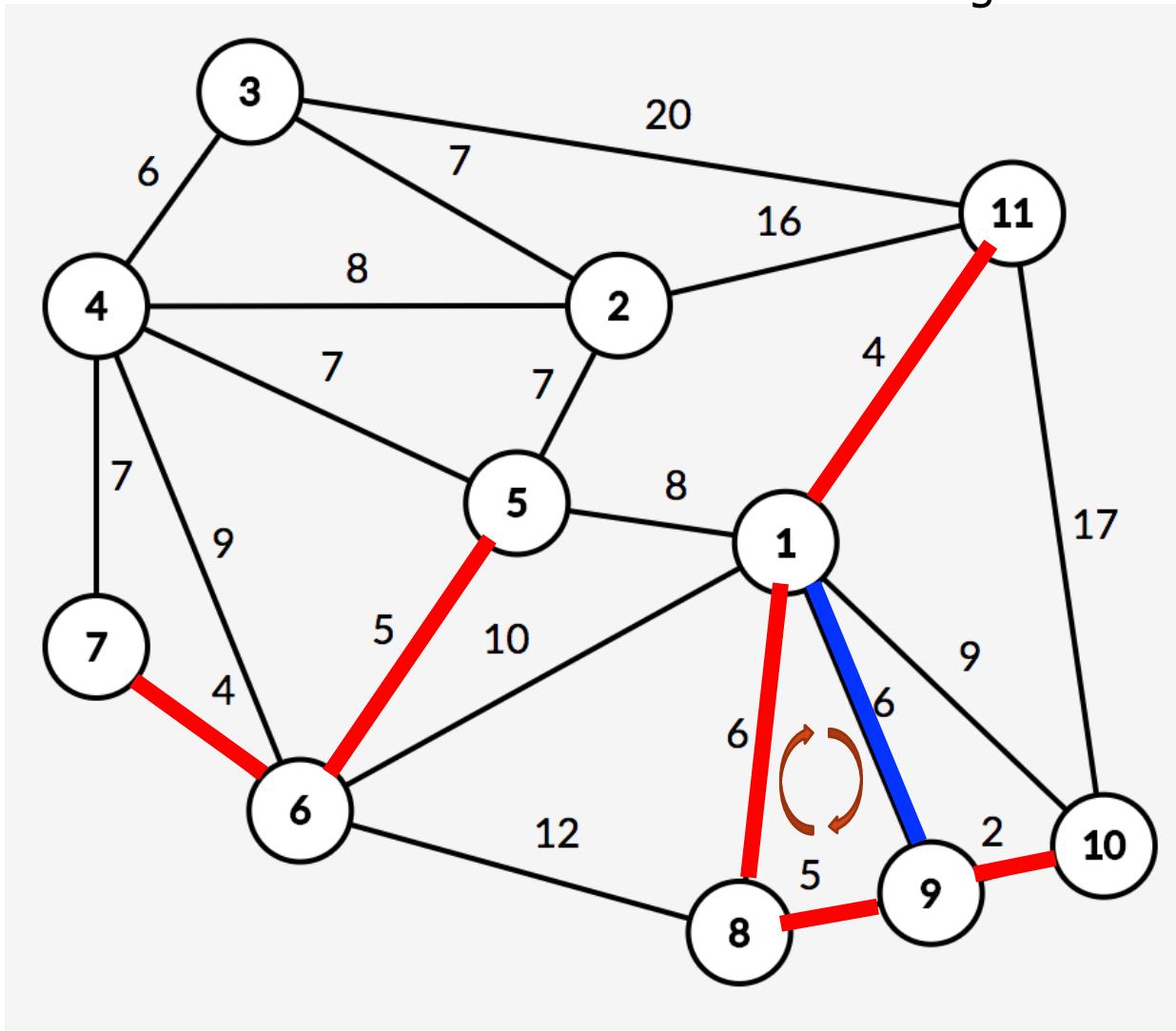
Xét (1, 8)
thêm (1, 8)



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

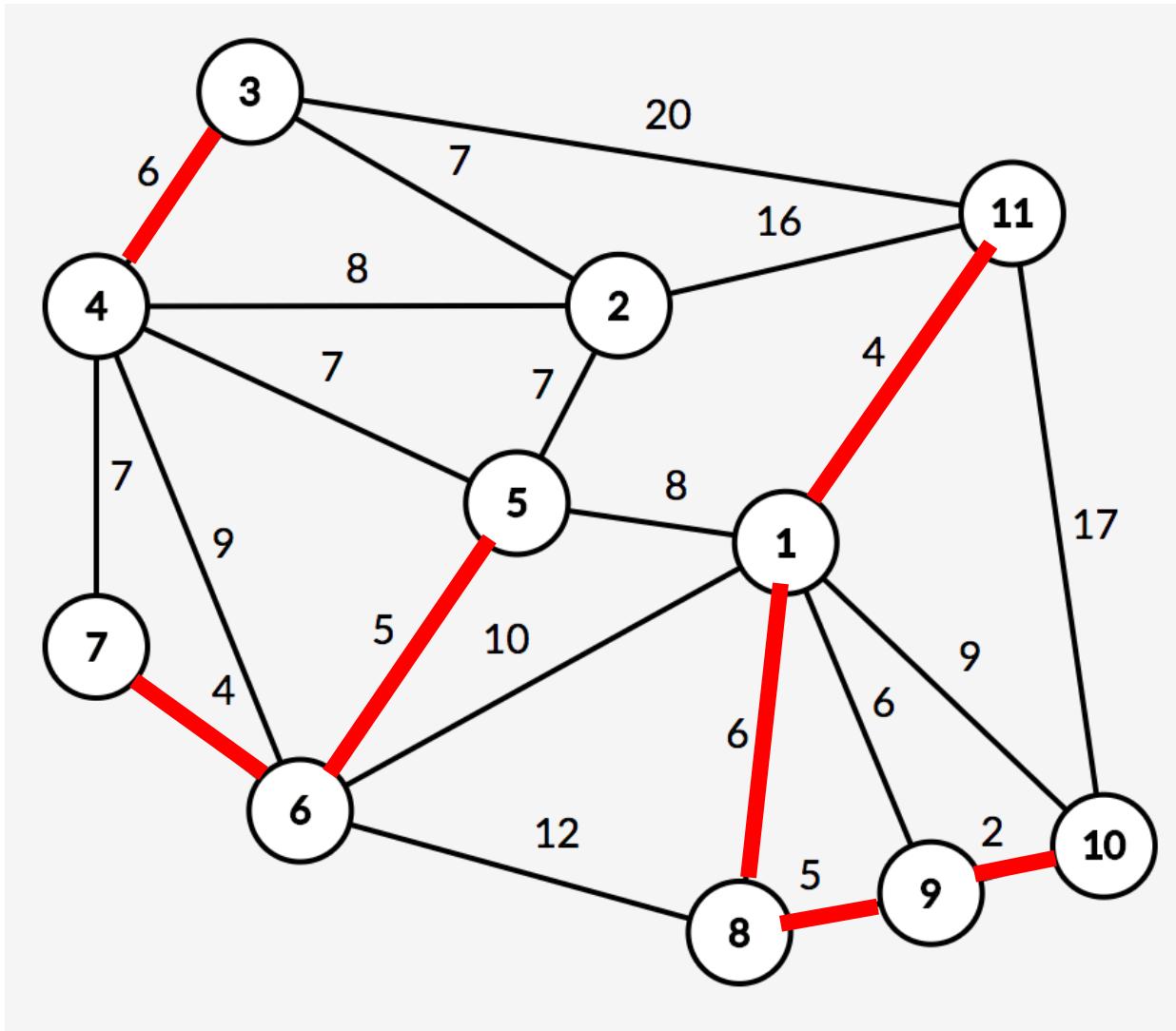
Xét (1, 9)
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

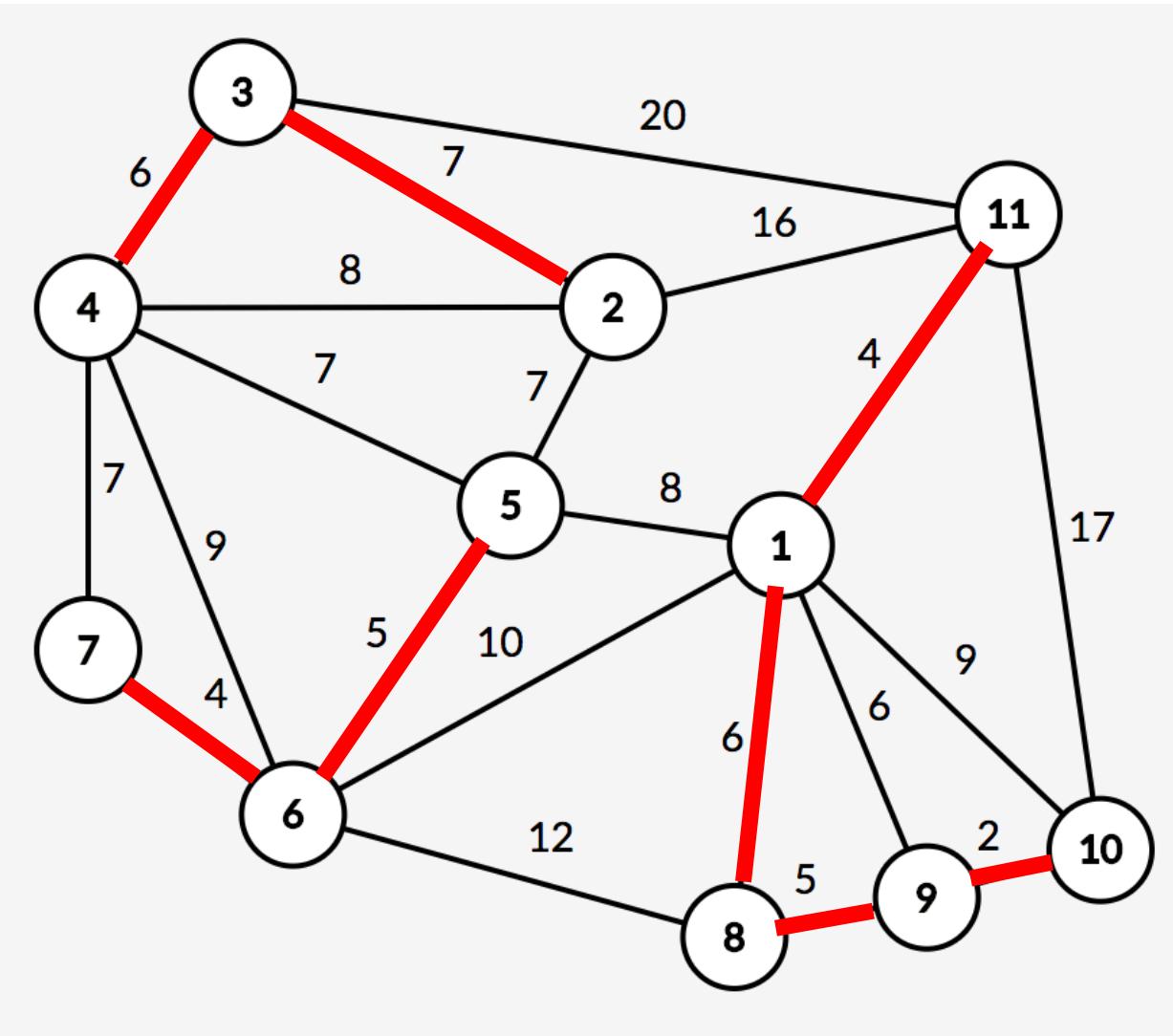
Xét $(3, 4)$
thêm $(3, 4)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

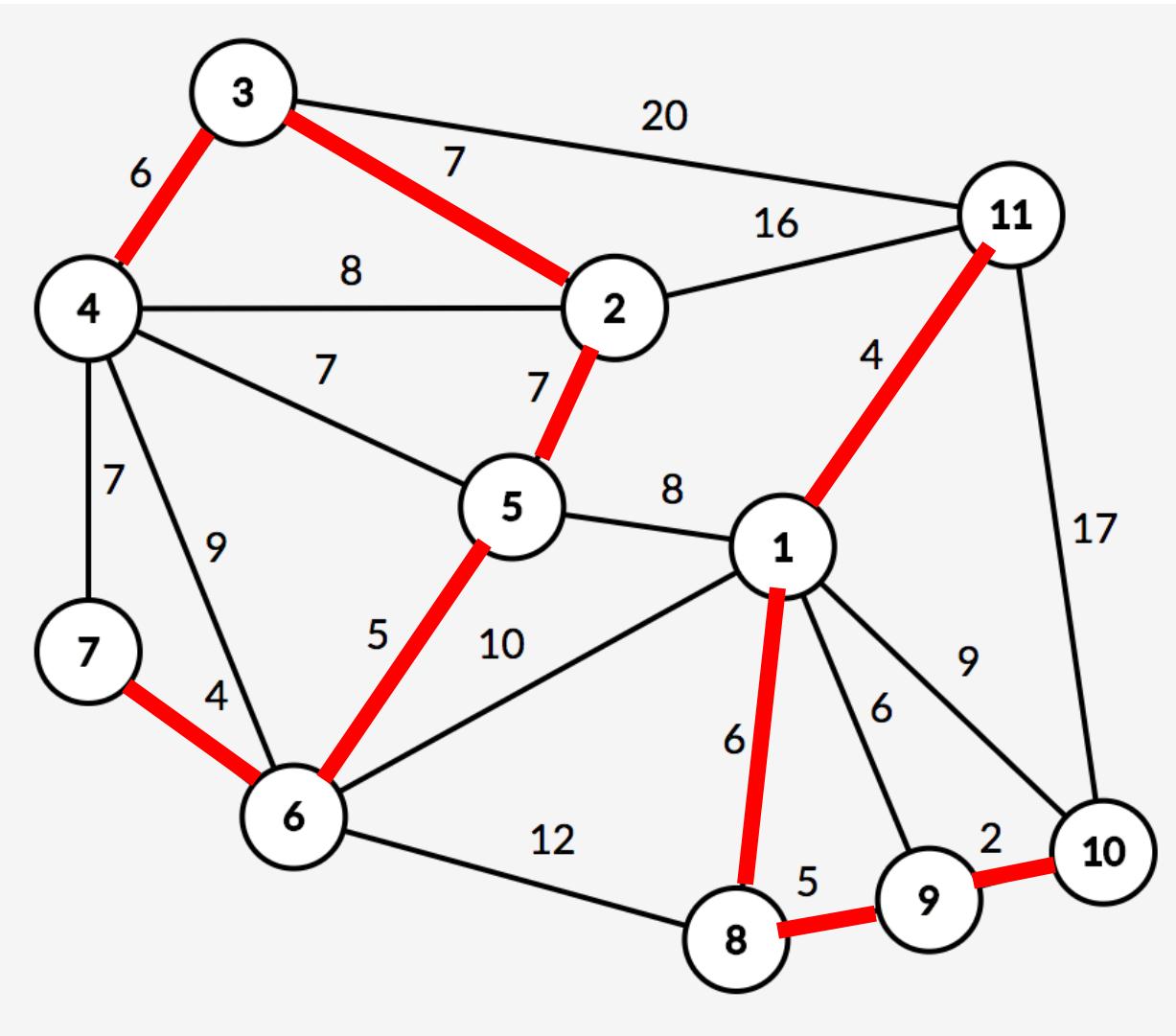
Xét $(2, 3)$
thêm $(2, 3)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

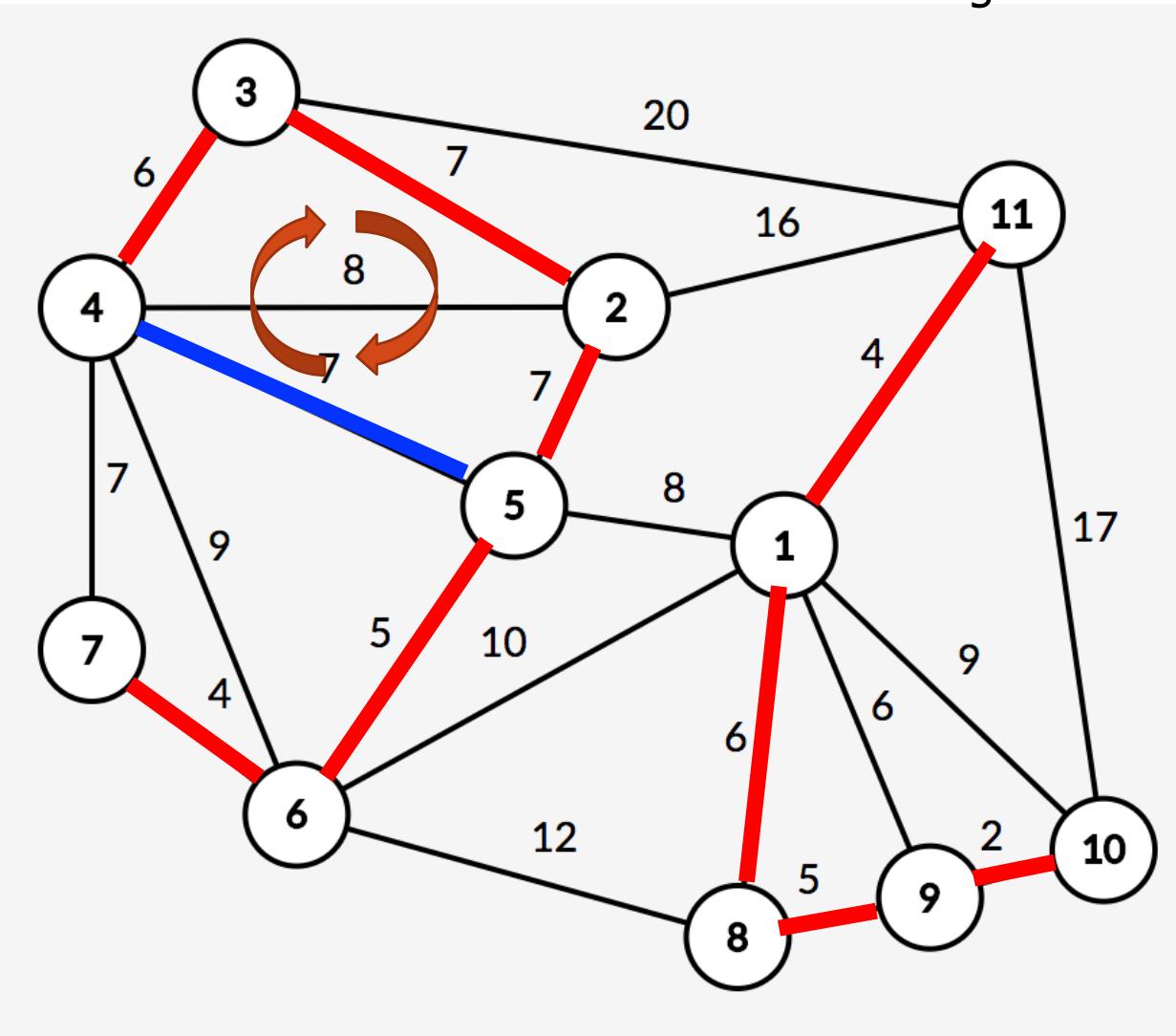
Xét $(2, 5)$
thêm $(2, 5)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

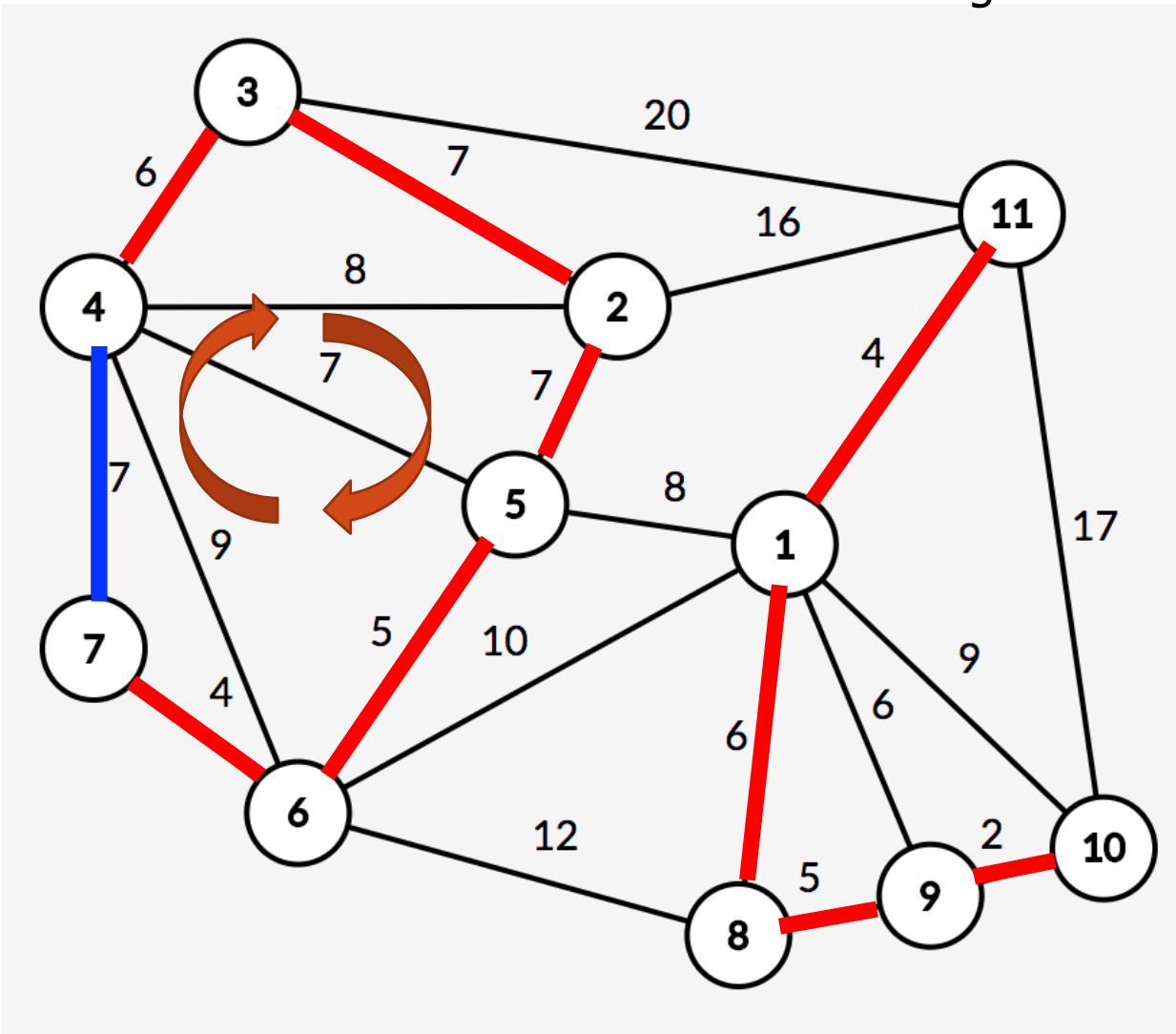
Xét (4, 5)
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

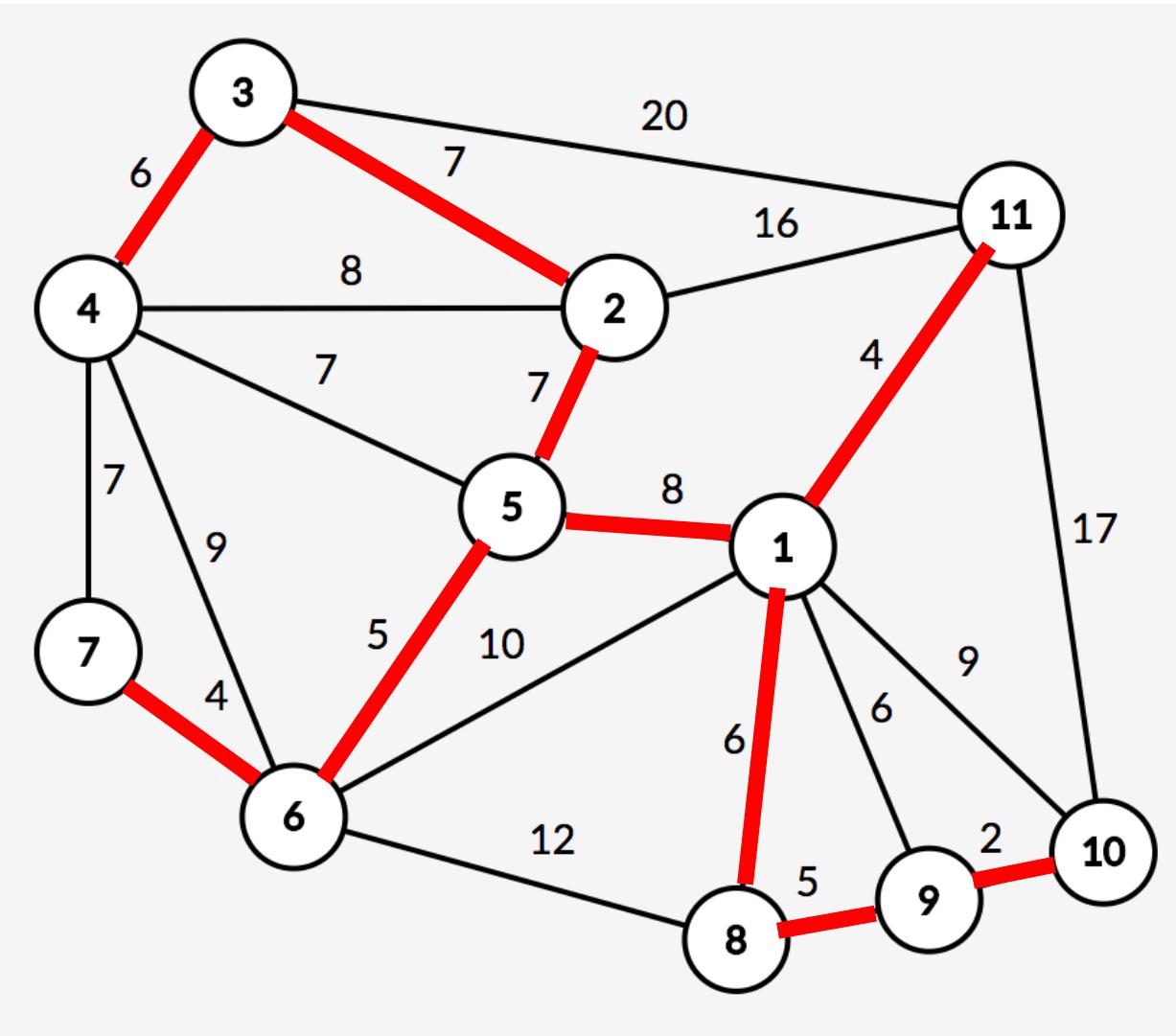
Xét (4, 7)
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

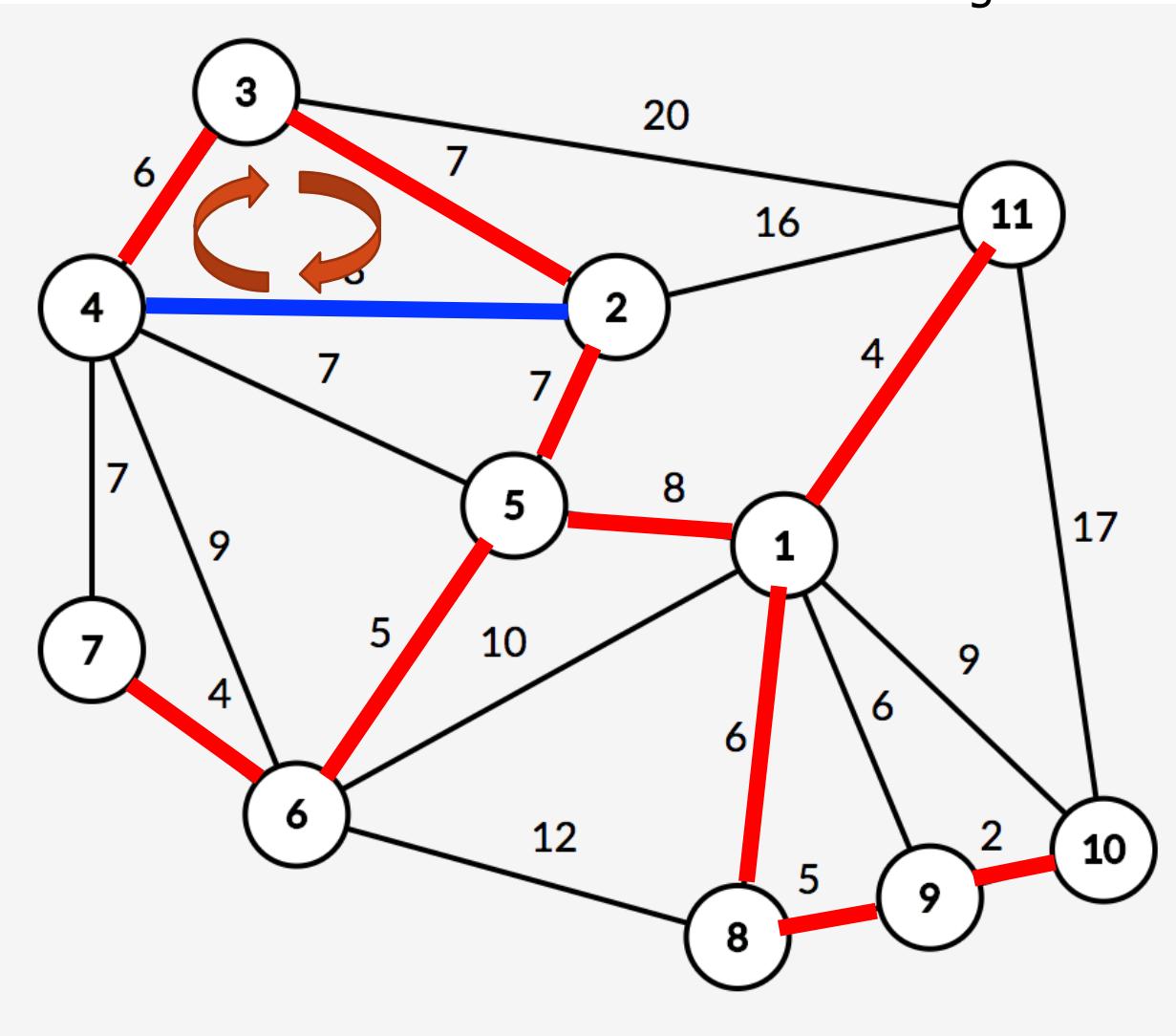
Xét $(1, 5)$
thêm $(1, 5)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

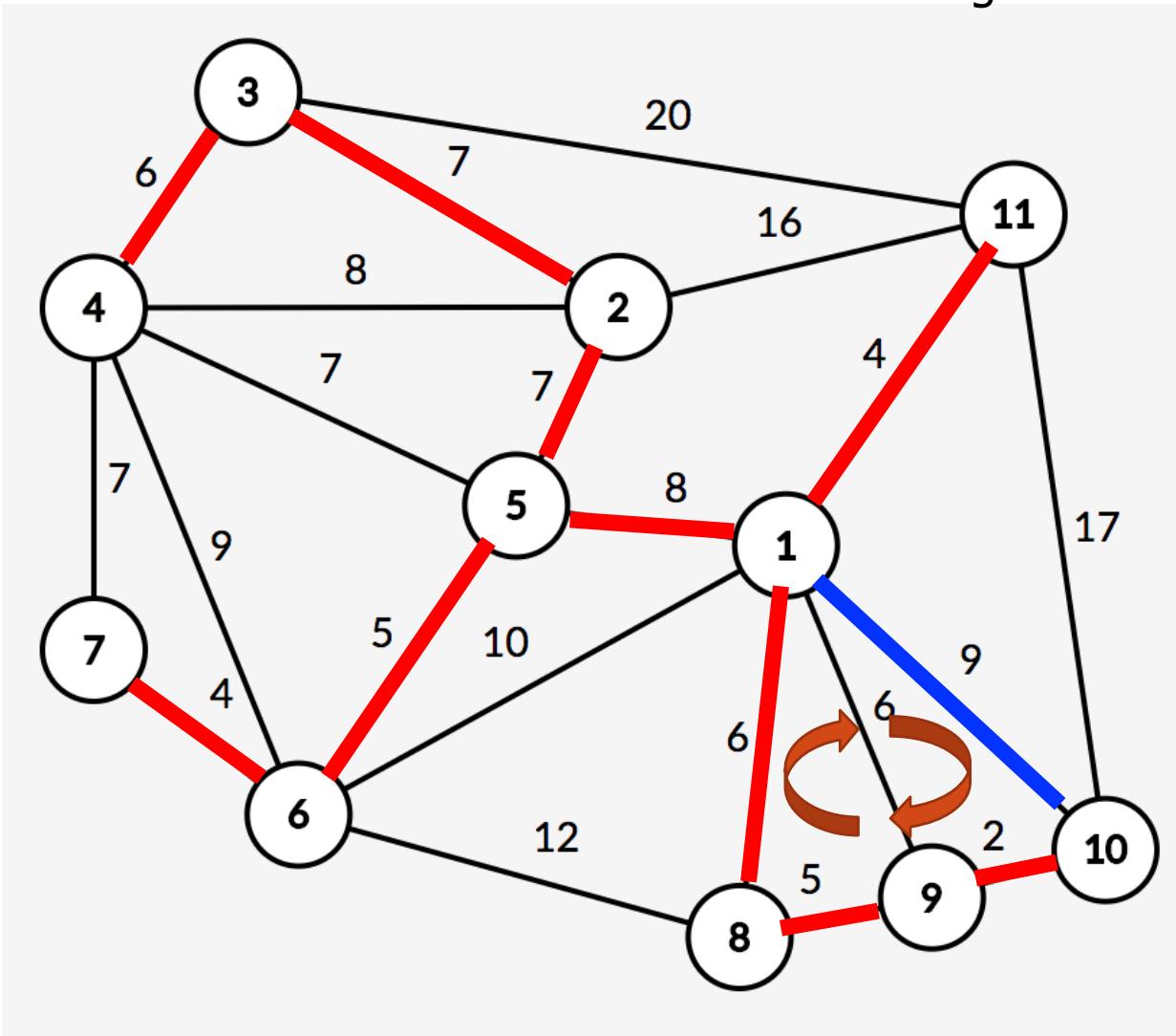
Xét $(2, 4)$
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

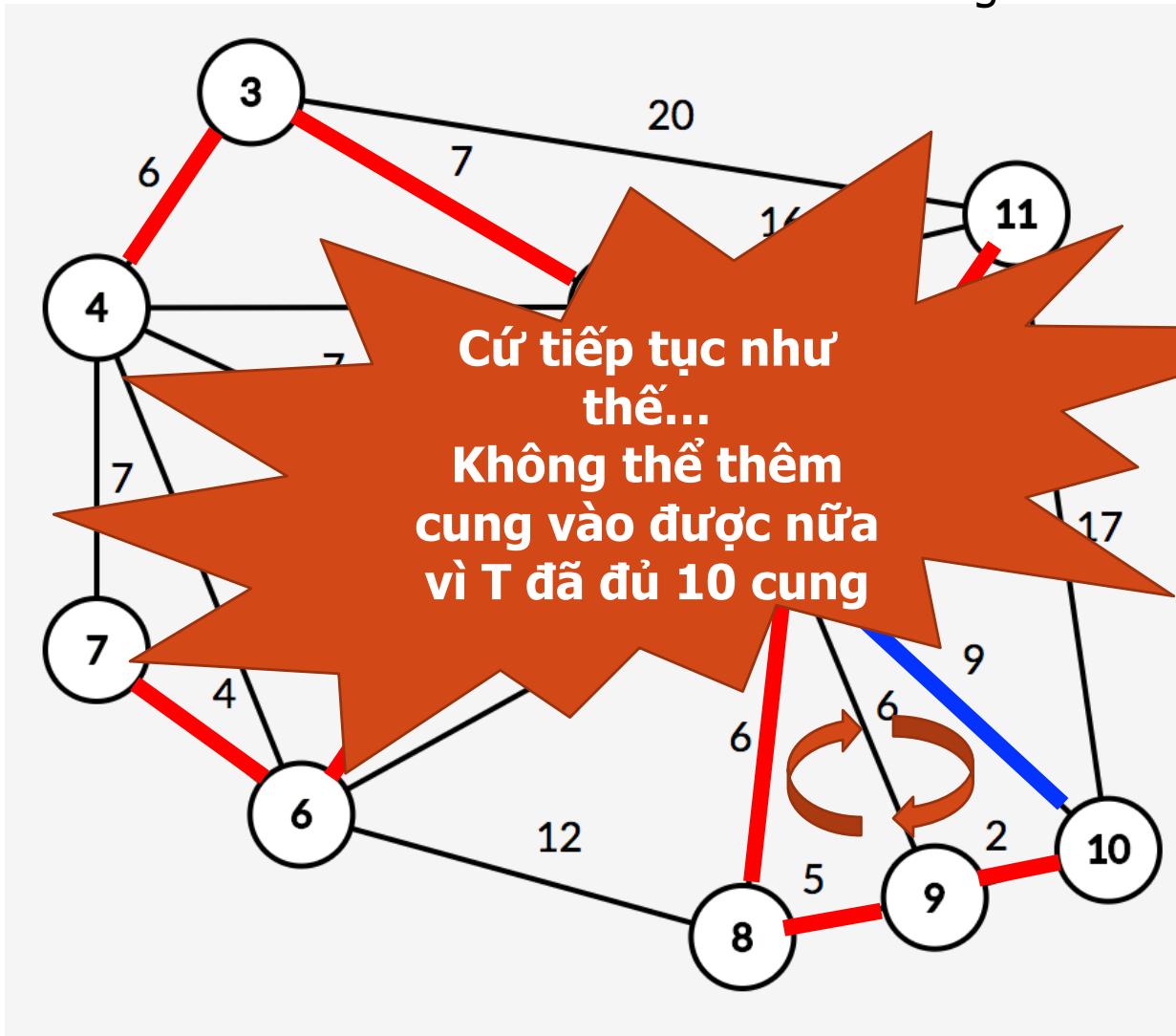
Xét (1, 10)
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Ví dụ

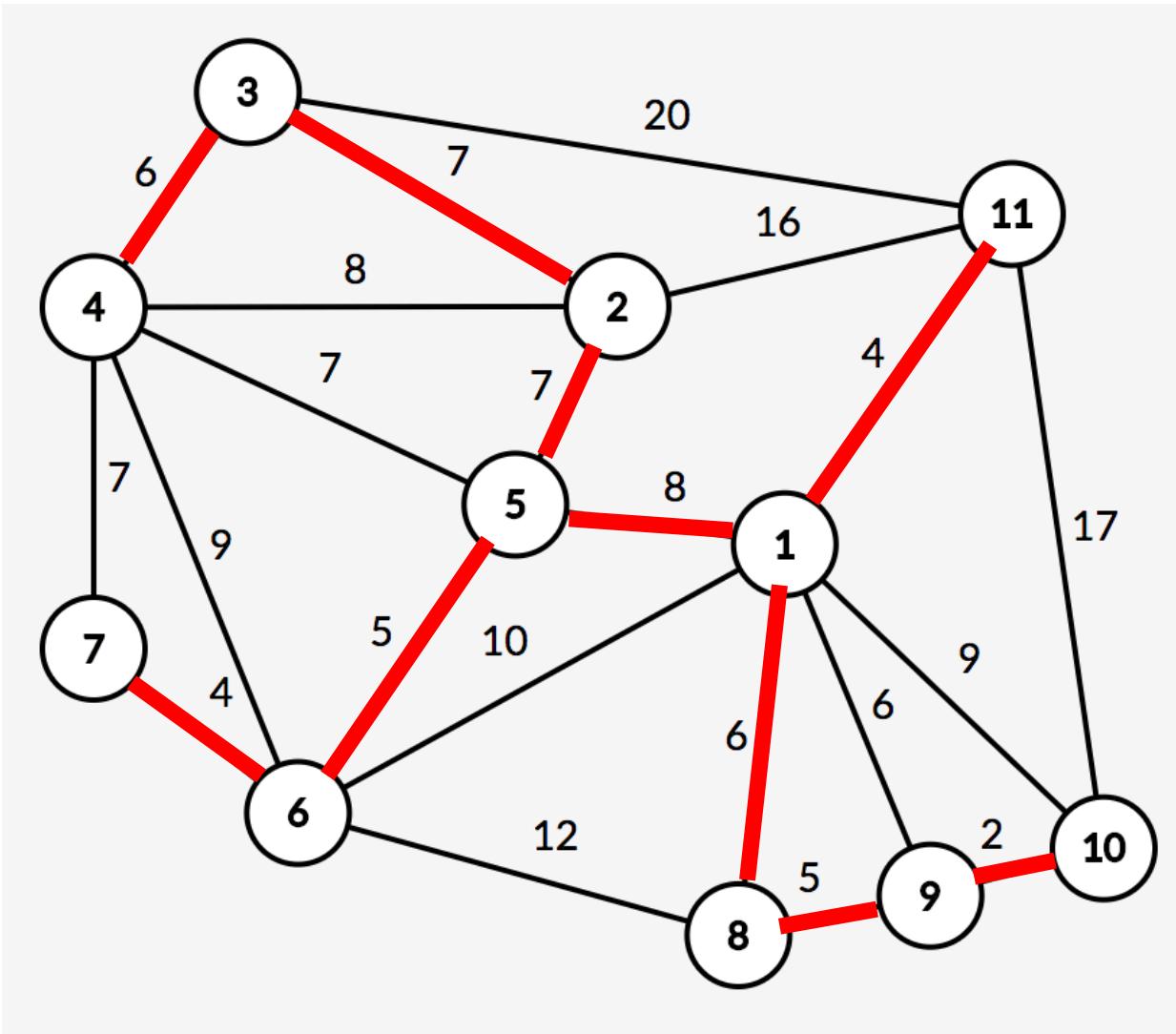
Xét (1, 10)
tạo chu trình
không thêm



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

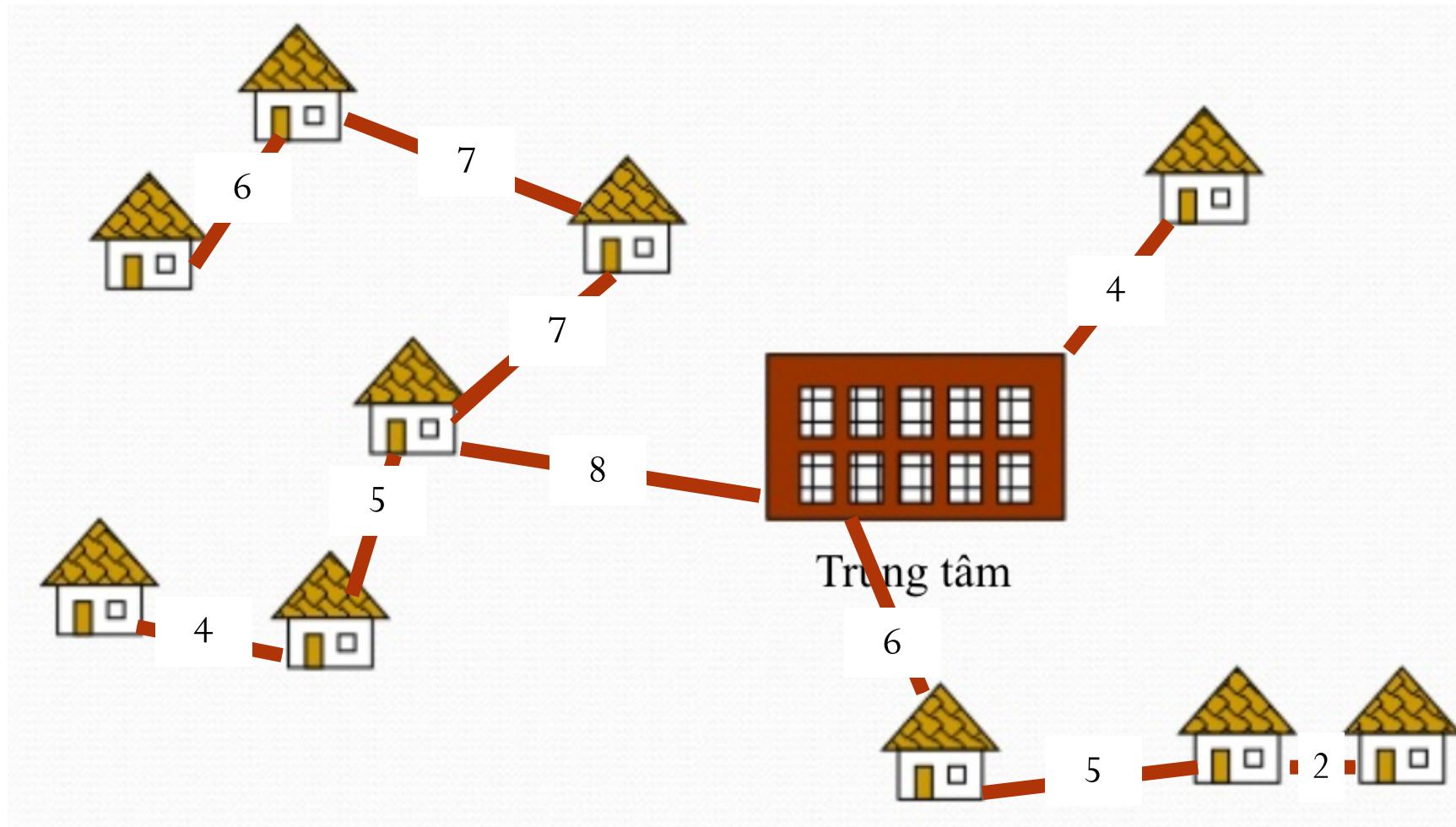
Ví dụ

Xét $(1, 5)$
thêm $(1, 5)$



u	v	w
9	10	2
1	11	4
6	7	4
5	6	5
8	9	5
1	8	6
1	9	6
3	4	6
2	3	7
2	5	7
4	5	7
4	7	7
1	5	8
2	4	8
1	10	9
4	6	9
1	6	10
6	8	12
2	11	16
10	11	17
3	11	20

Bài toán xây dựng đường



Ứng dụng

- Xét 1 ứng dụng có n máy trạm được kết nối với nhau thông qua hệ thống mạng truyền thông
- Mỗi kết nối giữa 2 máy trạm có 1 chi phí nào đó
- Bài toán: thiết lập một mạng truyền thông sao cho tất cả các máy trạm đều có thể kết nối được với nhau và tổng chi phí thấp nhất.

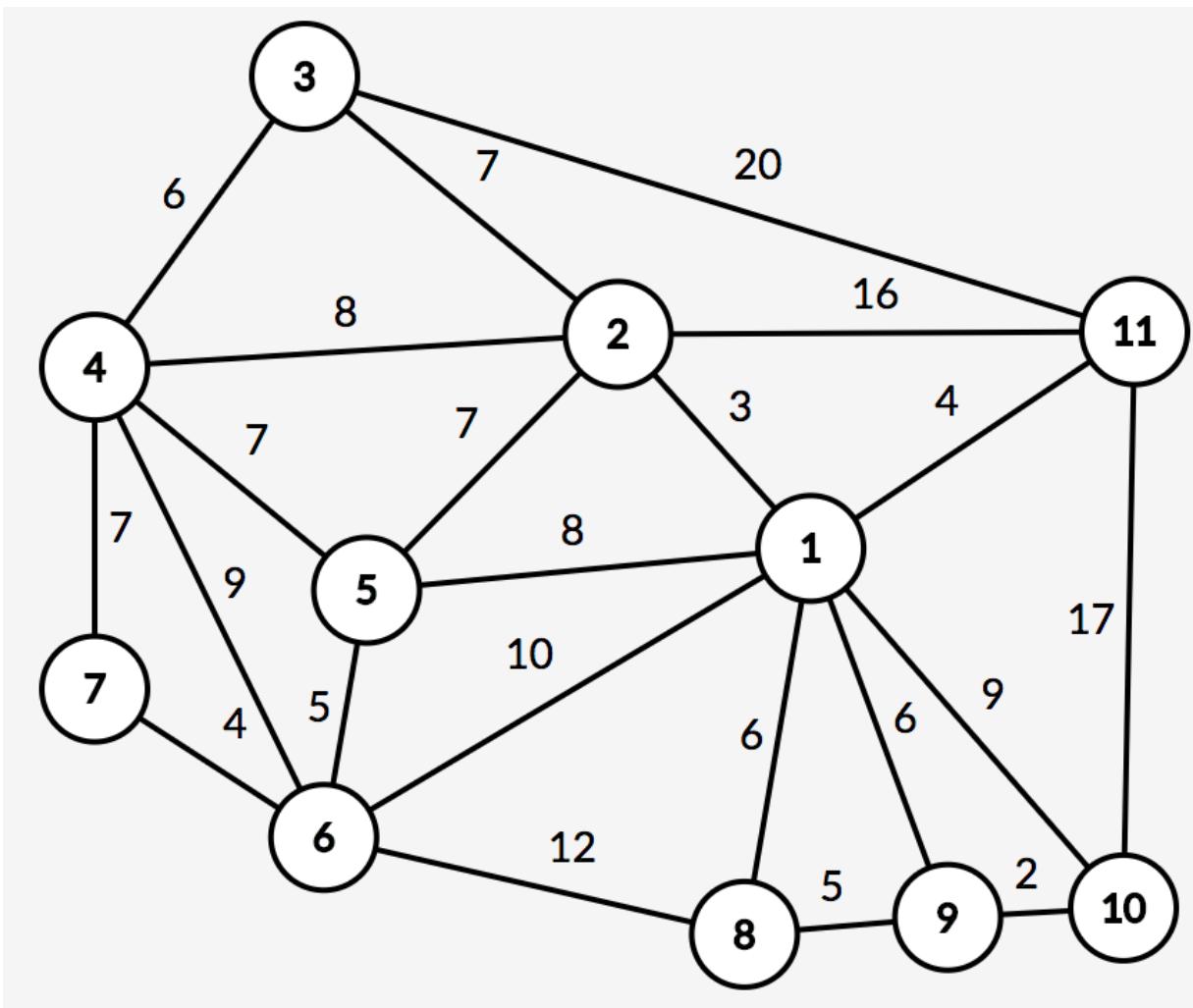
Ứng dụng

- Giả sử bạn là nhà cung cấp 1 trong các dịch vụ sau đây cho một số ngôi nhà ở trong 1 khu vực nào đó
 - Điện
 - Nước
 - Điện thoại
 - Truyền hình cáp
 - ...
- Để tiết kiệm chi phí, bạn cần 1 cây khung để nối các ngôi nhà lại với nhau.

Ứng dụng

- Xây dựng đường sắt, đường cao tốc với chi phí thấp nhất
- Thiết kế mạng máy tính cục bộ
- Thiết kế hệ thống điện (đi dây điện) cho 1 căn hộ

Bài tập



Bài tập

	A	B	C	D	E	F	G	H	K
A		1	1						1
B							9	2	8
C					6				2
D						1	5	4	3
E						2			
F							9	3	
G								9	
H									7
K									

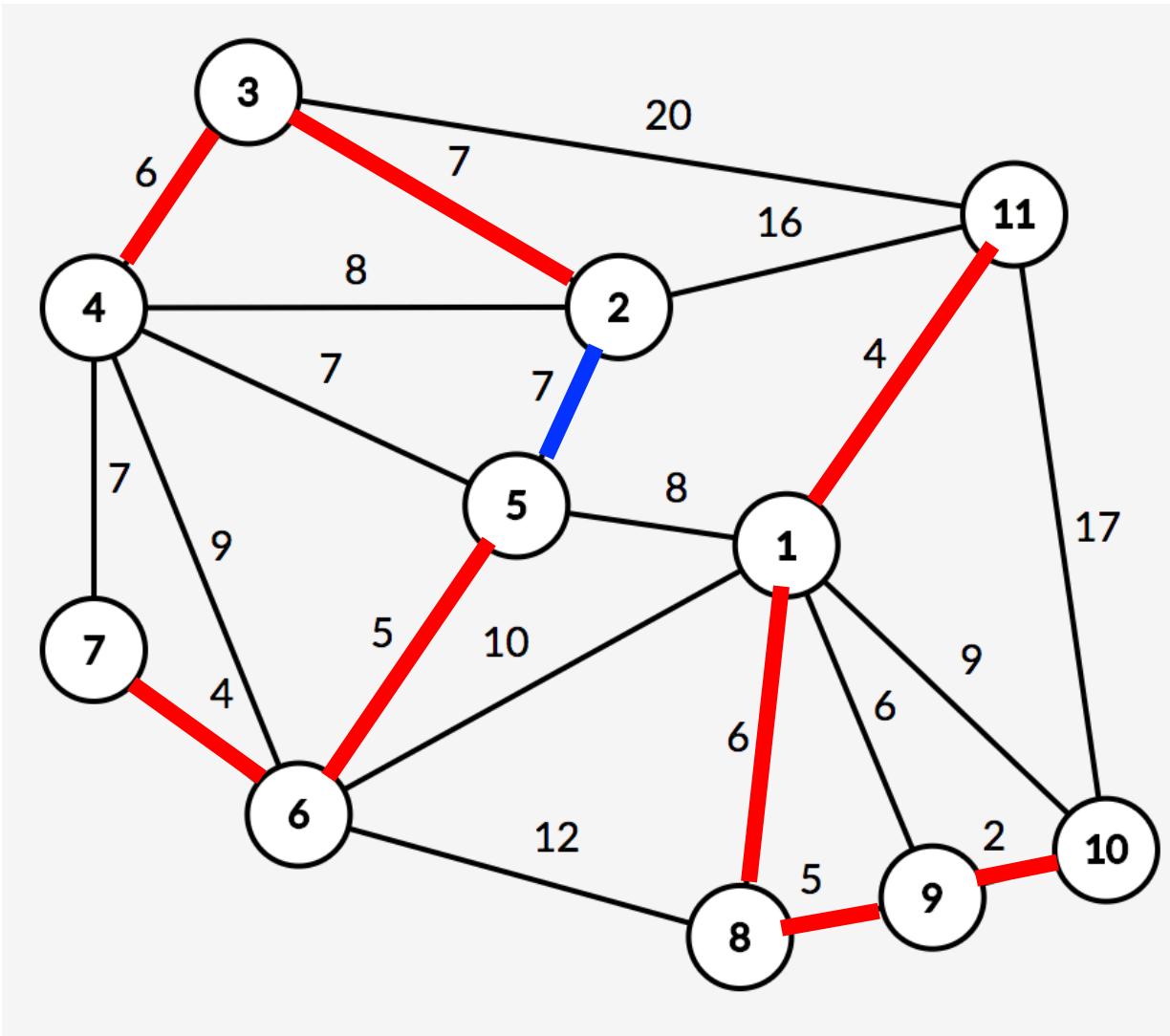
Tuần tới

- Giải thuật Kruskal
 - Kiểm tra việc tạo ra chu trình
- Giải thuật Prim
- Cây khung có hướng/cây có gốc
- Cây khung có hướng có trọng số nhỏ nhất

Giải thuật Kruskal (xem lại)

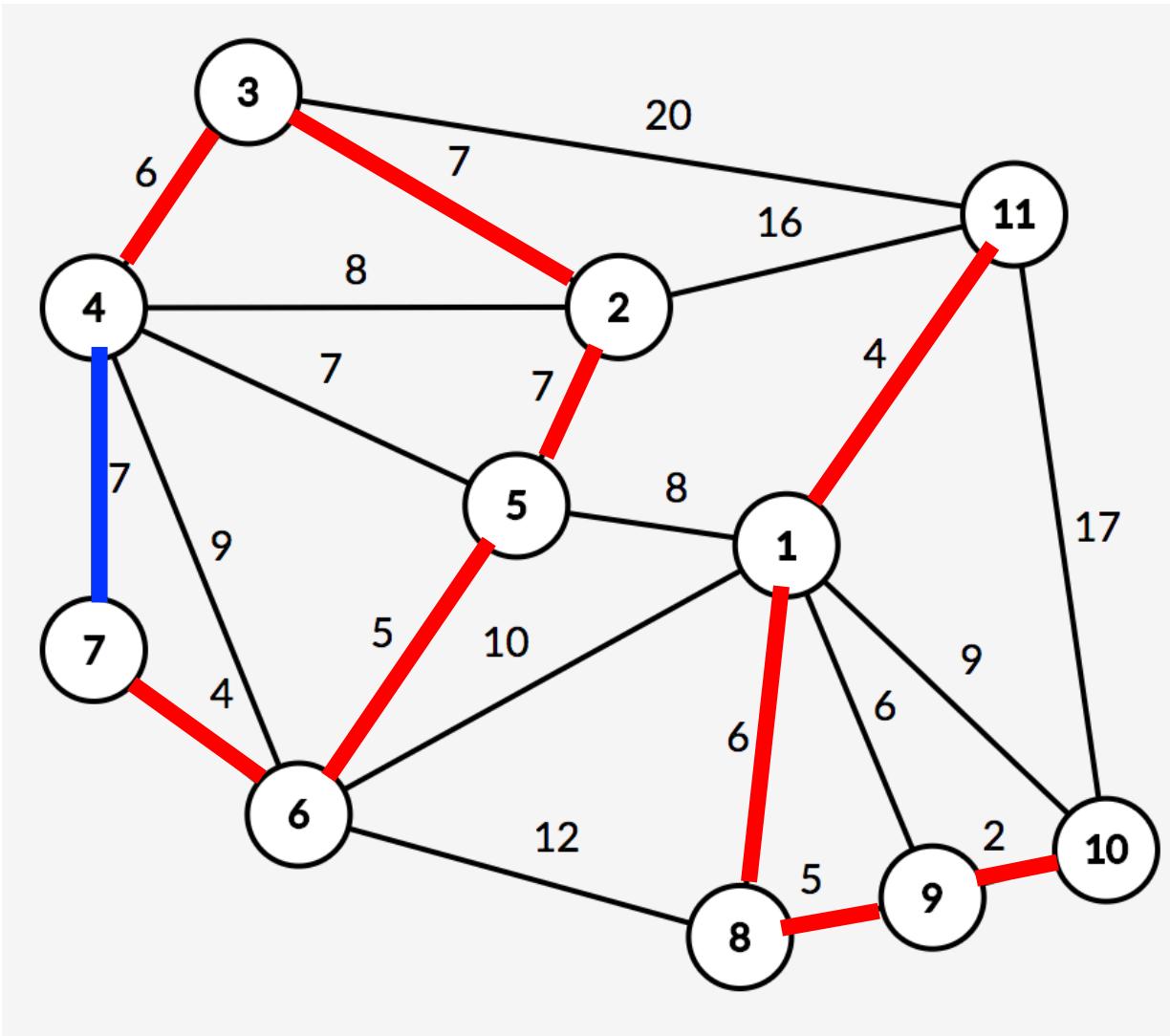
- Điểm mấu chốt của giải thuật Kruskal
 - Kiểm tra khi thêm một cung (u, v) vào cây T có tạo nên chu trình không.

Giải thuật Kruskal (xem lại)

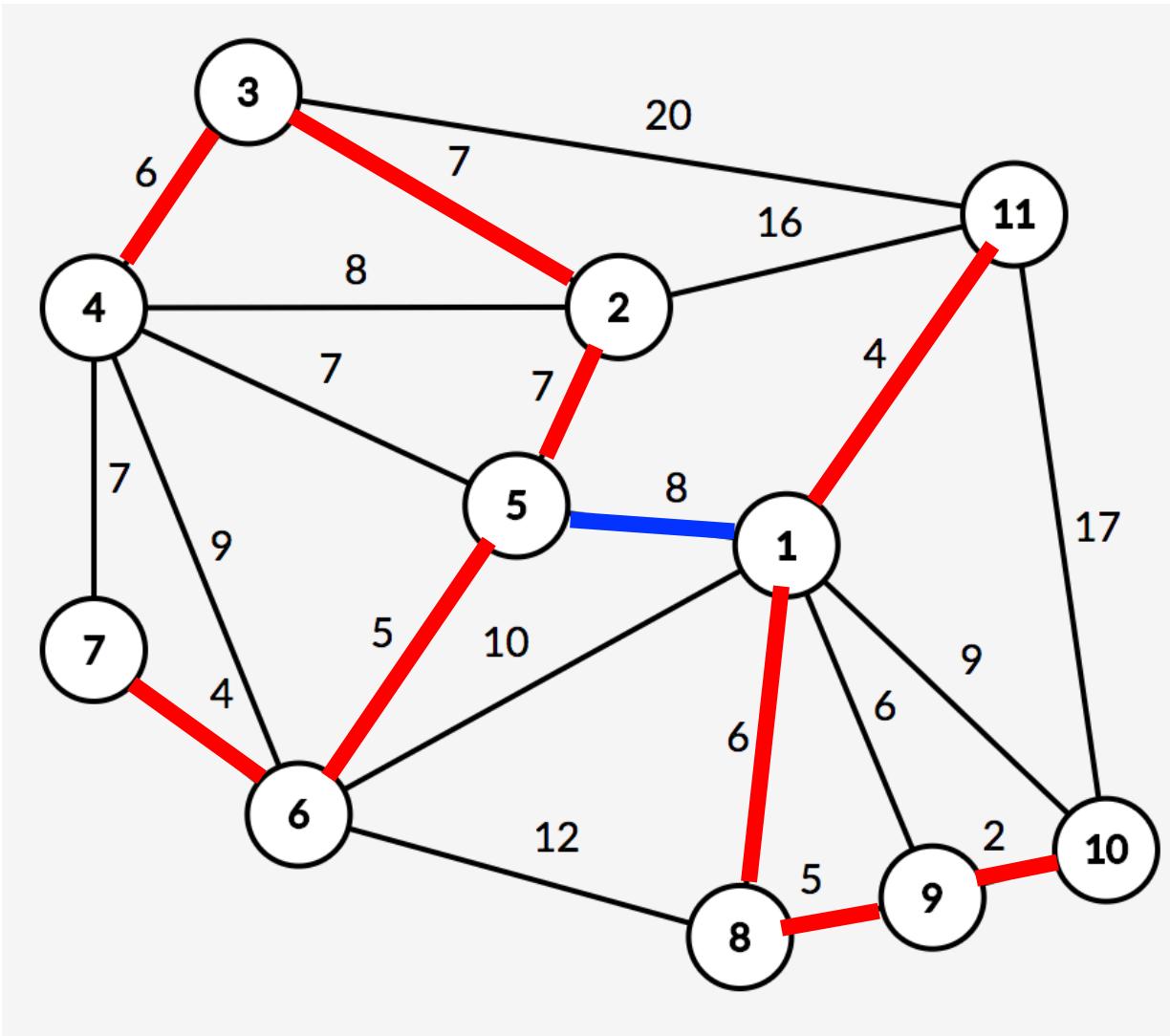


thêm cung (2, 5)

Giải thuật Kruskal (xem lại)



Giải thuật Kruskal (xem lại)



thêm cung (1, 5)

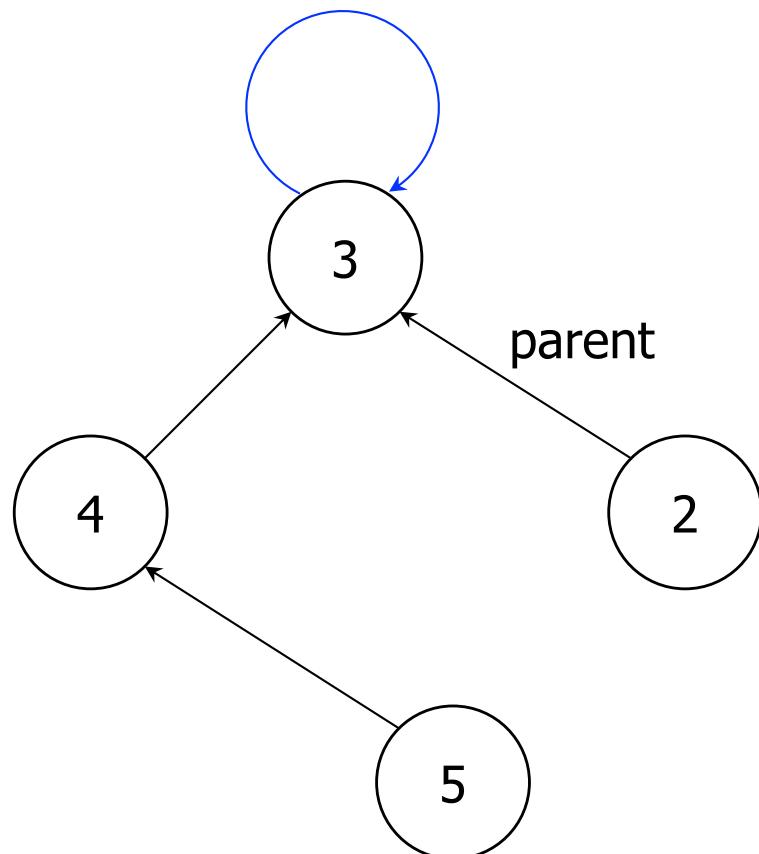
Giải thuật Kruskal (xem lại)

- Khi u và v nằm ở 2 BPLT khác nhau => không tạo chu trình
- Khi u và v ở cung 1 BPLT => tạo chu trình
- Bài toán kiểm tra chu trình khi thêm cung (u,v)
=> Bài toán tìm BPLT của u và v .

=> Quản lý các BPLT của cây T

- Tìm BPLT chứa đỉnh u
- Hợp 2 BPLT thành 1

Giải thuật Kruskal (xem lại)

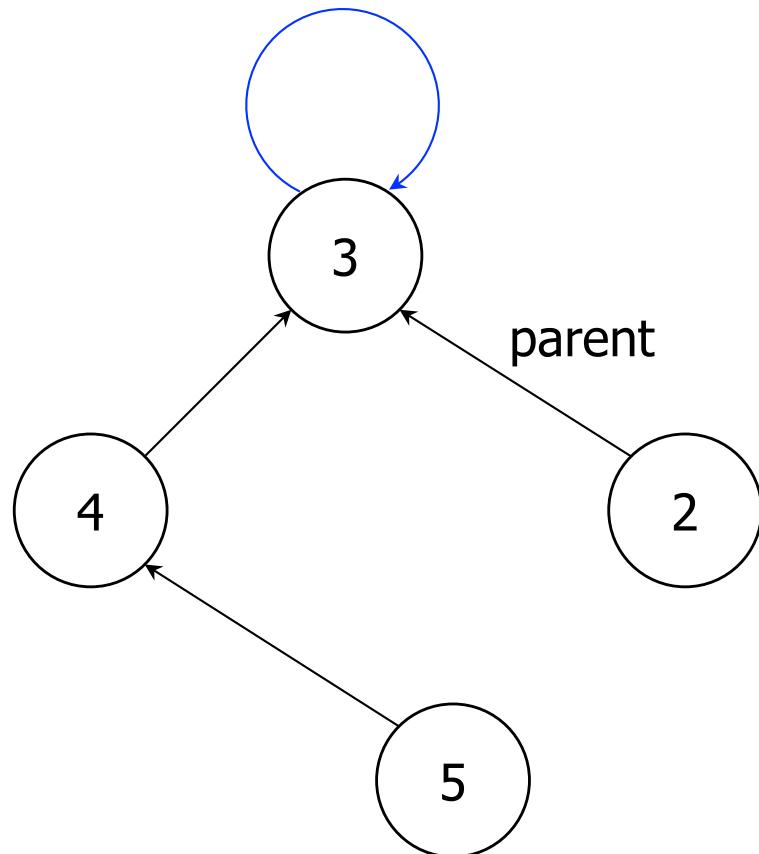


- 1 BPLT được tổ chức như 1 CTDL cây
- Đỉnh gốc: đại diện cho BPLT
- Mỗi đỉnh đều có 1 mũi tên chỉ về **đỉnh cha** (parent) của nó.
- Đỉnh gốc chỉ vào chính nó (có đỉnh cha là chính nó)

`int parent[MAXN];`

- $\text{parent}[3] = 3$
- $\text{parent}[2] = 3$
- $\text{parent}[4] = 3$
- $\text{parent}[5] = 4$

Giải thuật Kruskal (xem lại)

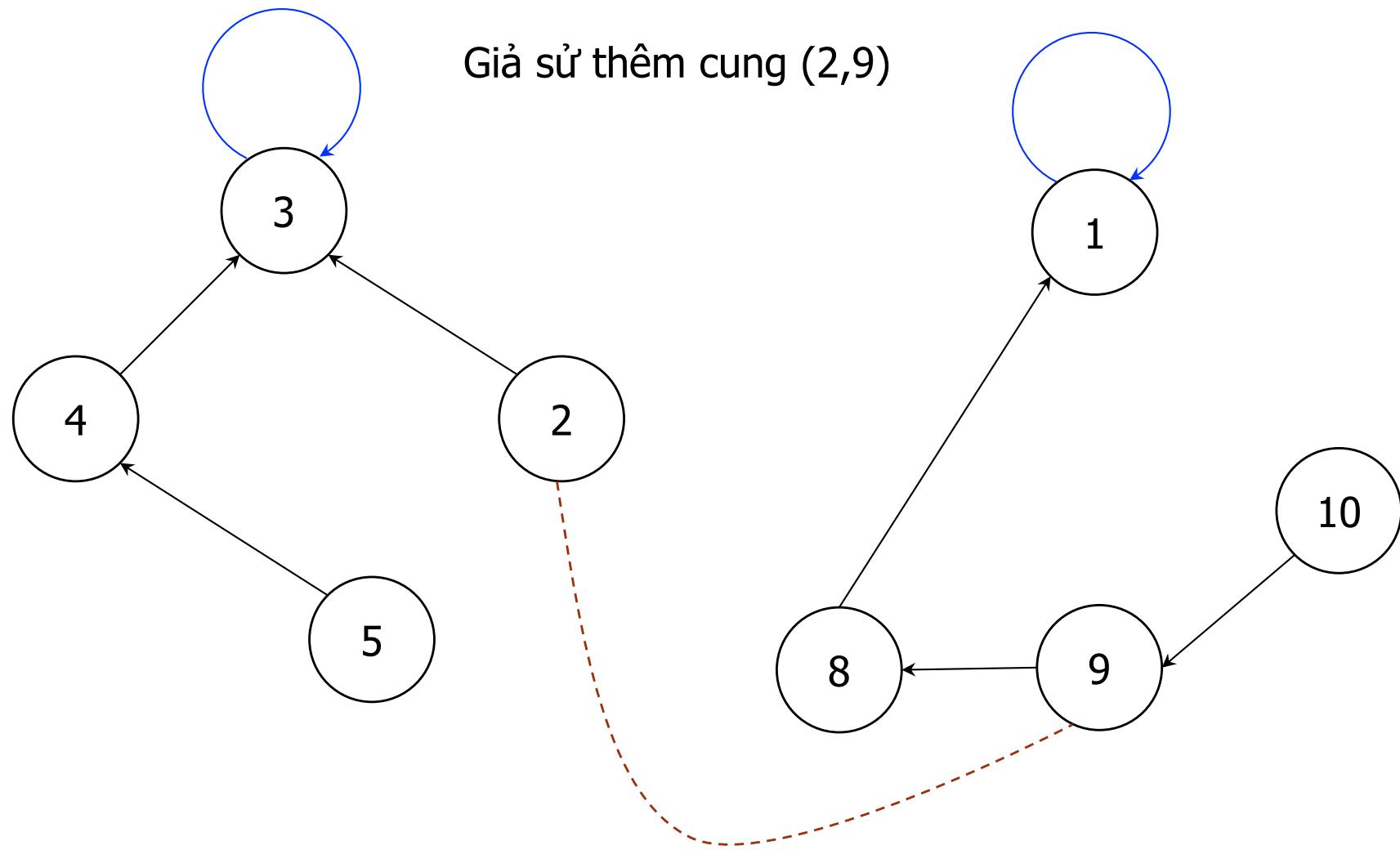


- **Tìm BPLT của u**

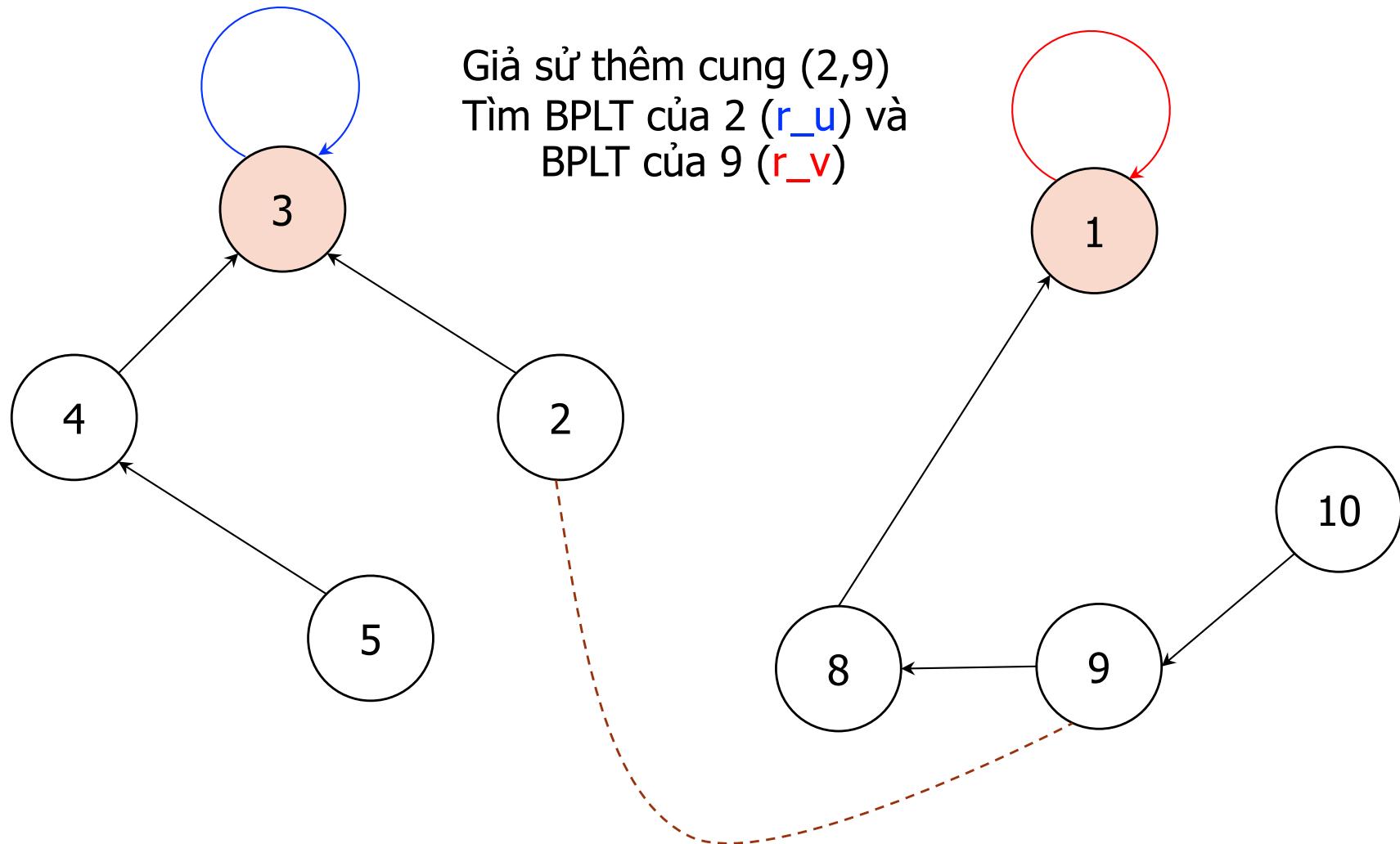
- Từ u lặp ngược theo `parent[u]`,
rồi `parent[parent[u]]`, ..., cho đến
khi gặp gốc.
- Vd: 5 -> 4 -> **3**

```
int find_root(int u) {  
    while (u != parent[u])  
        u = parent[u];  
    return u;  
}
```

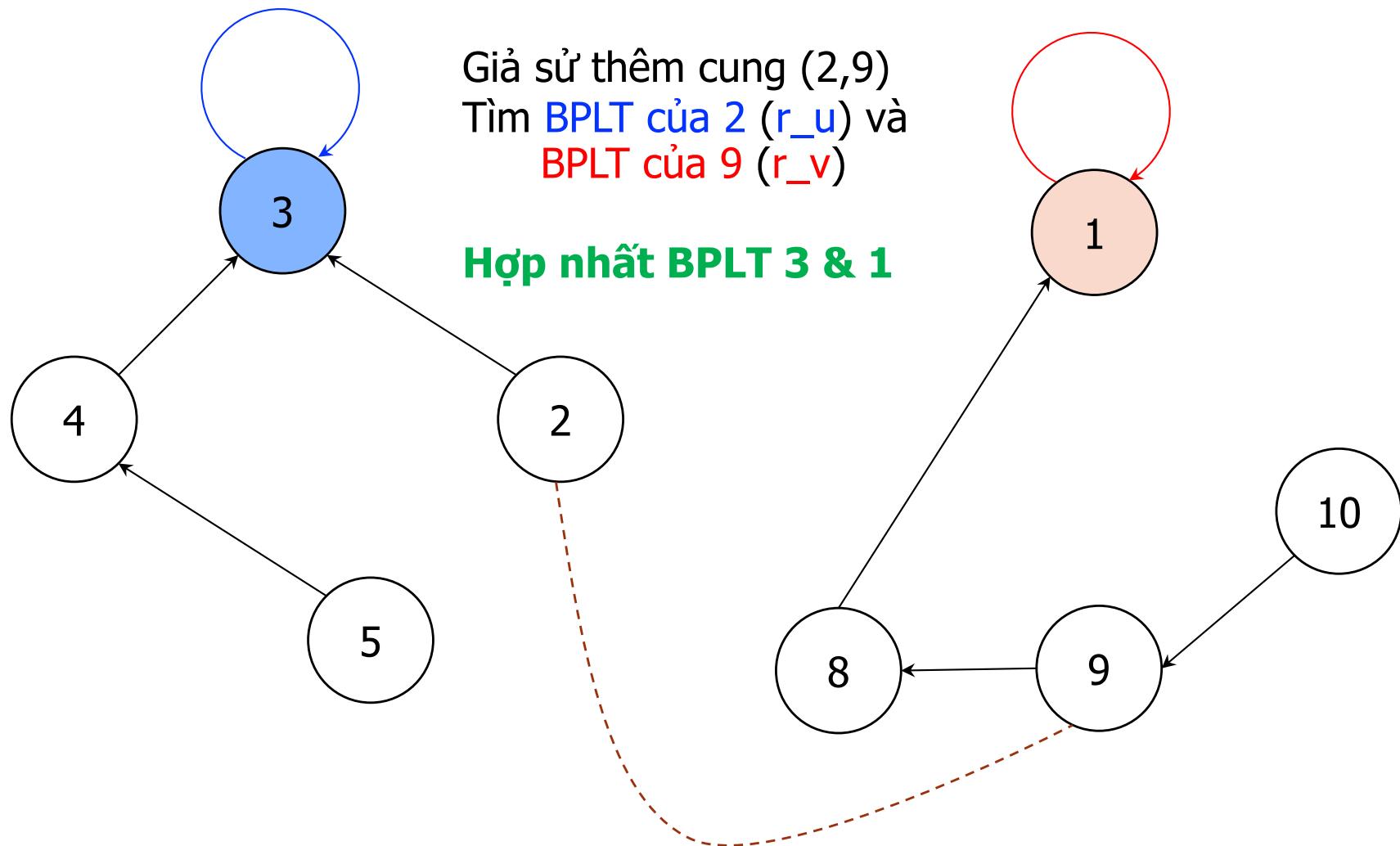
Giải thuật Kruskal (xem lại)



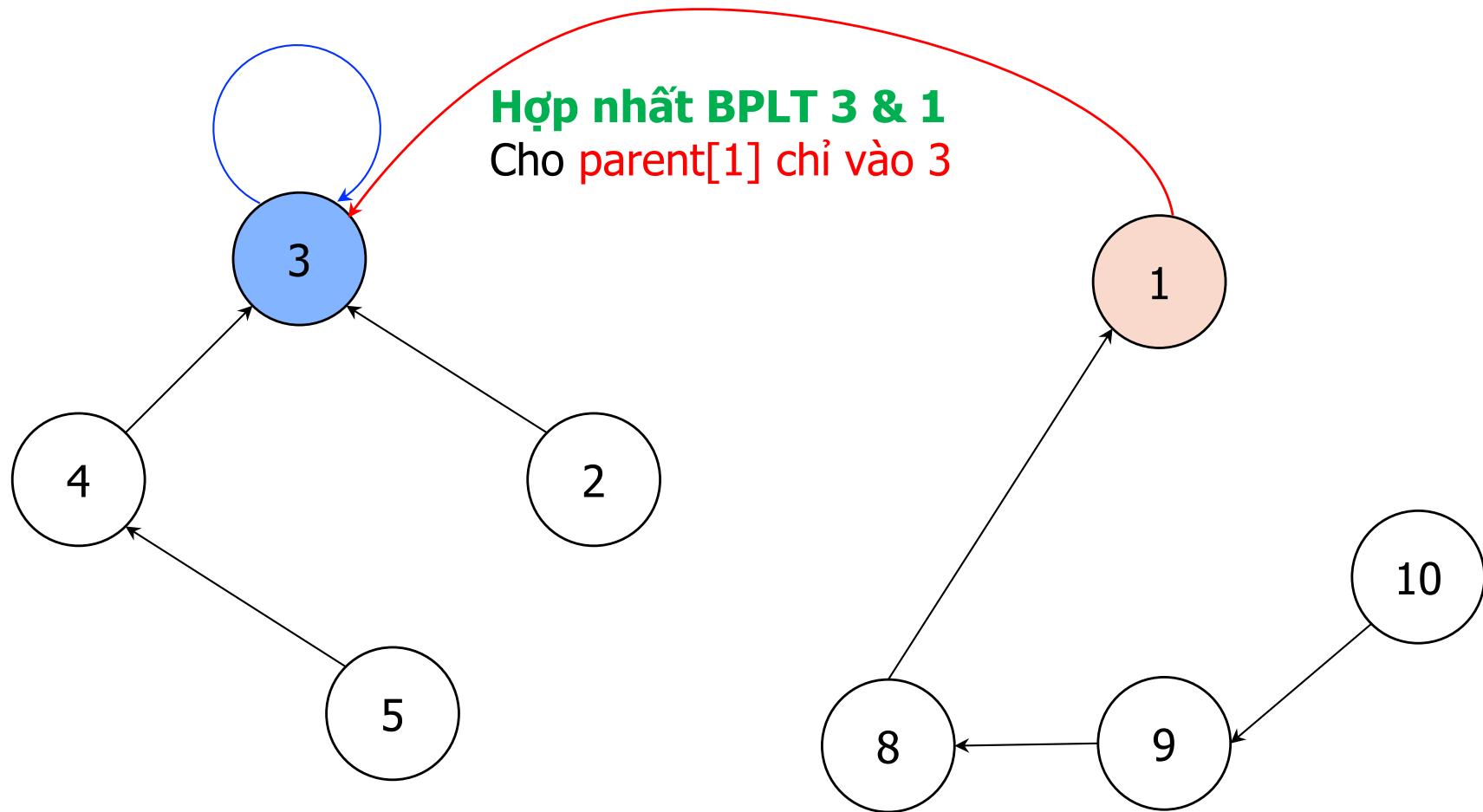
Giải thuật Kruskal (xem lại)



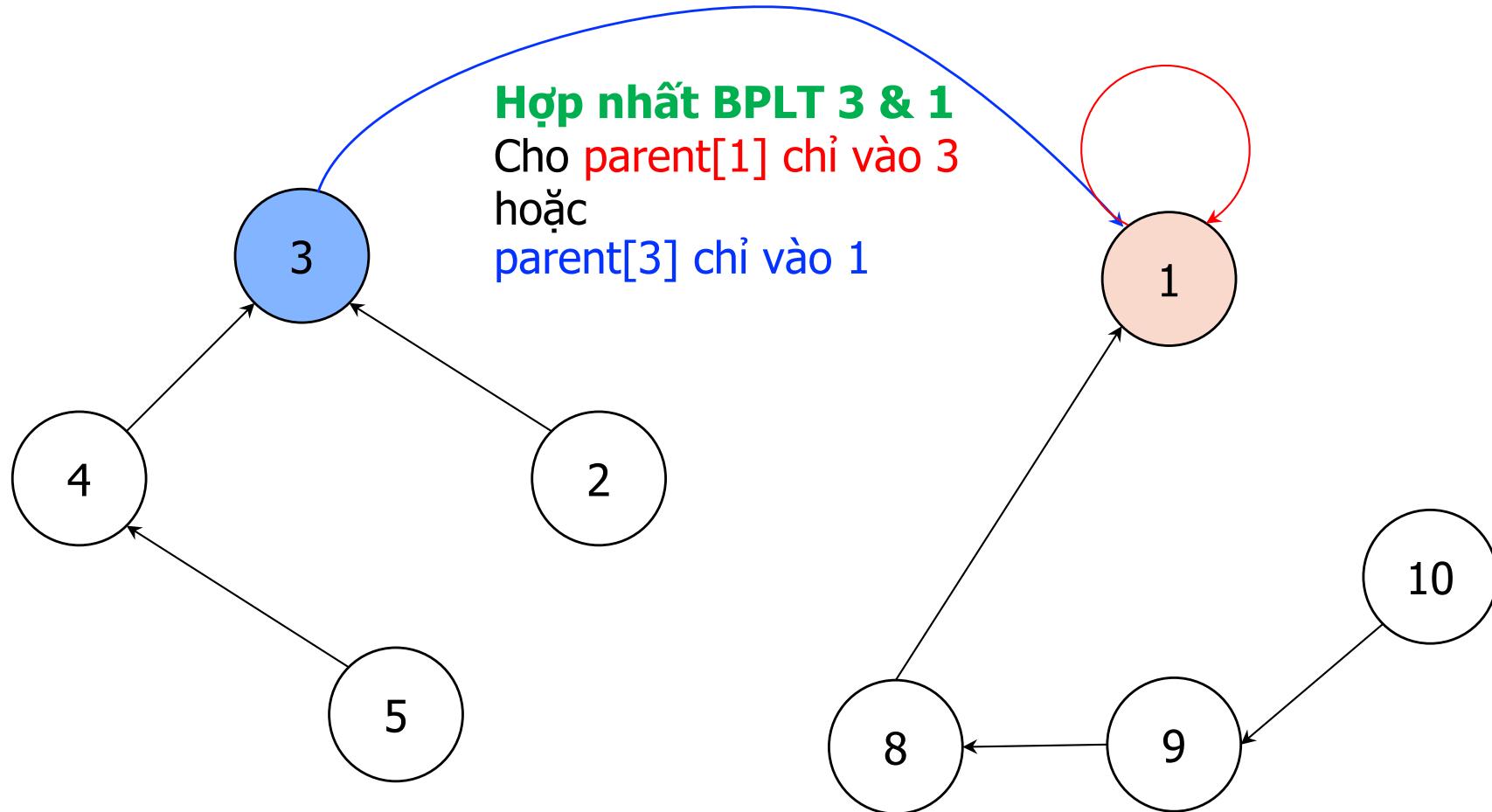
Giải thuật Kruskal (xem lại)



Giải thuật Kruskal (xem lại)



Giải thuật Kruskal (xem lại)



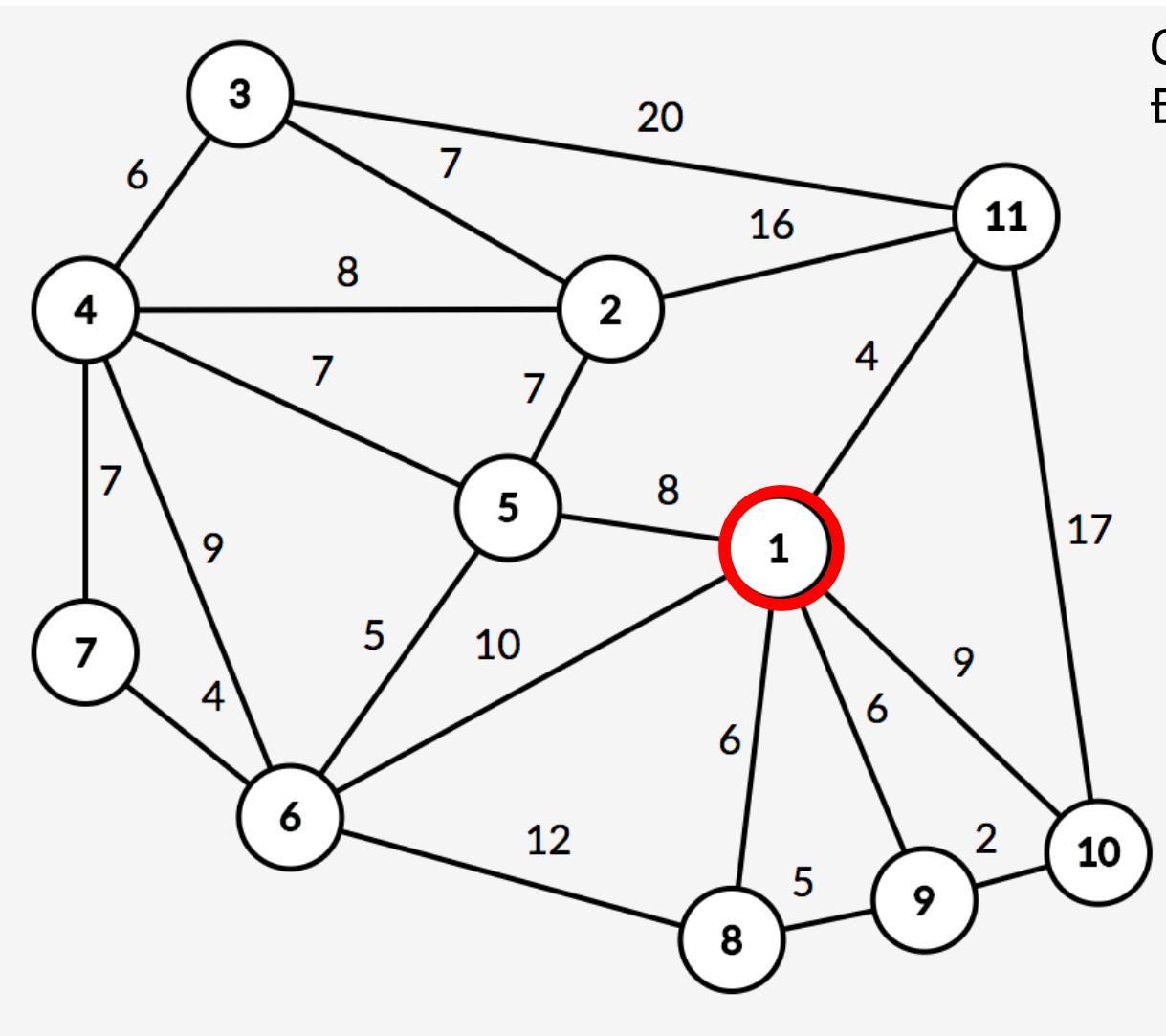
Giải thuật Kruskal (xem lại)

- Sắp xếp các cung của G theo thứ tự trọng số tăng dần
- Khởi tạo cây T rỗng (không chứa cung nào cả)
- **for** (các đỉnh u của G)
 - `parent[u] = u;` //Mỗi đỉnh là 1 BPLT
- Lần lượt xét từng cung của G
 - `r_u = find_root(u);` //r_u là gốc của u
 - `r_v = find_root(v);` //r_v là gốc của v
 - `if (r_u != r_v)` //2 BPLT khác nhau => không tạo CT
 - Thêm cung (u, v) vào T
 - `parent[r_v] = r_u;` //cho gốc của v làm con của gốc của u
- Điều kiện dừng:
 - T có $n - 1$ cung hoặc tất cả các cung của G đều đã được xét

Giải thuật Prim

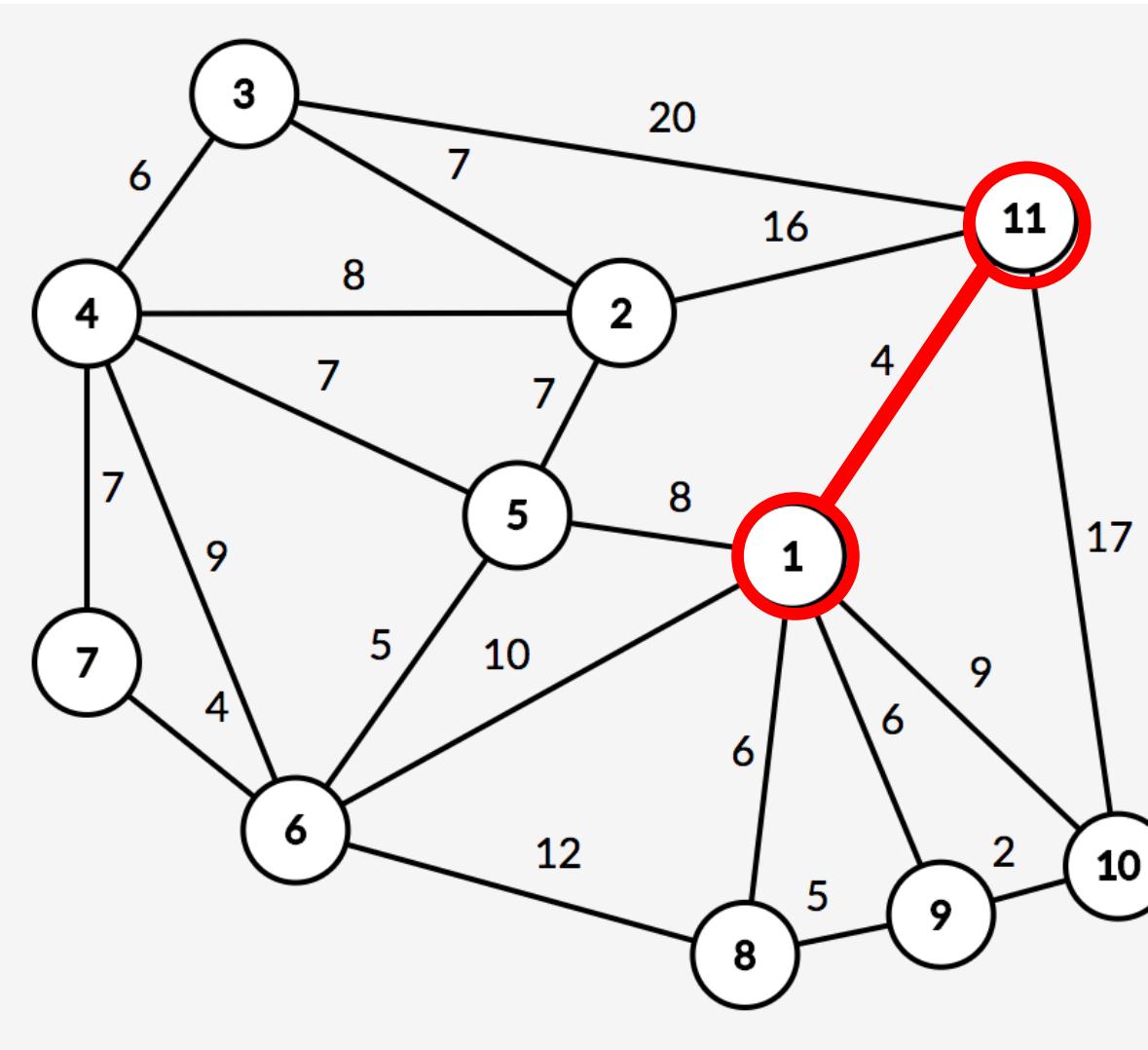
- Khởi tạo cây T rỗng
- Chọn 1 đỉnh u bất kỳ làm đỉnh bắt đầu, đánh dấu nó đã xét ($\text{mark}[u] = 1$), các đỉnh khác đều chưa xét.
- Lặp $n - 1$ lần
 - Tìm đỉnh v chưa xét, gần với 1 trong các đỉnh u đã xét nhất
 - Thêm cung (u, v) vào T
 - Đánh dấu v đã xét

Giải thuật Prim



Chọn 1 làm đỉnh bắt đầu
Đánh dấu nó đã xét

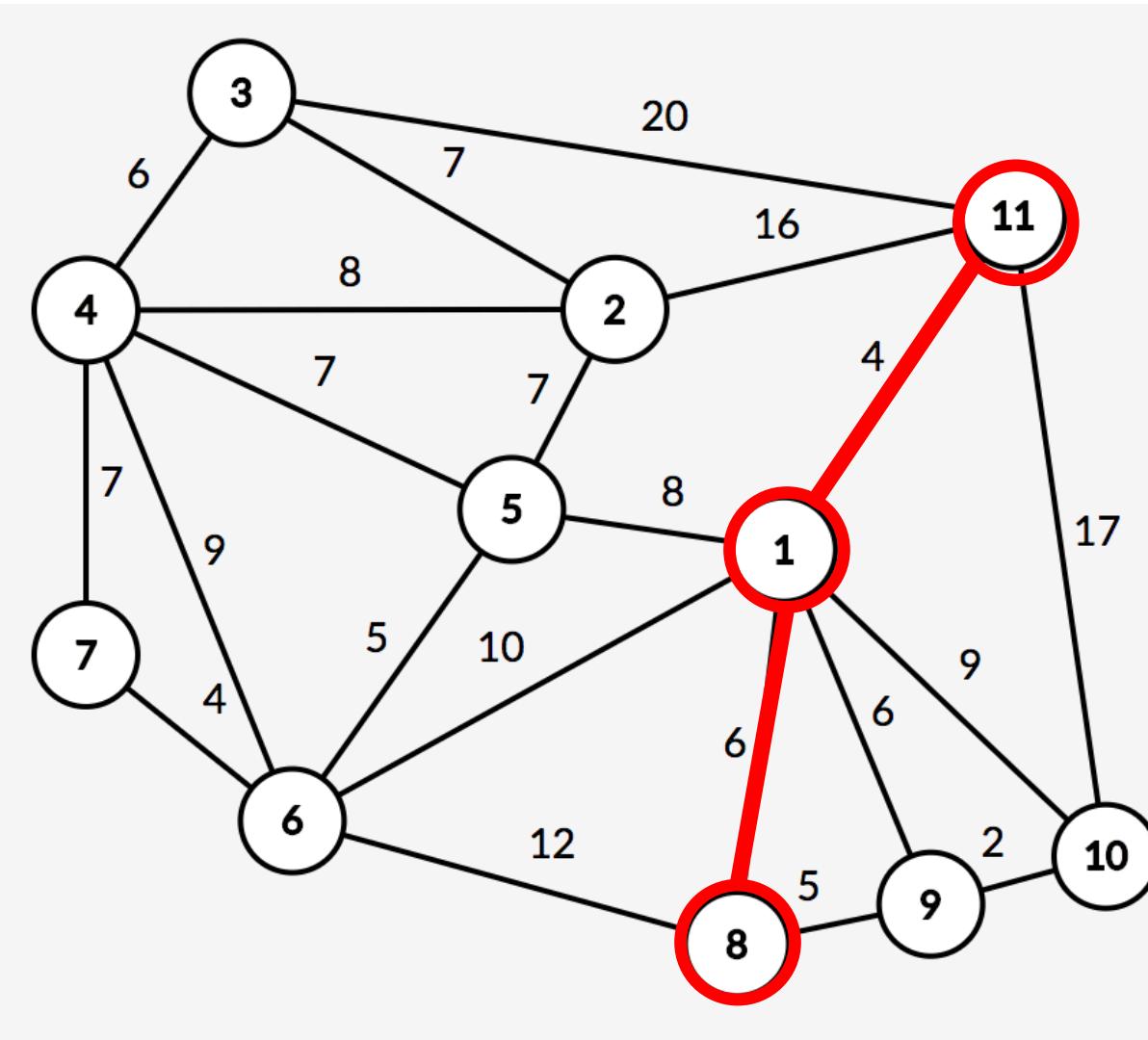
Giải thuật Prim



Chọn đỉnh chưa xét, gần với 1 nhất
=> 11 (11 gần với 1)

Thêm cung (1, 11) vào T
Đánh dấu 11 đã xét

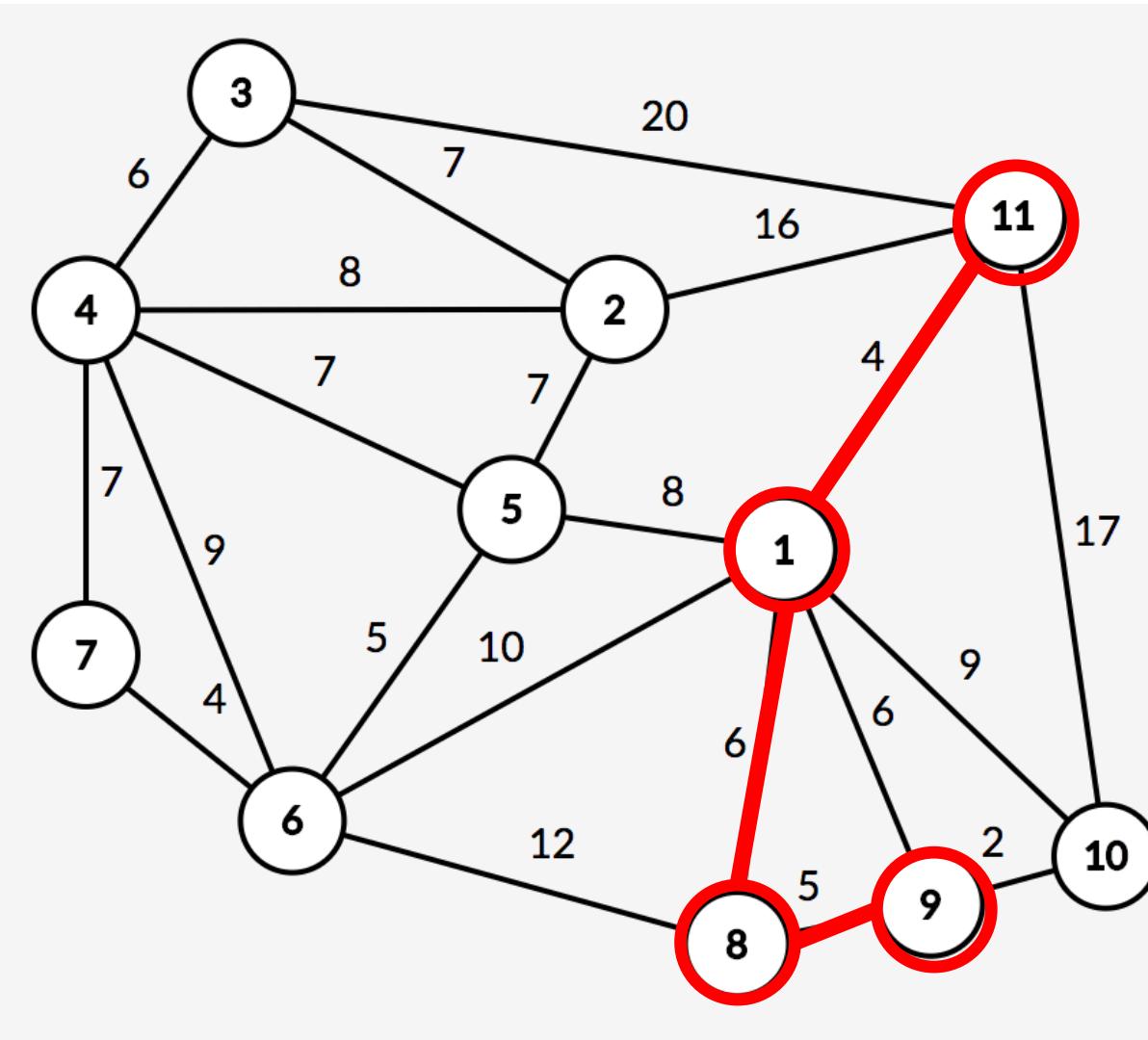
Giải thuật Prim



Chọn đỉnh chưa xét, gần với $\{1, 11\}$ nhất
=> 8 (8 gần với 1)

Thêm cung $(1, 8)$ vào T
Đánh dấu 8 đã xét

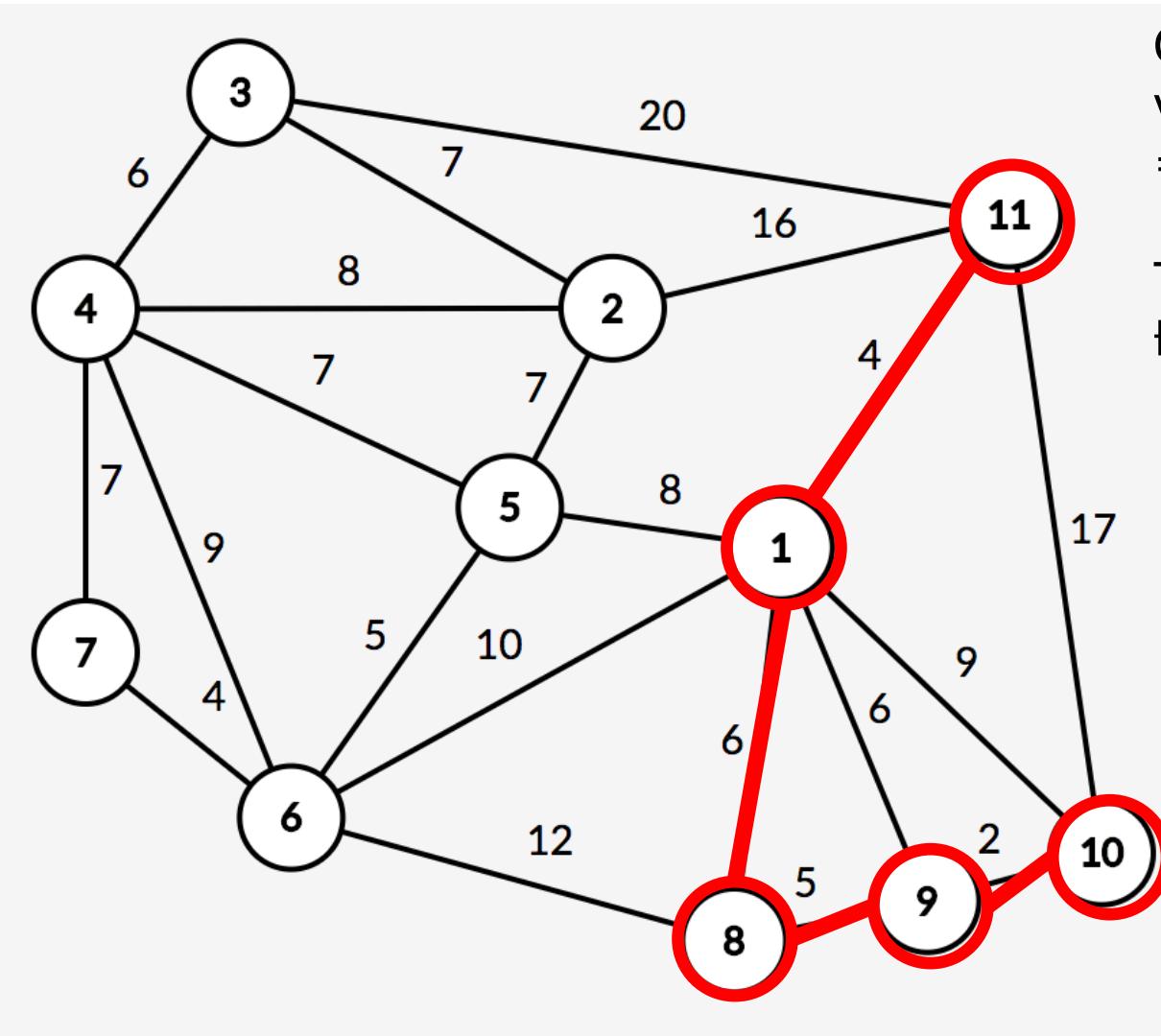
Giải thuật Prim



Chọn đỉnh chưa xét, gần với {1, 8, 11} nhất
=> 9 (9 gần với 8)

Thêm cung (8, 9) vào T
Đánh dấu 9 đã xét

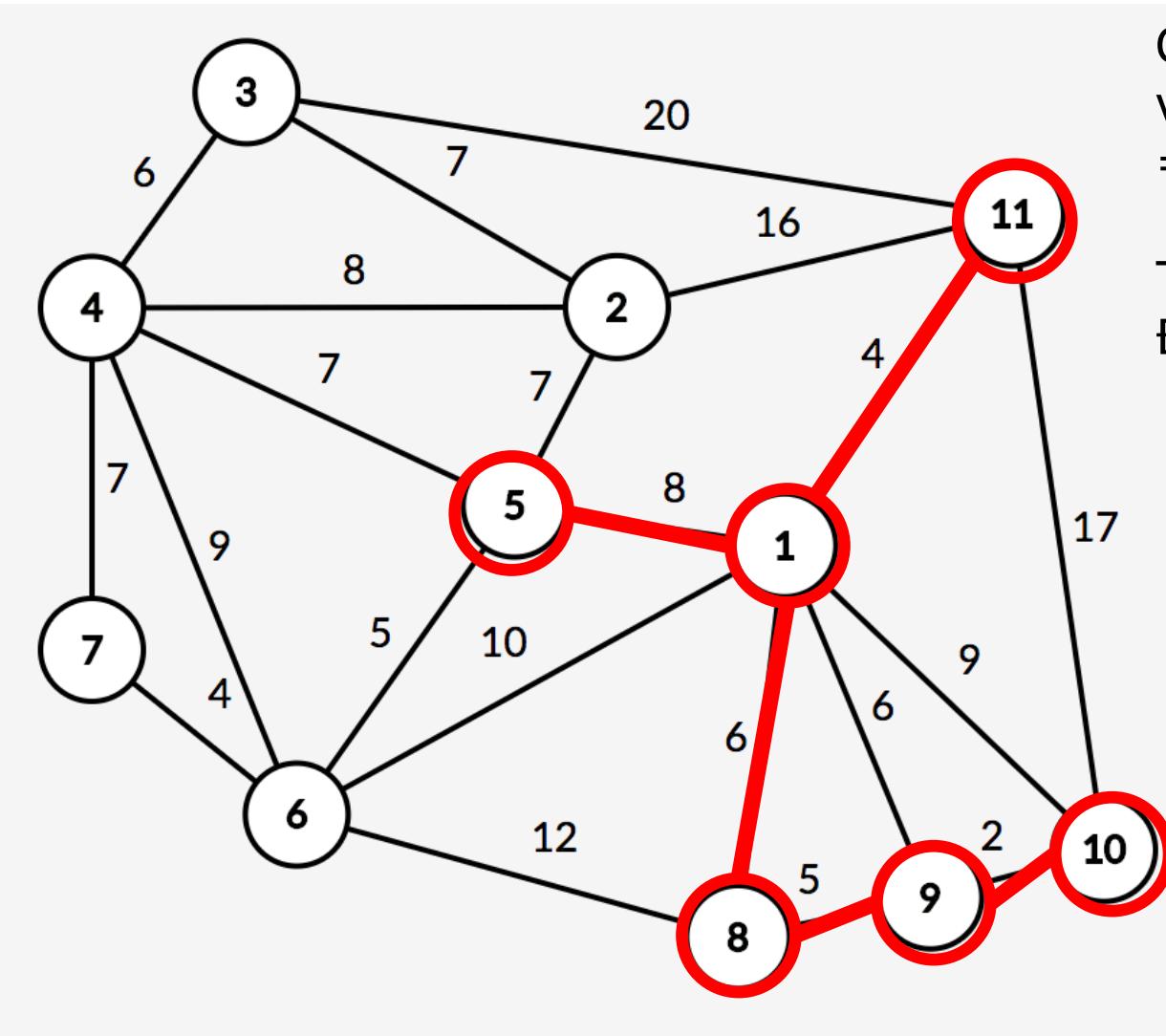
Giải thuật Prim



Chọn đỉnh chưa xét, gần với $\{1, 8, 9, 11\}$ nhất
=> 10 (10 gần với 9)

Thêm cung (9, 10) vào T
Đánh dấu 10 đã xét

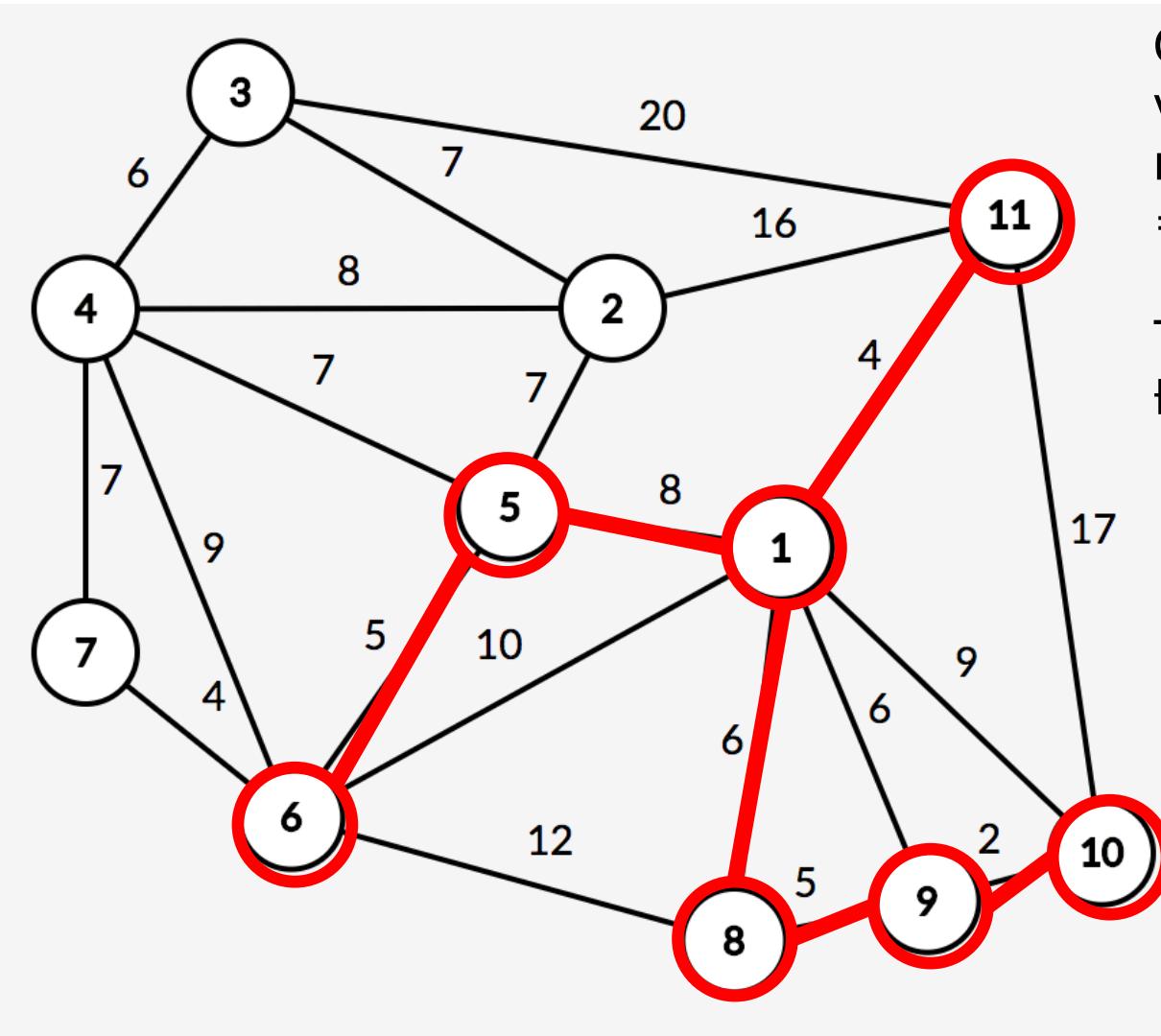
Giải thuật Prim



Chọn đỉnh chưa xét, gần với $\{1, 8, 9, 10, 11\}$ nhất
=> 5 (5 gần với 1)

Thêm cung (1, 5) vào T
Đánh dấu 5 đã xét

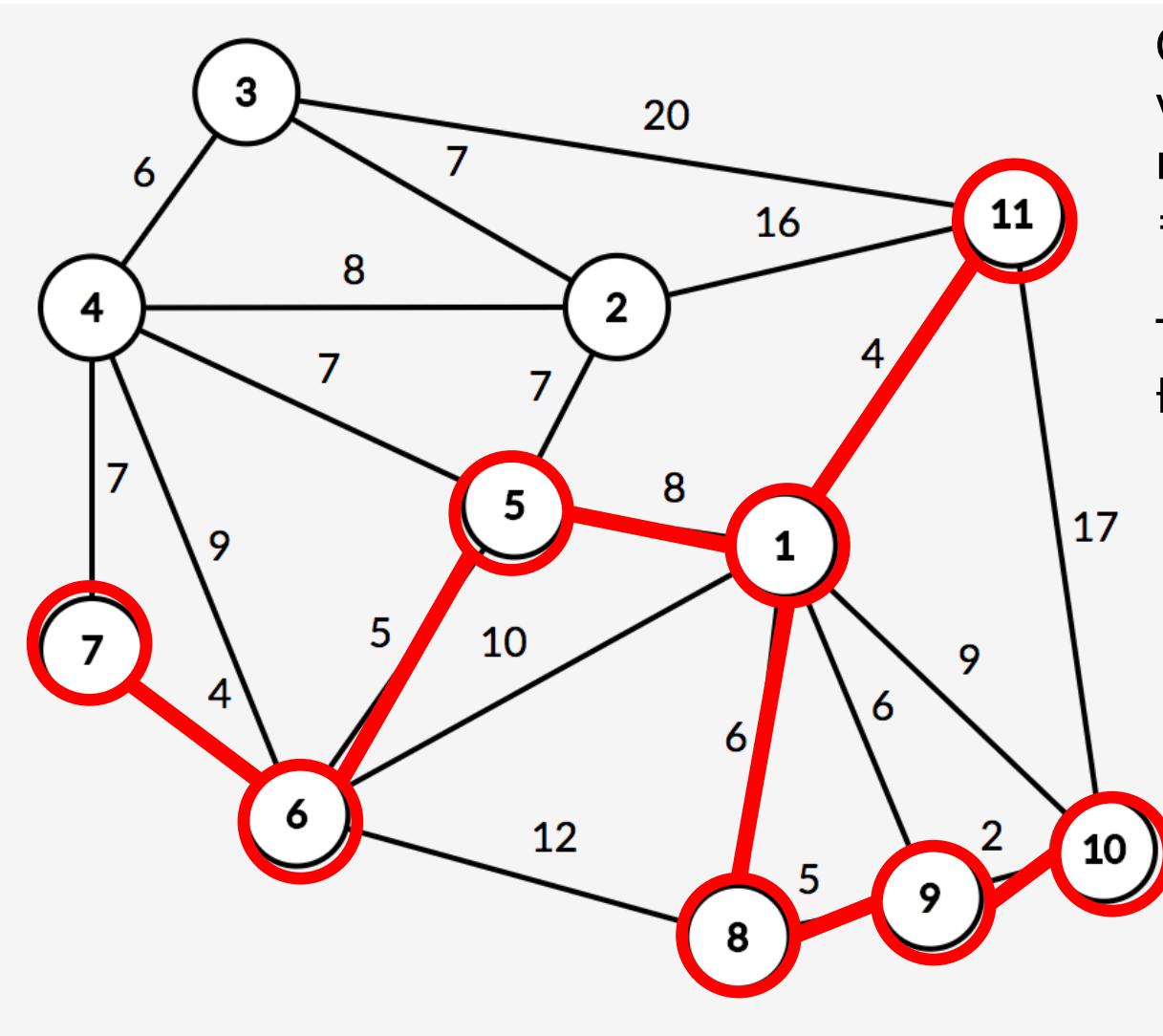
Giải thuật Prim



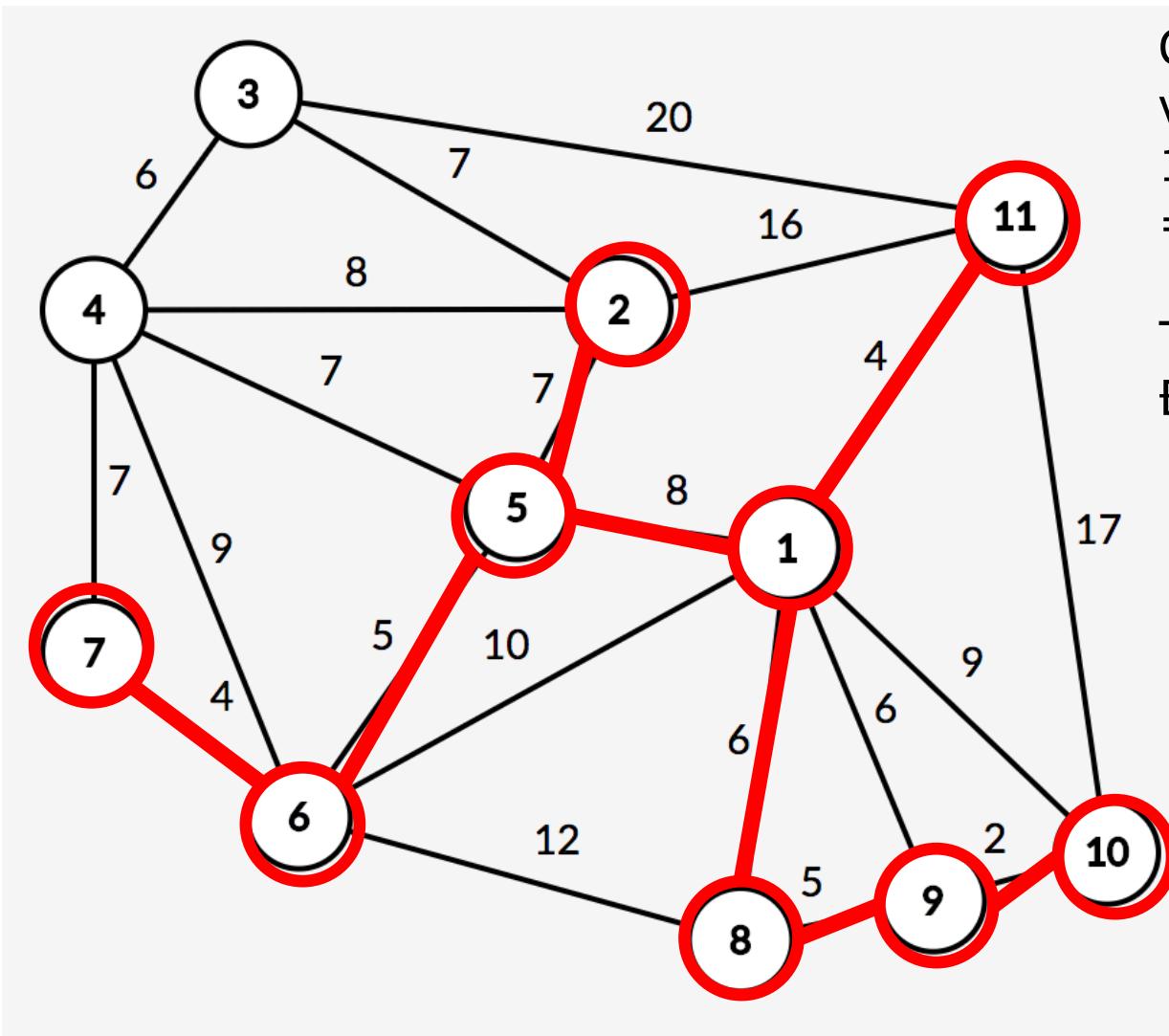
Chọn đỉnh chưa xét, gần với $\{1, 5, 8, 9, 10, 11\}$ nhất
=> 6 (6 gần với 5)

Thêm cung (5, 6) vào T
Đánh dấu 6 đã xét

Giải thuật Prim



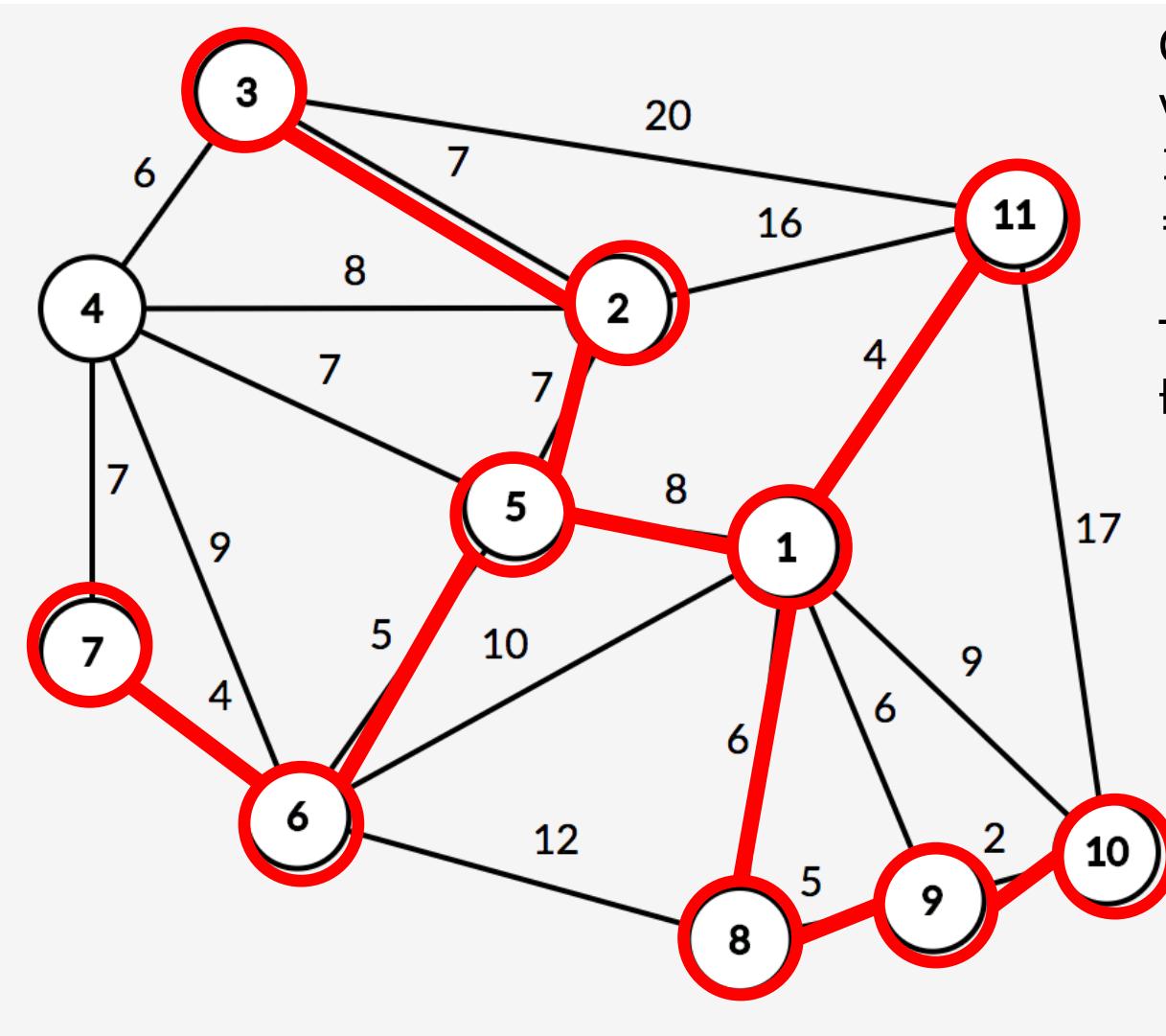
Giải thuật Prim



Chọn đỉnh chưa xét, gần với {1, 5, 6, 7, 8, 9, 10, 11} nhất
=> 2 (2 gần với 5)

Thêm cung (2, 5) vào T
Đánh dấu 2 đã xét

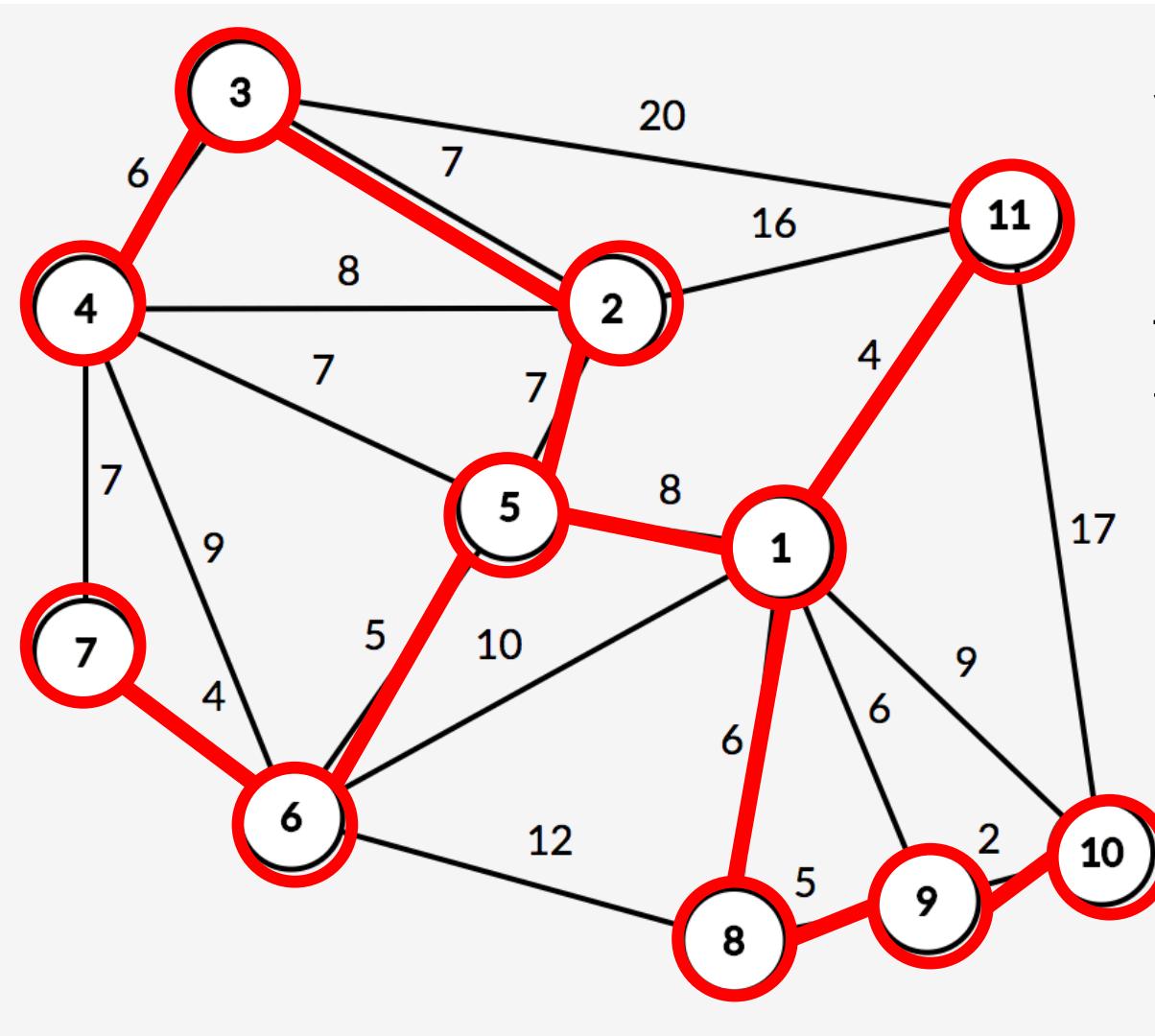
Giải thuật Prim



Chọn đỉnh chưa xét, gần với {1, 2, 5, 6, 7, 8, 9, 10, 11} nhất
=> 3 (3 gần với 2)

Thêm cung (2, 3) vào T
Đánh dấu 3 đã xét

Giải thuật Prim



Chọn đỉnh chưa xét, gần
với {1, 2, 3, 5, 6, 7, 8, 9,
10, 11} nhất
=> 4 (4 gần với 3)

Thêm cung (3, 4) vào T
Đánh dấu 4 đã xét

Giải thuật Prim

- Biểu diễn đồ thị bằng ma trận trọng số

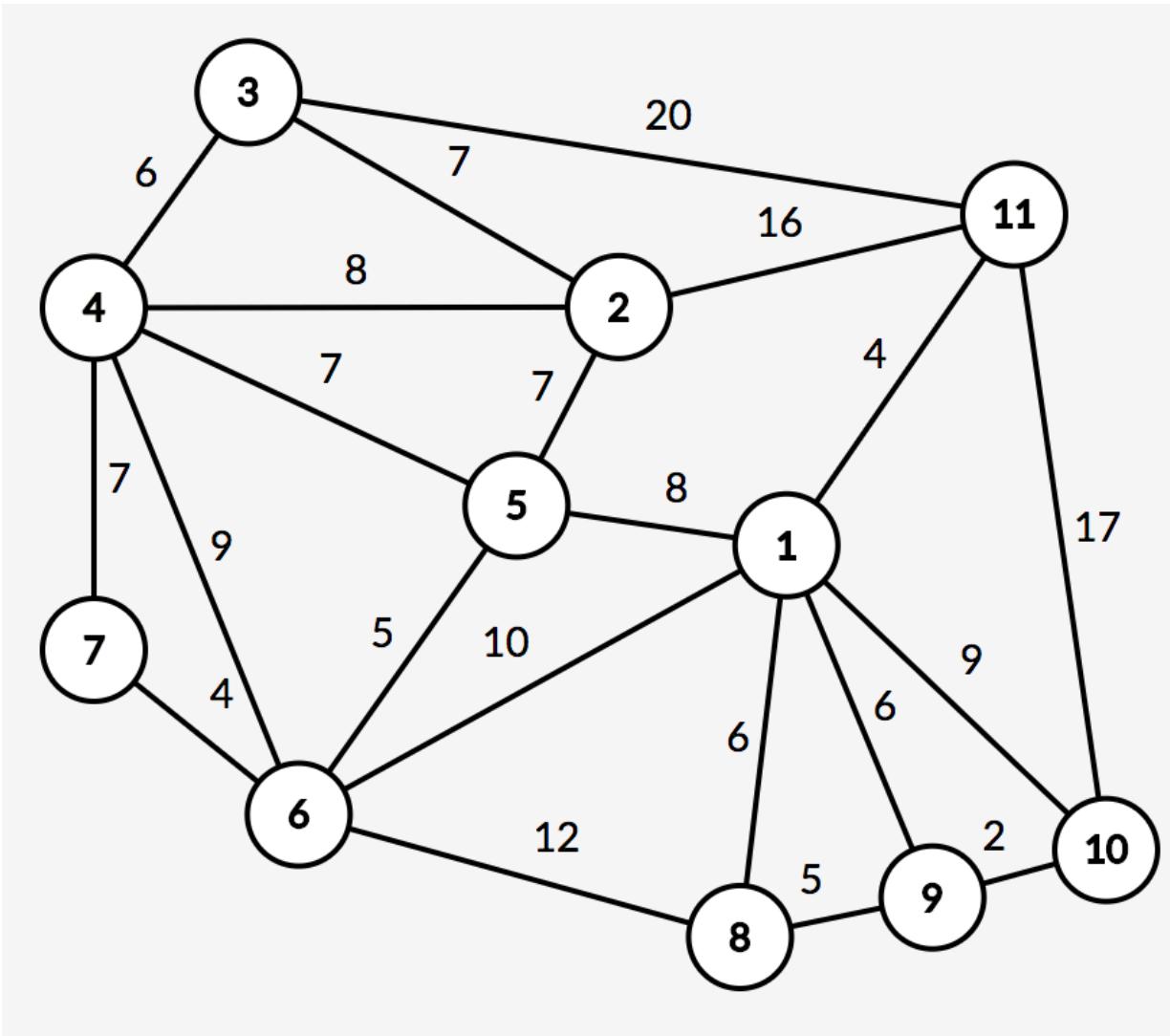
```
typedef struct {  
    int n, m;  
    int L[100][100];  
} Graph;
```

- Các biến hỗ trợ
 - mark[u]: đỉnh u đã xét (1) hay chưa (0)
 - pi[u]: khoảng cách ngắn nhất từ u đến 1 trong các đỉnh đã xét
 - p[u]: đỉnh đã xét gần với u nhất

Giải thuật Prim

- Khởi tạo
 - Với mọi u , $\text{mark}[u] = 0$; $\pi[u] = \infty$
 - $\text{mark}[1] = 1$
 - for (các đỉnh kề v của 1)
 - $\pi[v] = L[1][v]$; //khoảng cách từ v đến 1 = TS cung $(1, v)$
 - $p[v] = 1$; //1 là đỉnh đã xét gần với v nhất
- Lặp $n - 1$ lần
 - Chọn đỉnh v chưa xét, có $\pi[v]$ nhỏ nhất
 - $\text{mark}[v] = 1$; //Đánh dấu v đã xét
 - Thêm cung $(p[v], v)$ vào T
 - for (các đỉnh kề x chưa xét của v)
 - Cập nhật lại $\pi[x]$ và $p[x]$ (nếu $L[v][x] < \pi[x]$)

Giải thuật Prim – Ví dụ



Giải thuật Prim – Ví dụ

Bước	1	2	3	4	5	6	7	8	9	10	11	Công việc
KT	*	$\pi[2]=\infty$	$\pi[3]=\infty$	$\pi[4]=\infty$	$\pi[5]=8$ $p[5]=1$	$\pi[6]=10$ $p[6]=1$	$\pi[7]=\infty$	$\pi[8]=6$ $p[8]=1$	$\pi[9]=6$ $p[9]=1$	$\pi[10]=9$ $p[10]=1$	$\pi[11]=4$ $p[11]=1$	Khởi tạo

Giải thuật Prim – Ví dụ

Bước	1	2	3	4	5	6	7	8	9	10	11	Công việc
KT	*	$\pi[2]=\infty$	$\pi[3]=\infty$	$\pi[4]=\infty$	$\pi[5]=8$ $p[5]=1$	$\pi[6]=10$ $p[6]=1$	$\pi[7]=\infty$	$\pi[8]=6$ $p[8]=1$	$\pi[9]=6$ $p[9]=1$	$\pi[10]=9$ $p[10]=1$	$\pi[11]=4$ $p[11]=1$	Khởi tạo
1		$\pi[2]=16$ $p[2]=11$	$\pi[2]=20$ $p[2]=11$							Không cập nhật (17>9)	*	Chọn $v=11$, $p[v]=1$ Thêm cung (1,11) Cập nhật 2,3
2						Không cập nhật (12>10)		*	$\pi[9]=5$ $p[9]=8$			Chọn $v=8$, $p[v]=1$ Thêm cung (1,8) Cập nhật 9

Có 1 chỗ sai
trong slide này.
Hãy tìm nó 😊

Bài tập

- Cho đồ thị như hình bên
 - Áp dụng giải thuật Kruskal tìm MST.
 - Áp dụng giải thuật Prim tìm MST.
 - Áp dụng giải thuật Dijkstra tìm đường đi ngắn nhất từ đỉnh A đến các đỉnh khác. Vẽ cây đường đi ngắn nhất.
 - So sánh **cây kết quả** của giải thuật Kruskal/Prim với giải thuật Dijkstra.
 - Có thể dùng giải thuật Dijkstra để tìm MST được không

