CHƯƠNG 7

CÂY VÀ CÂY KHUNG CỰC TIỂU CỦA ĐỒ THỊ

Nội dung

1 Bài toán cây khung cực tiểu MST

2 Khái niệm cây và cây khung đồ thị

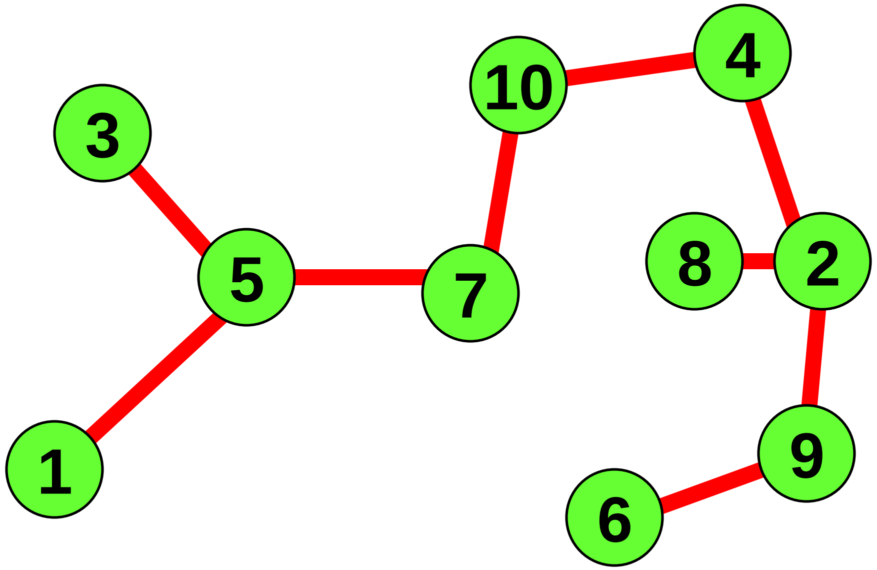
3 Thuật toán Prim

4 Thuật toán Kruskal

5 Thảo luận & Bài tập

Ứng dụng thực tế: Mạng truyền thông

- Công ty truyền thông AT&T cần xây dựng mạng truyền thông kết nối *n* khách hàng. Chi phí thực hiện kênh nối *i* và *j* là c_{ij}. Hỏi chi phí nhỏ nhất để thực hiện việc kết nối tất cả các khách hàng là bao nhiêu?



Giả thiết là: Chỉ có cách kết nối duy nhất là đặt kênh nối trực tiếp giữa hai nút.

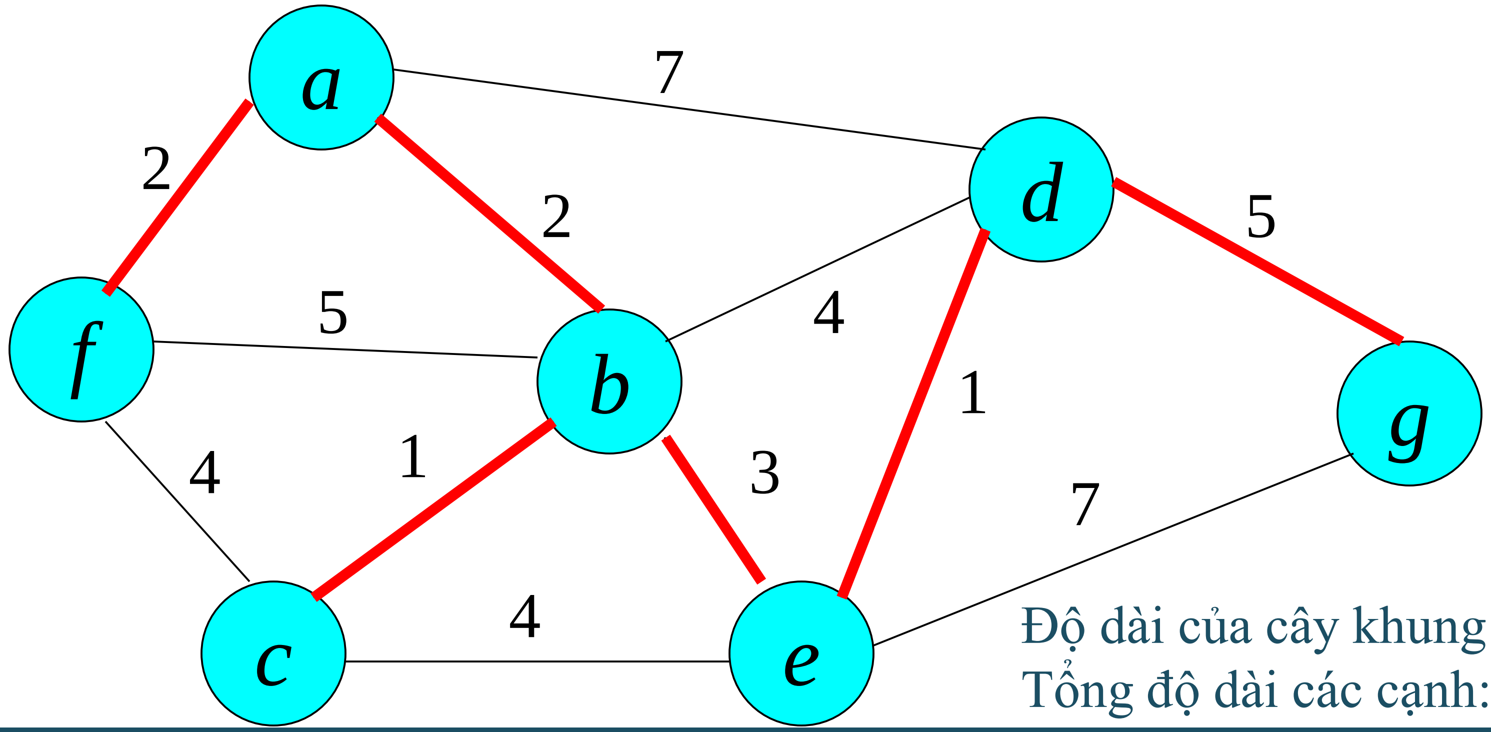
Bài toán xây dựng hệ thống đường sắt

- Giả sử ta muốn xây dựng một hệ thống đường sắt nối n thành phố sao cho hành khách có thể đi lại giữa hai thành phố bất kỳ đồng thời tổng chi phí xây dựng phải là nhỏ nhất.
- Rõ ràng là đồ thị mà đỉnh là các thành phố còn các cạnh là các tuyến đường sắt nối các thành phố tương ứng với phương án xây dựng tối ưu phải là cây.
- Vì vậy, bài toán đặt ra dẫn về bài toán tìm cây khung nhỏ nhất trên đồ thị đầy đủ n đỉnh, mỗi đỉnh tương ứng với một thành phố, với độ dài trên các cạnh chính là chi phí xây dựng đường ray nối hai thành phố tương ứng

- Chú ý: *Trong bài toán này ta giả thiết là không được xây dựng tuyến đường sắt có các nhà ga phân tuyến nằm ngoài các thành phố.*

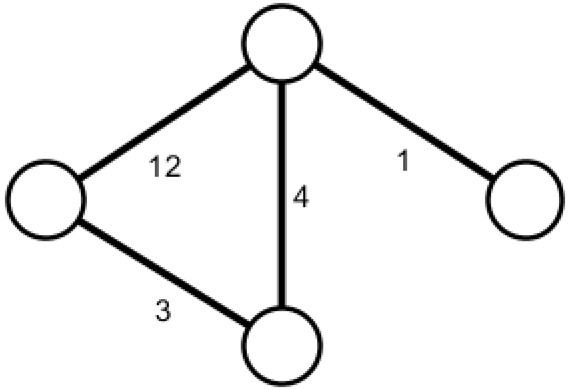
Bài toán CKCT

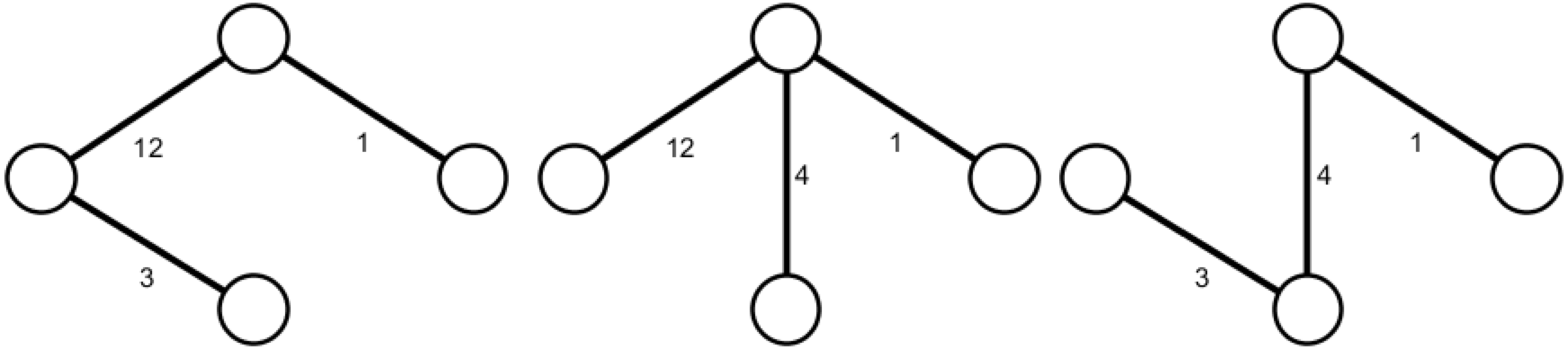
Bài toán: Cho đồ thị vô hướng liên thông *G*=(*V*,*E*) với trọng số *c*(*e*), *e* ∈ *E*. Độ dài của cây khung là tổng trọng số trên các cạnh của nó. Cần tìm cây khung có độ dài nhỏ nhất.



Độ dài của cây khung là
Tổng độ dài các cạnh: 14

Bài toán CKCT





C(T)=16 C(T)=17 **C(T*)=8**

Bài toán CKCT

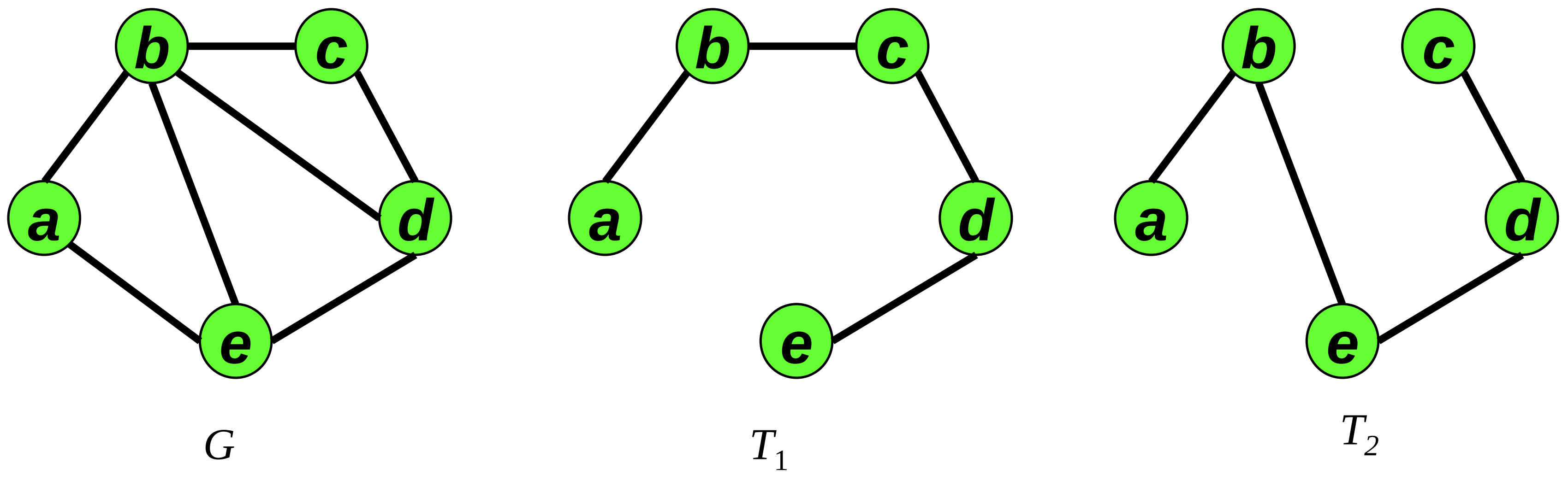
Phát biểu

- Cho đồ thị vô hướng, liên thông có trọng số G(V,E,C)

Yêu cầu

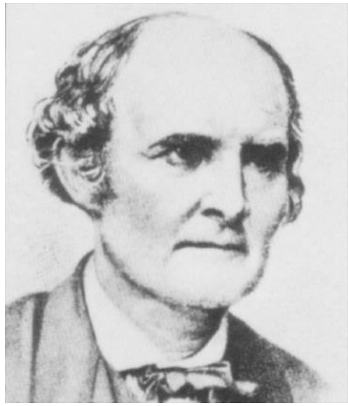
- Tìm cây khung của đồ thị thỏa mãn điều kiện:
- Tổng trọng số của nó $\rightarrow$ MIN

❖ **Định nghĩa 2.** ***Giả sử G=(V,E) là đồ thị vô hướng liên thông. Cây T=(V,F) với $F \subset E$ được gọi là cây khung của đồ thị G.***



Đồ thị G và 2 cây khung T_1 và T_2 của nó

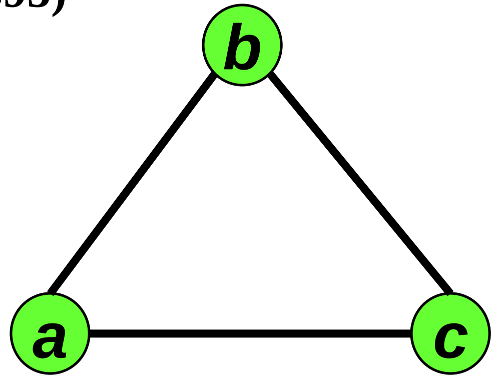
Số lượng cây khung của đồ thị



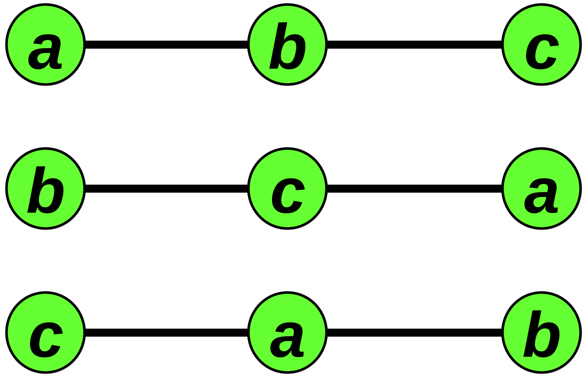
Arthur Cayley
(1821 – 1895)

- Định lý sau đây cho biết số lượng cây khung của đồ thị đầy đủ K_n:

- Định lý 2 (Cayley). *Số cây khung của đồ thị K_n là n^{n-2}* .



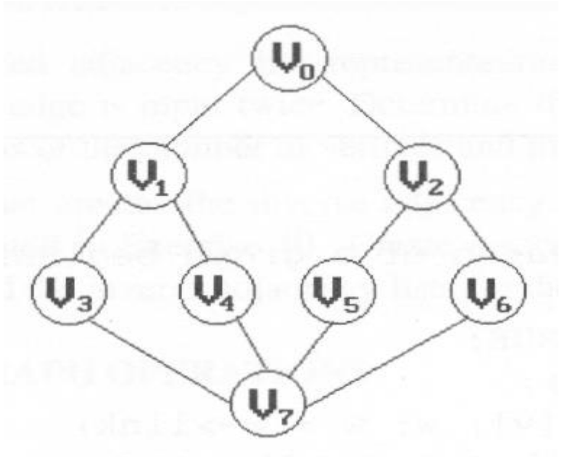
K_3



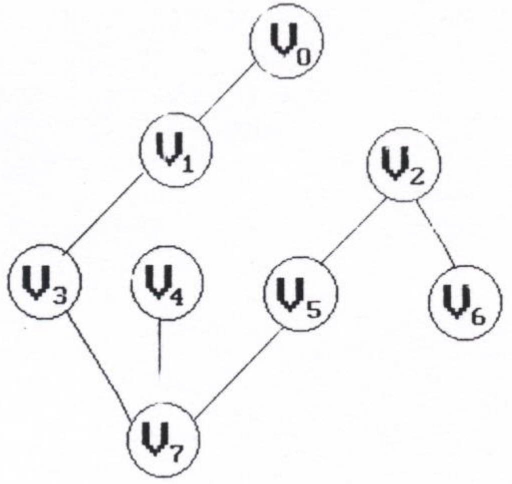
Ba cây khung của K_3

Finding a spanning tree

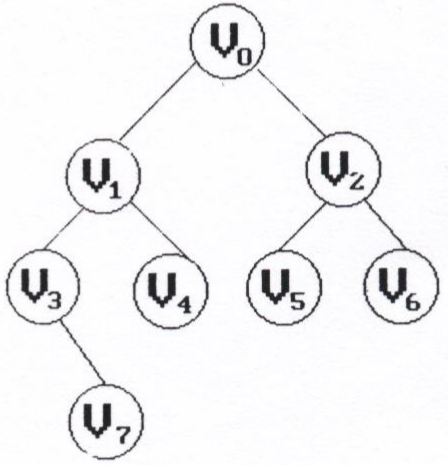
- When graph *G* is connected, a depth first or breadth first search starting at any vertex will visit all vertices in *G*
- We may use DFS or BFS to create a spanning tree
  - Depth first spanning tree when DFS is used
  - Breadth first spanning tree when BFS is used



(a)

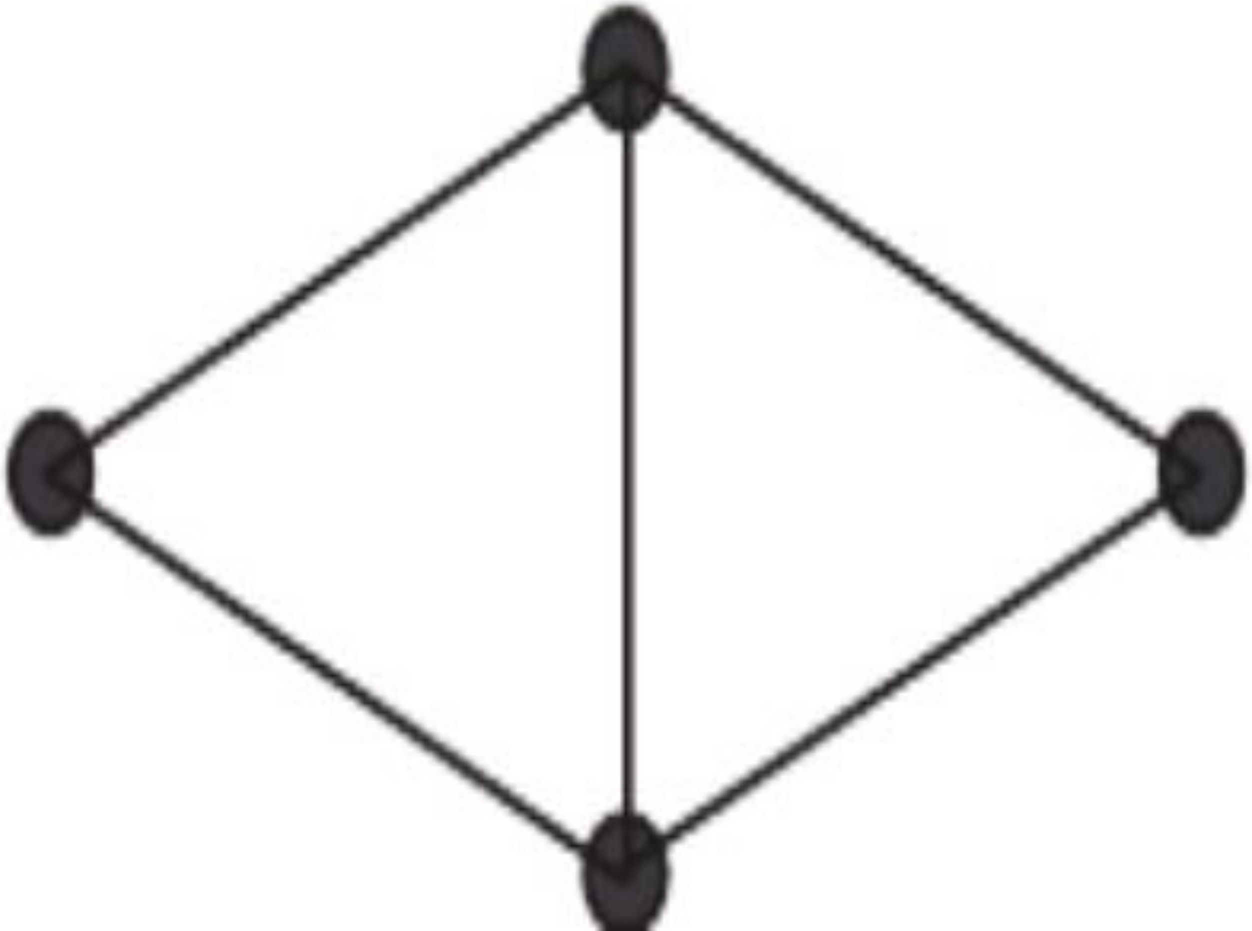


(a) dfs(0) spanning tree

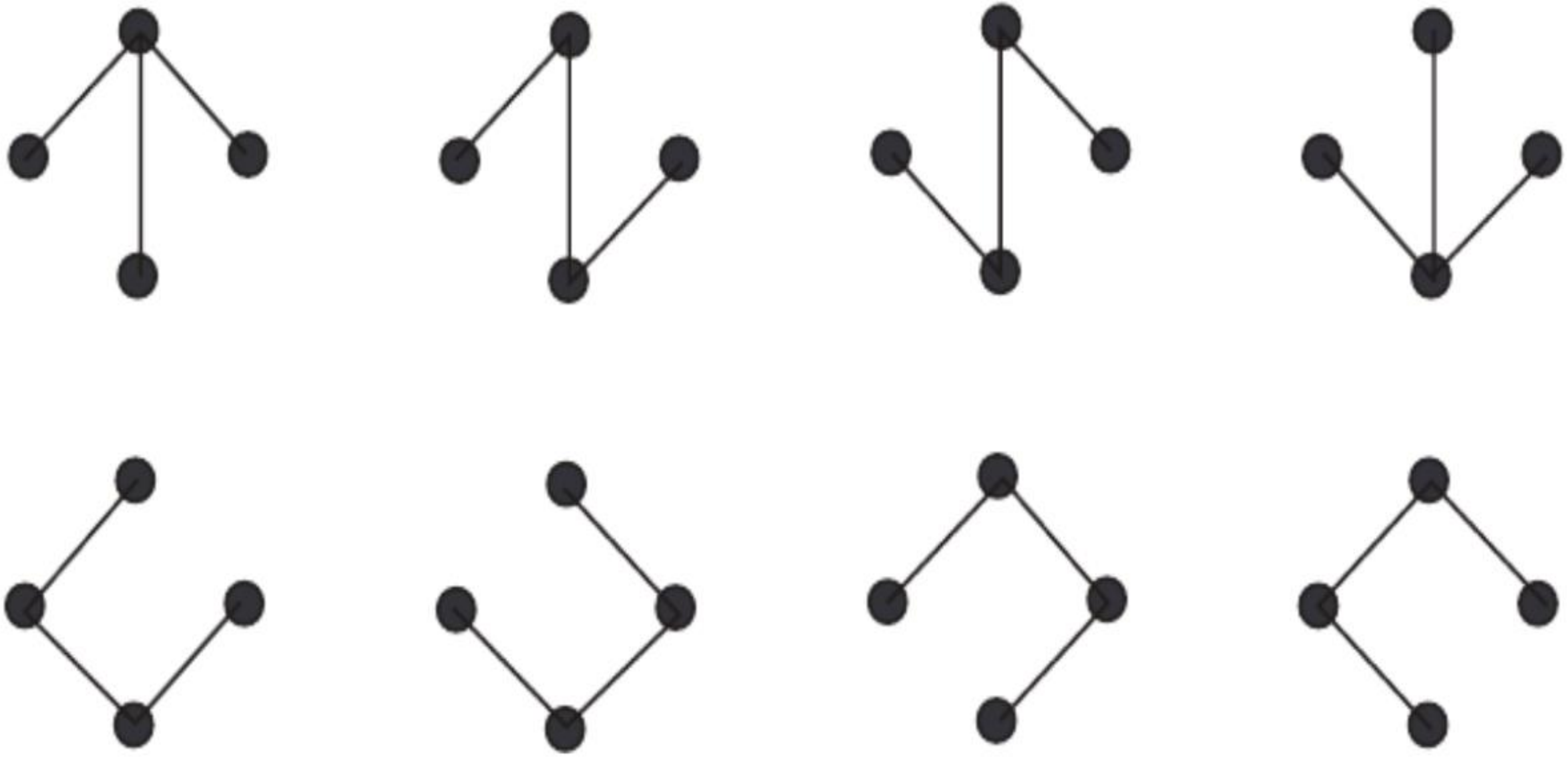


(b) bfs(0) spanning tree

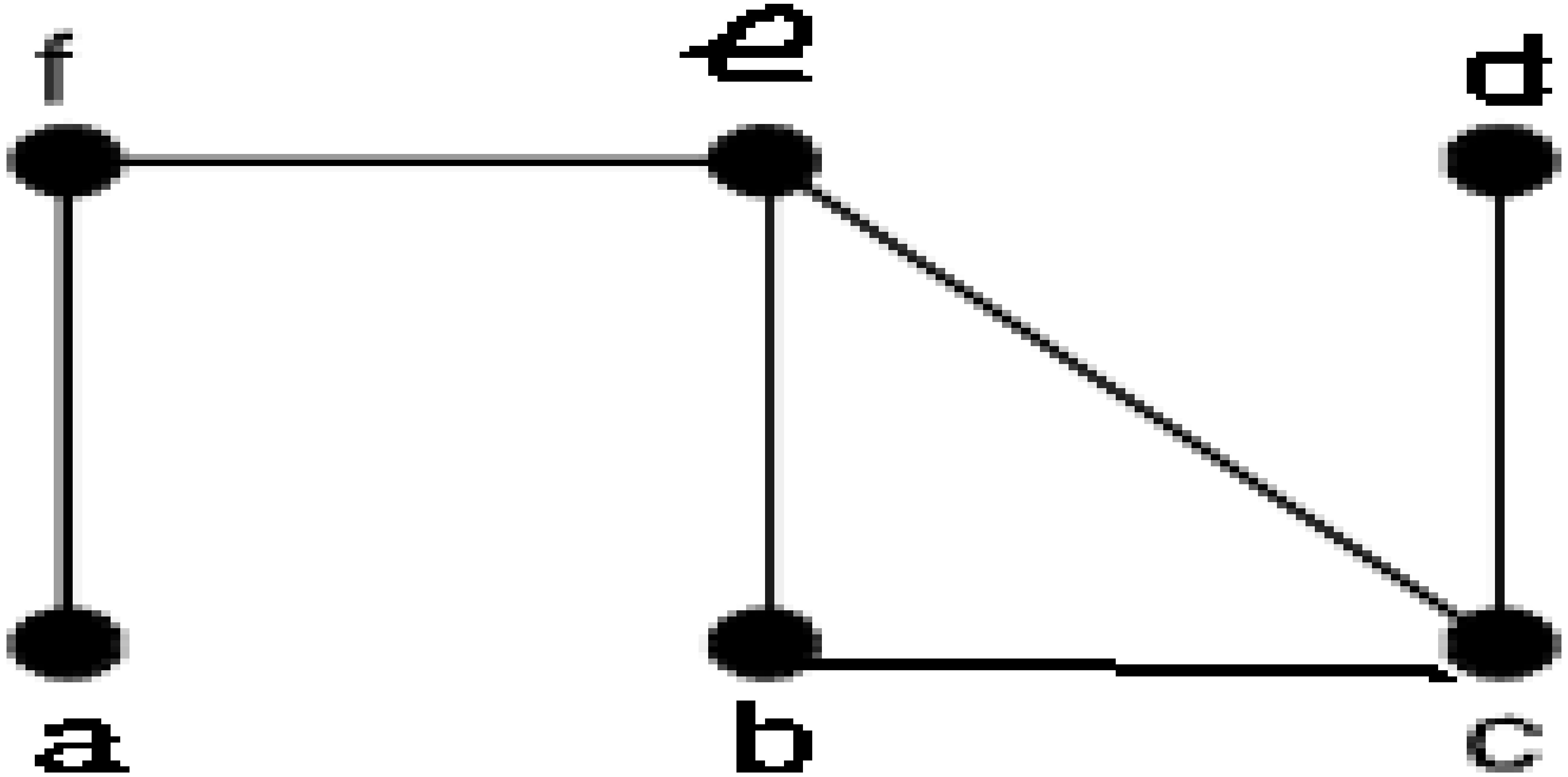
Find all spanning trees of the following graph.

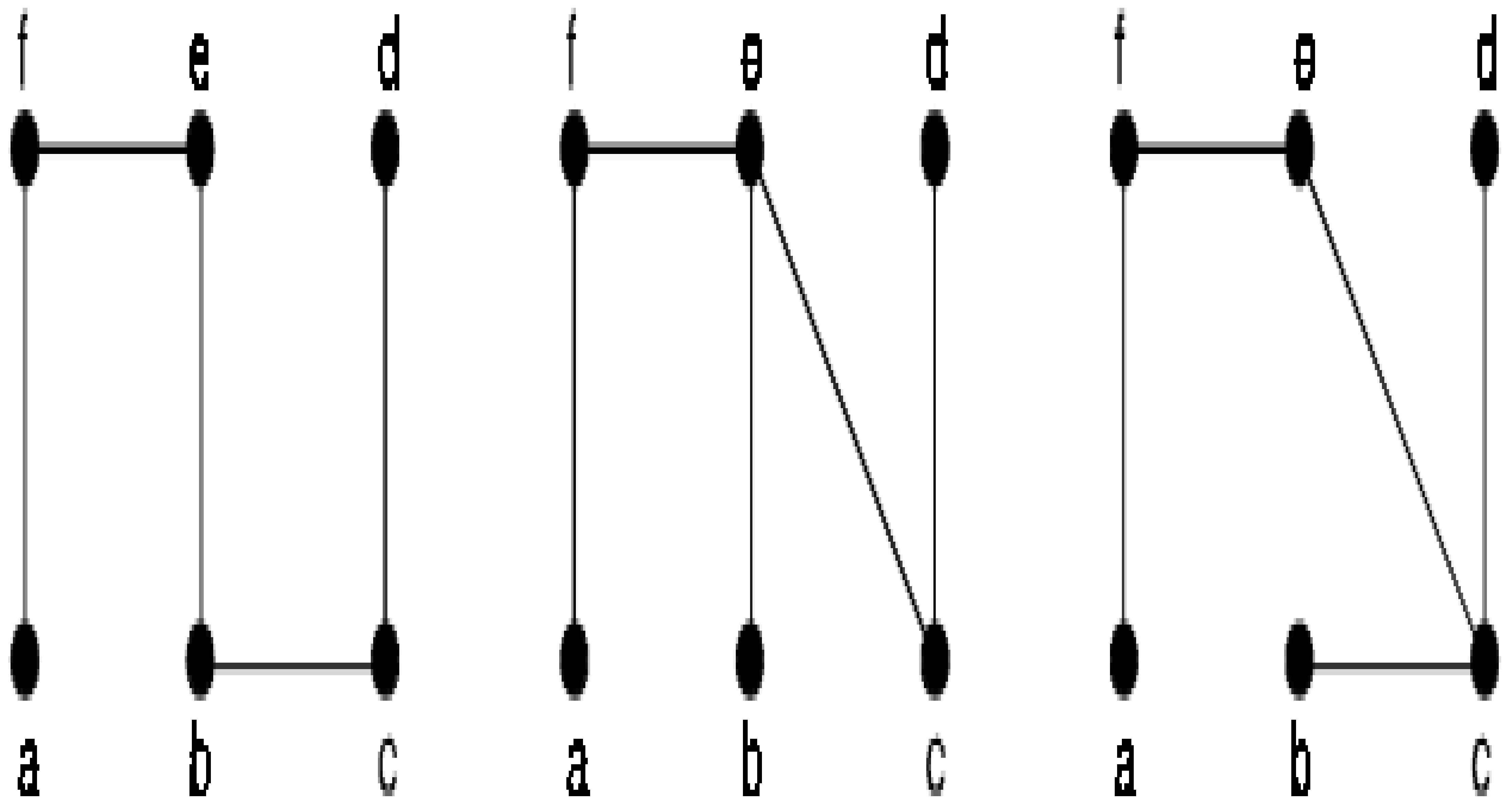


Solution. The graph has 8 spanning trees which are shown below:

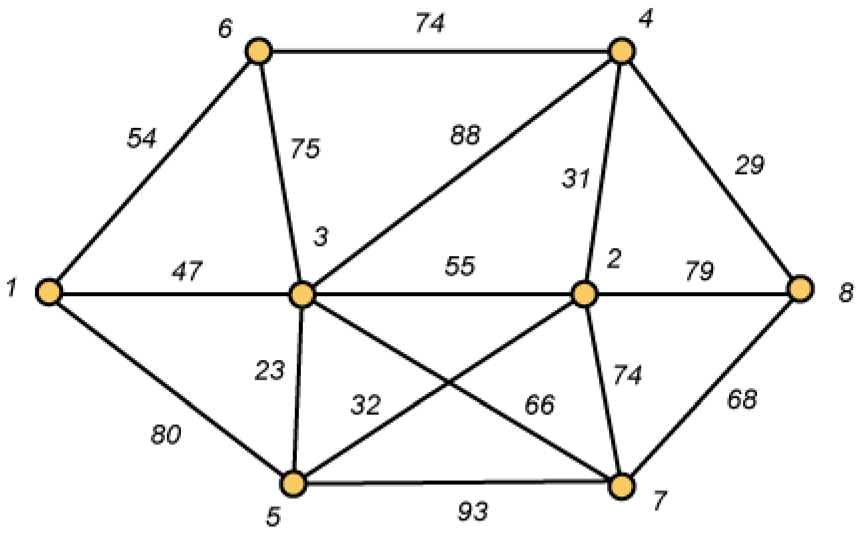


Find all spanning trees of the following graph.




f
e
d
a
b
c
f
e
d
a
b
c
f
e
d
a
b
c

Minimum Weight Spanning Trees



Problem: Given a connected graph with non-negative weights on the edges, find a spanning tree T for which the sum of the weights on the edges in T is as small as possible.

Why Not Try *All* Possibilities?

- Suppose the graph has n vertices. Then the number of possible spanning trees can be as large as n^{n-2}.
- When $n = 75$, this means that the number of spanning trees can be as large as

7576562804644601479086318651590413464814067\

83308840339247043281018024279971356804708193\

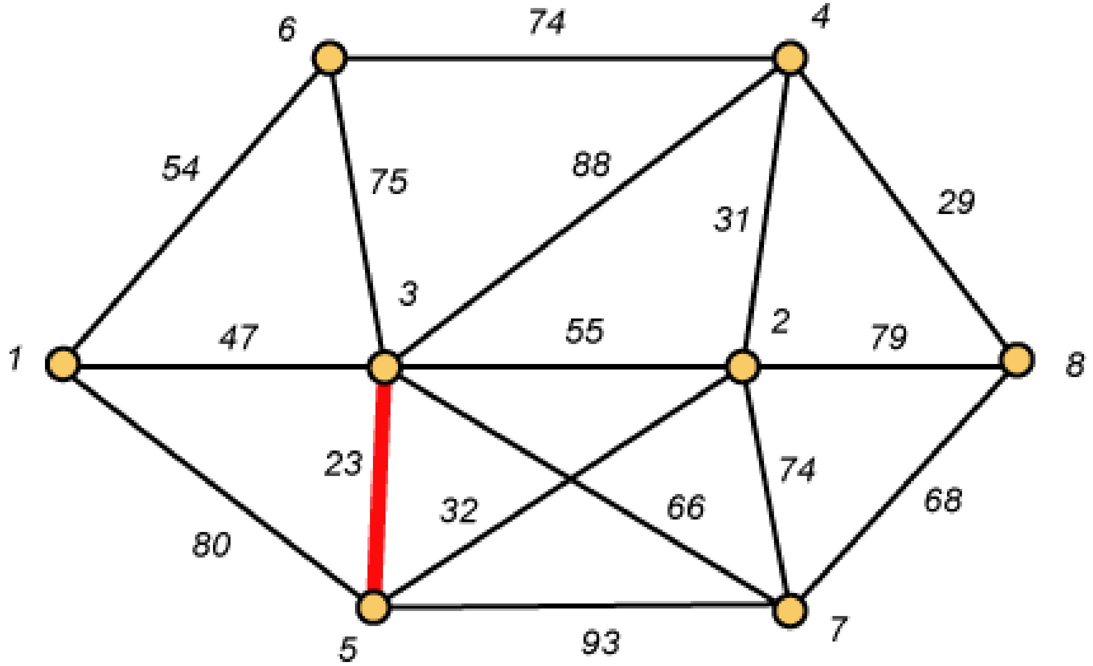
5219466686248779296875

Kruskal’s Algorithm (Avoid Cycles)

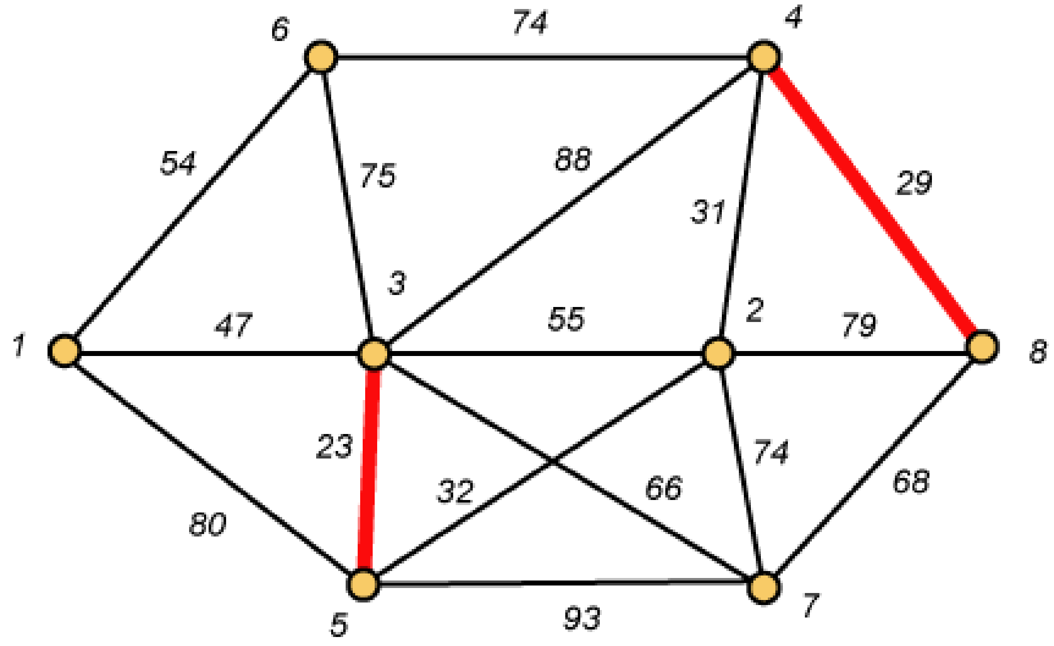
- Sort the edges by weight

- Build a spanning forest (that eventually becomes a tree) by adding the edge of minimum weight which when added to those already chosen does not form a cycle.

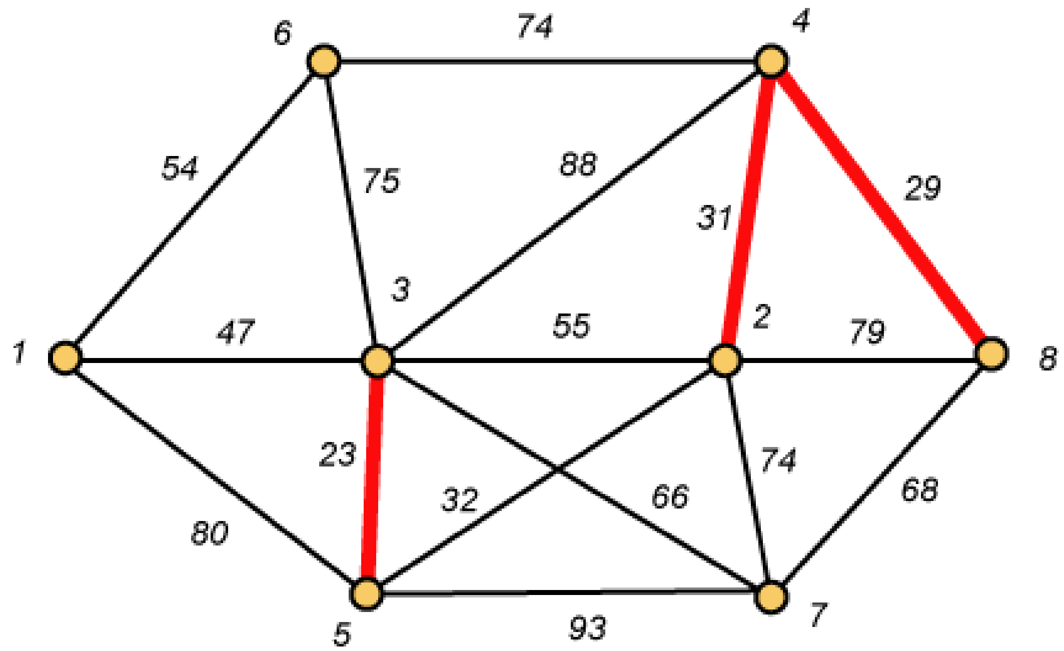
Kruskal – Step 1



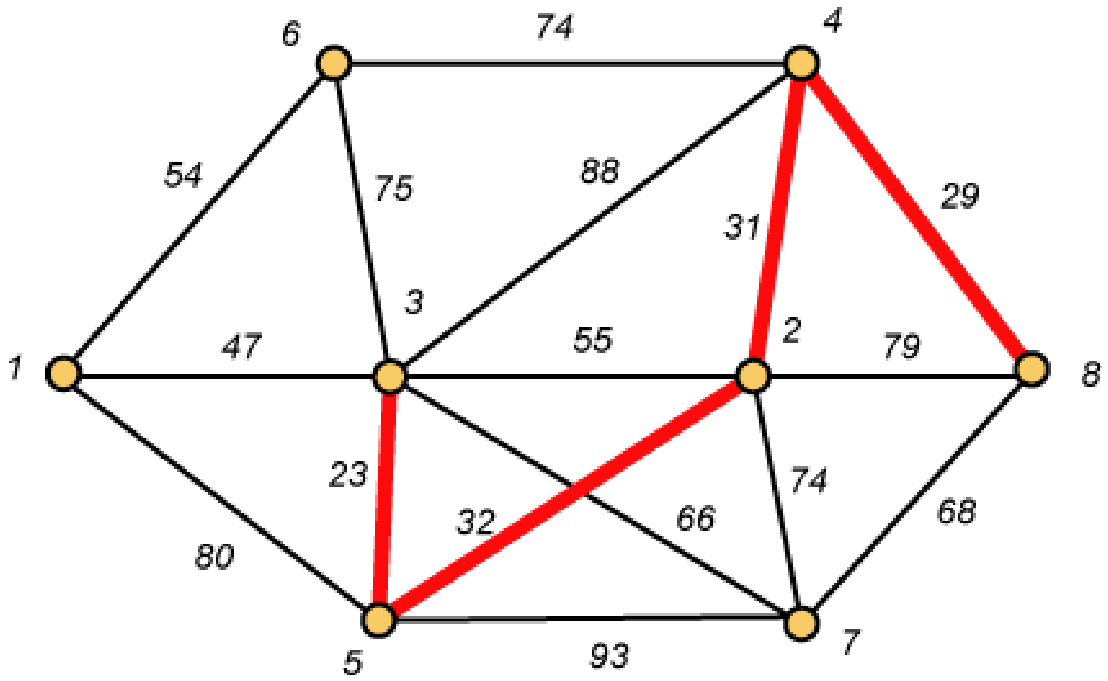
Kruskal – Step 2



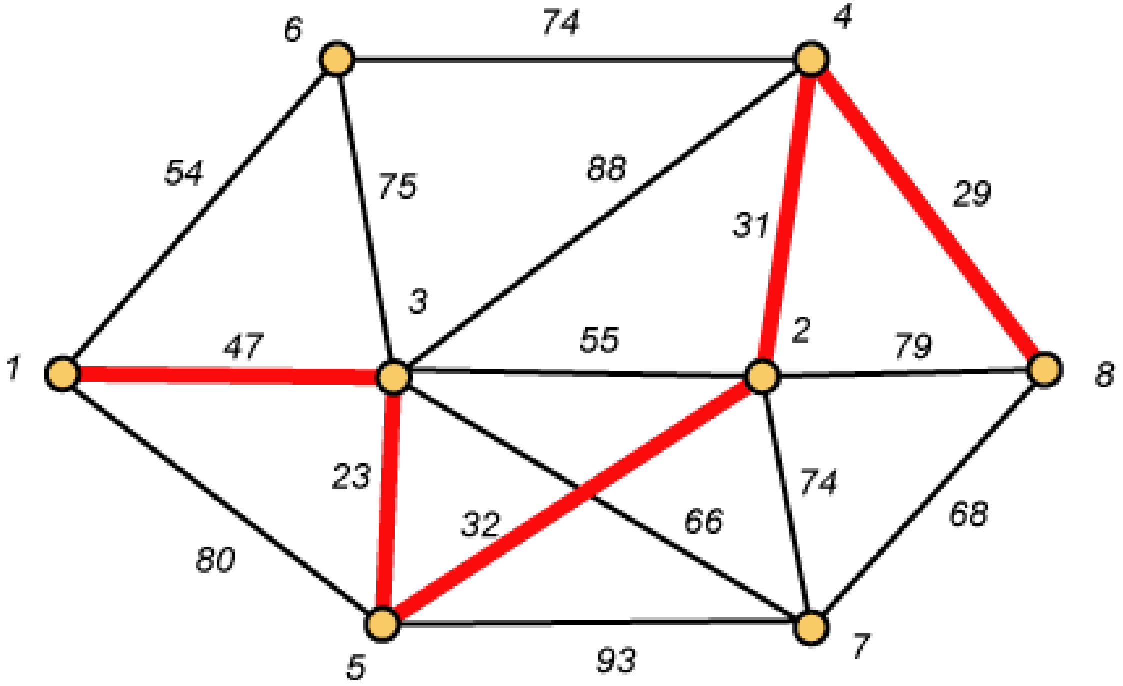
Kruskal – Step 3



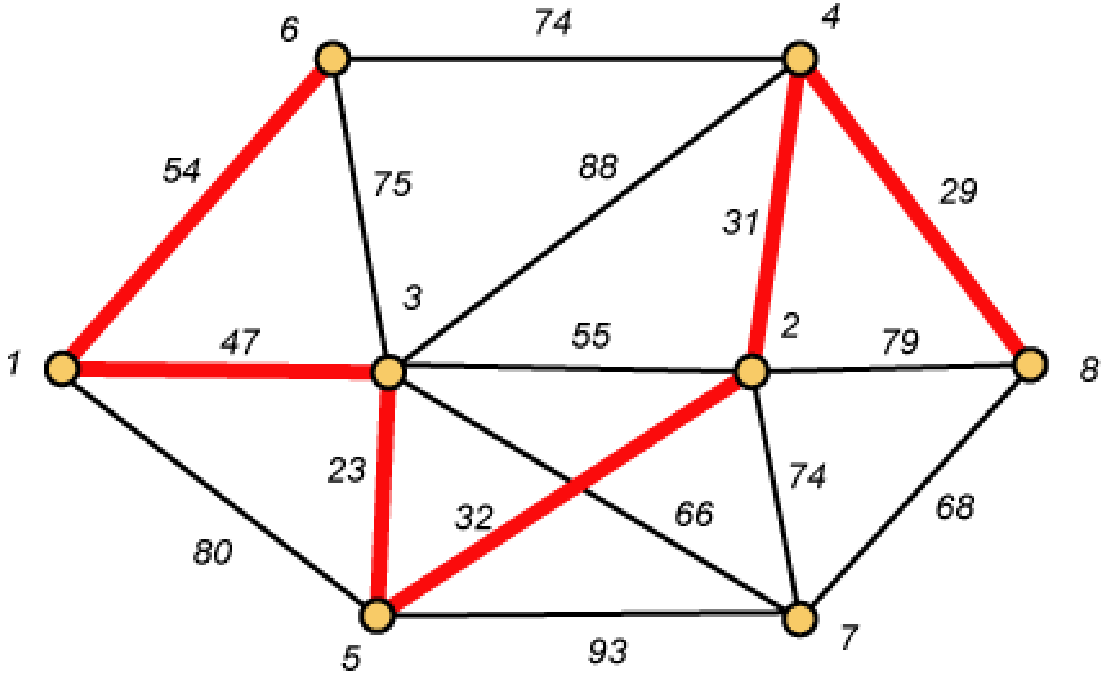
Kruskal – Step 4



Kruskal – Step 5



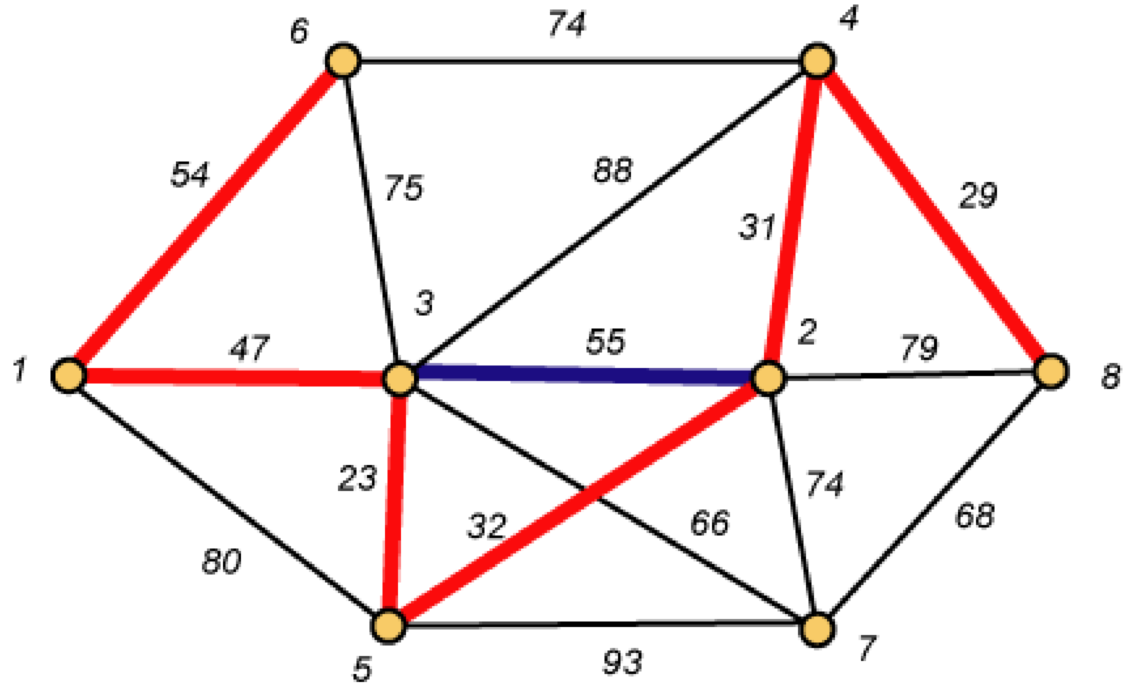
Kruskal – Step 6



Why Avoiding Cycles Matters

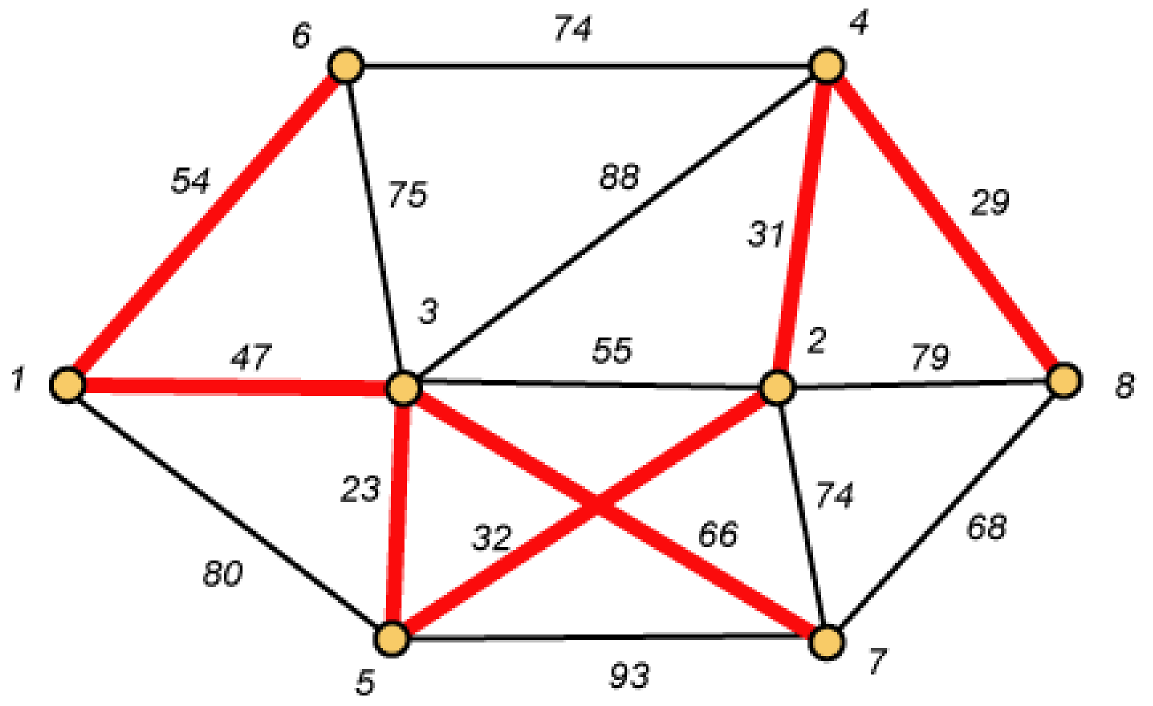
Up to this point, we have simply taken the edges in order of their weight. But now we will have to reject an edge since it forms a cycle when added to those already chosen.

Forms a Cycle



So we cannot take the blue edge having weight 55.

Kruskal – Step 7 *DONE!!*

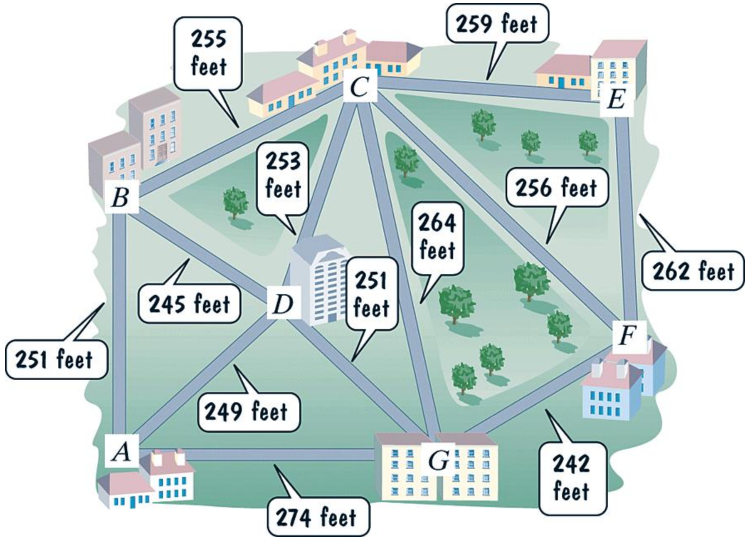


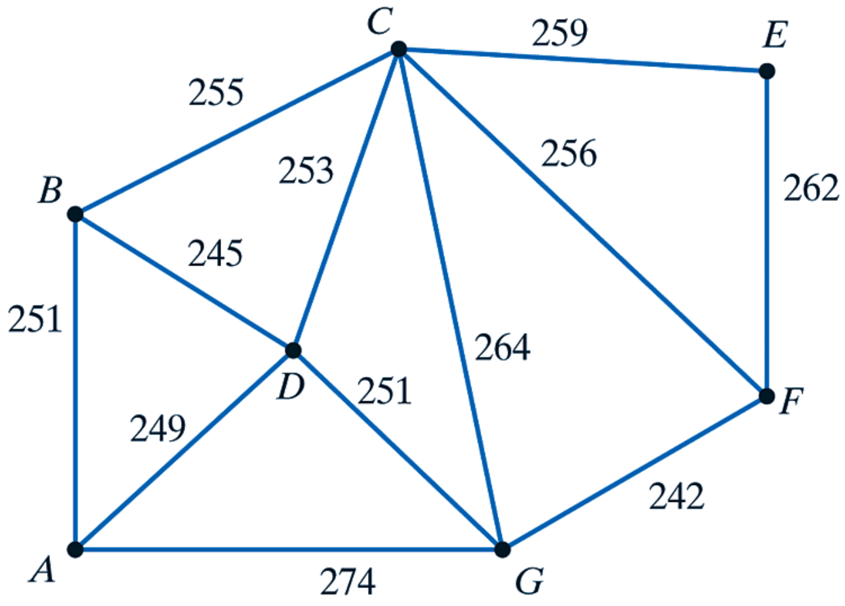
Weight (T) = 23 + 29 + 31 + 32 + 47 + 54 + 66 = **282**

Ví dụ 1 (tự làm): Using Kruskal's Algorithm

- Seven buildings on a college are connected by the sidewalks shown in the figure. (next slide)
- The weighted graph represents buildings as vertices, sidewalks as edges, and sidewalk lengths as weights.
- A heavy snow has fallen and the sidewalks need to be cleared quickly. Campus decides to clear as little as possible and still ensure that students walking from building to building will be able to do so along cleared paths.
- Determine the shortest series of sidewalks to clear. What is the total length of the sidewalks that need to be cleared?

Ví dụ 1 (tự làm): Using Kruskal’s Algorithm (continued)

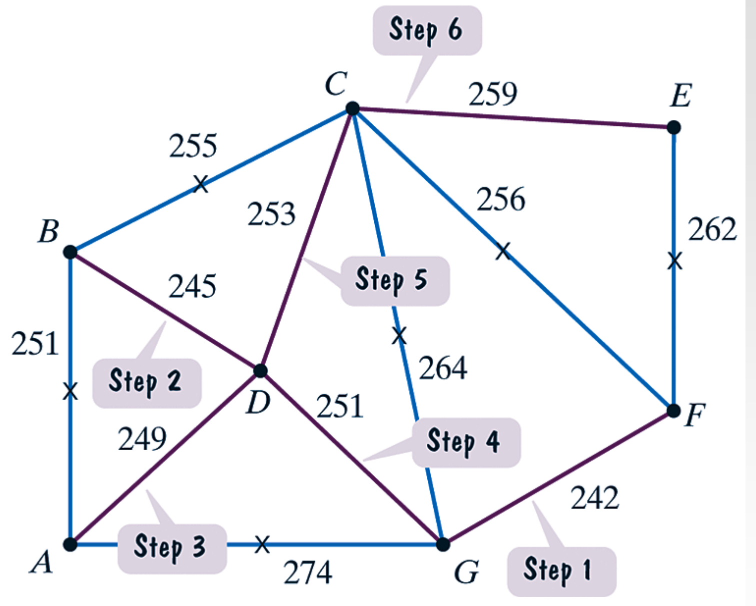




- Using Kruskal’s Algorithm, we complete minimizing the spanning tree in a series of steps given below. Refer to the next graph coinciding with the steps.

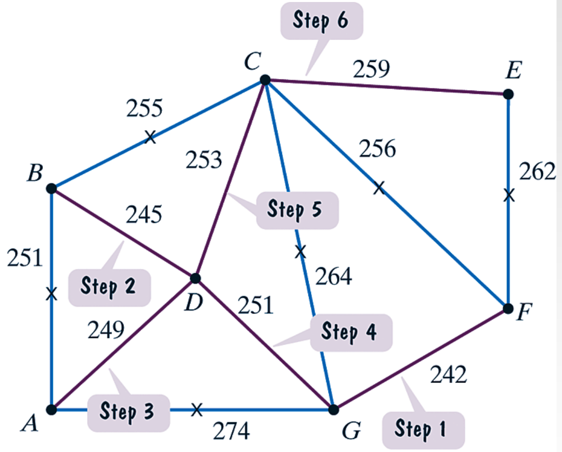
Ví dụ 1 (tự làm): Using Kruskal’s Algorithm (continued)

- Step 1. Find the edge with the smallest weight. Select edge *GF* by marking it in red.
- Step 2. Find the next-smallest edge
- in the graph. Select *BD* by marking it in red.
- Step 3. Find the next-smallest edge
- in the graph. Select *AD* by marking it in red.
- Step 4. Find the next-smallest edge
- in the graph that does not create a circuit. Select *DG*, since it does not create a circuit, by marking it in red.



Ví dụ 1 (tự làm): Using Kruskal’s Algorithm (continued)

- Step 5. Find the next-smallest edge in the
- graph that does not create a circuit.
- Select *CD*, since it does not create
- a circuit, by marking it in red.
- Step 6. Find the next-smallest edge
- in the graph that does not create
- a circuit. Select *CE*, since it does
- not create a circuit, by marking it in red.

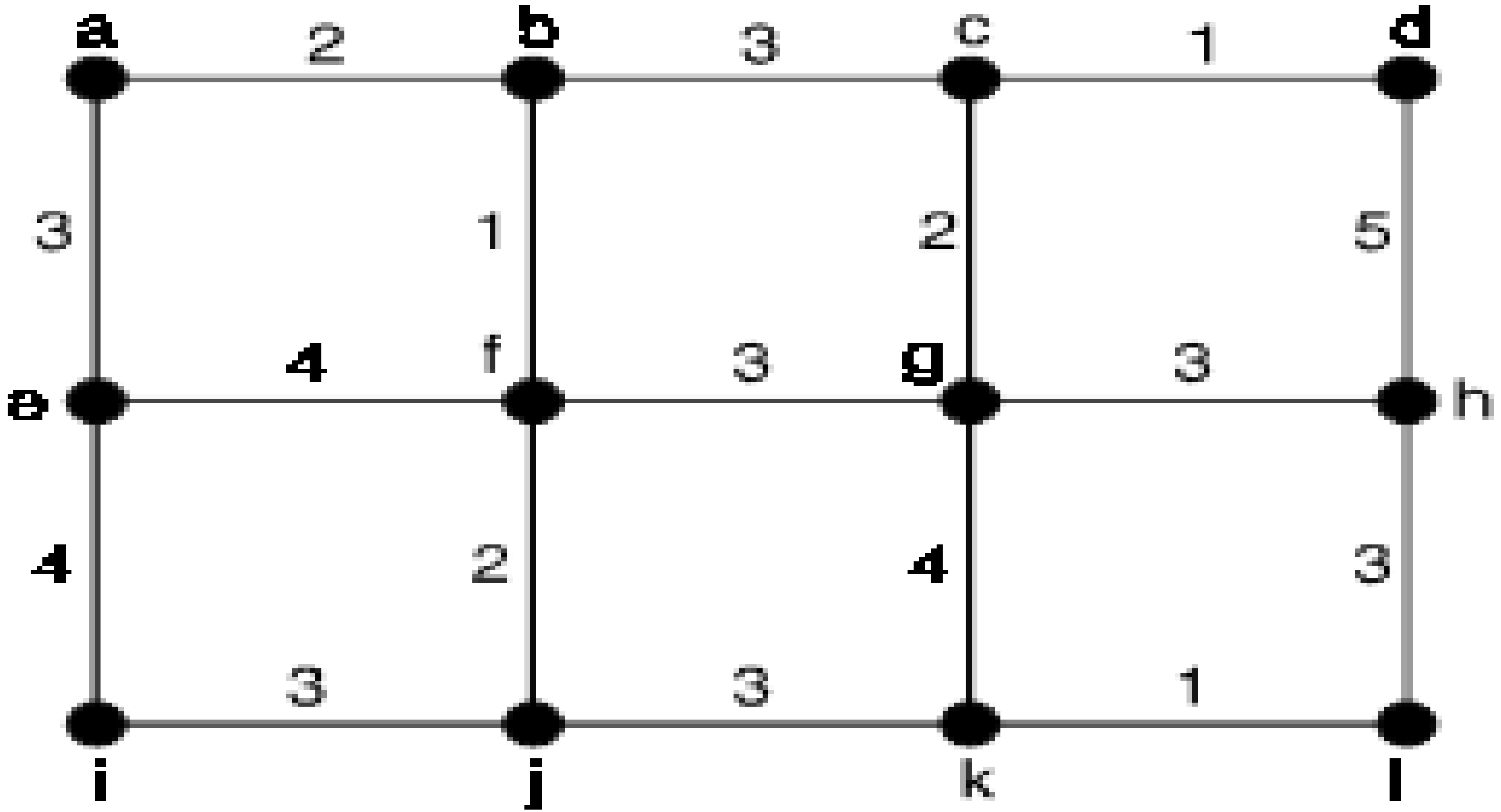


- The minimum spanning tree is completed. The red subgraph contains all seven vertices, six edges, and no circuits.

Ví dụ 1 (tự làm): Using Kruskal's Algorithm (continued)

- The total length of the sidewalks that need to be cleared is 242 + 245 + 249 + 251 + 253 + 259, or 1499 feet.

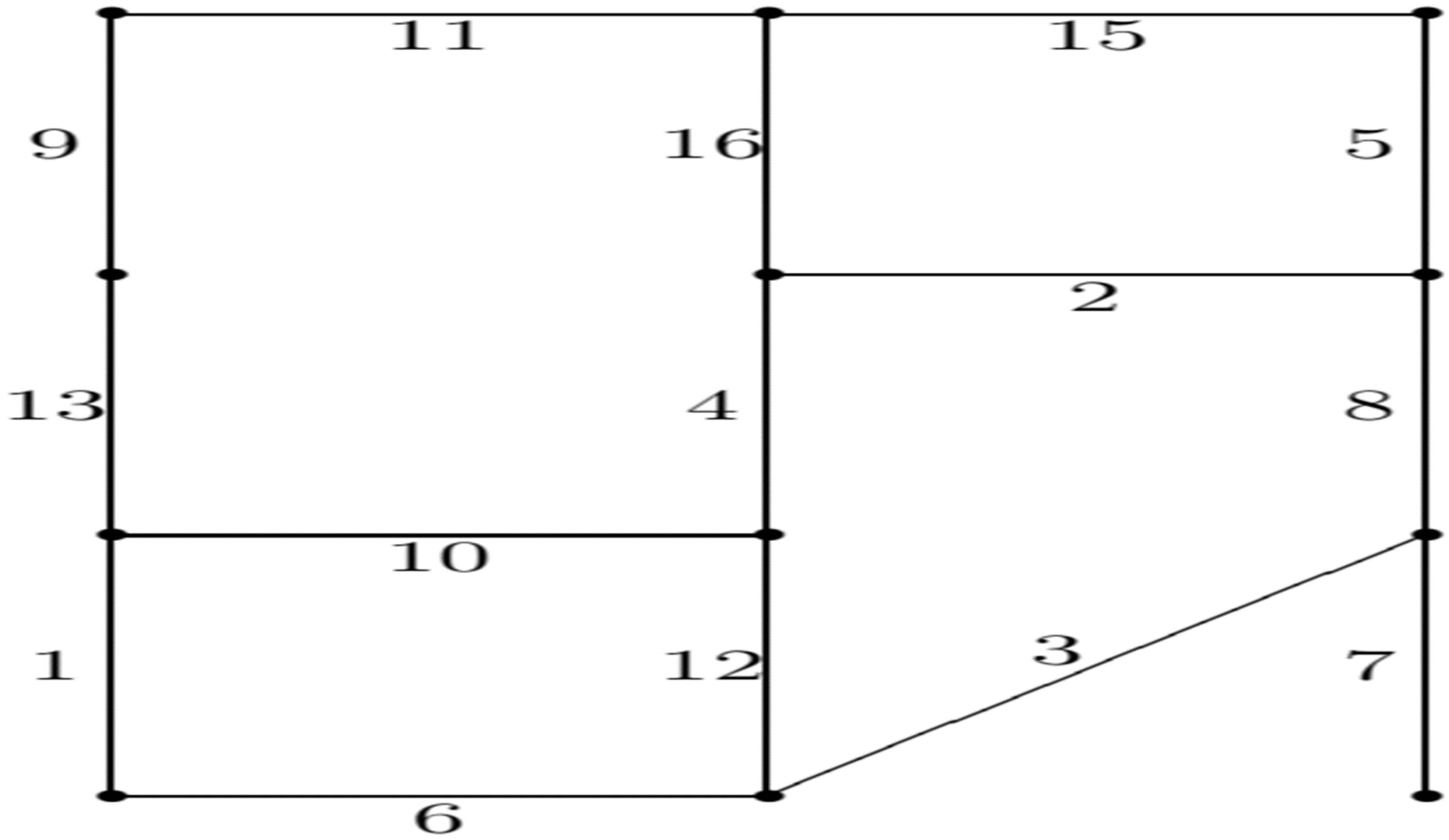
Ví dụ 2 (tự làm): Using Kruskal's Algorithm



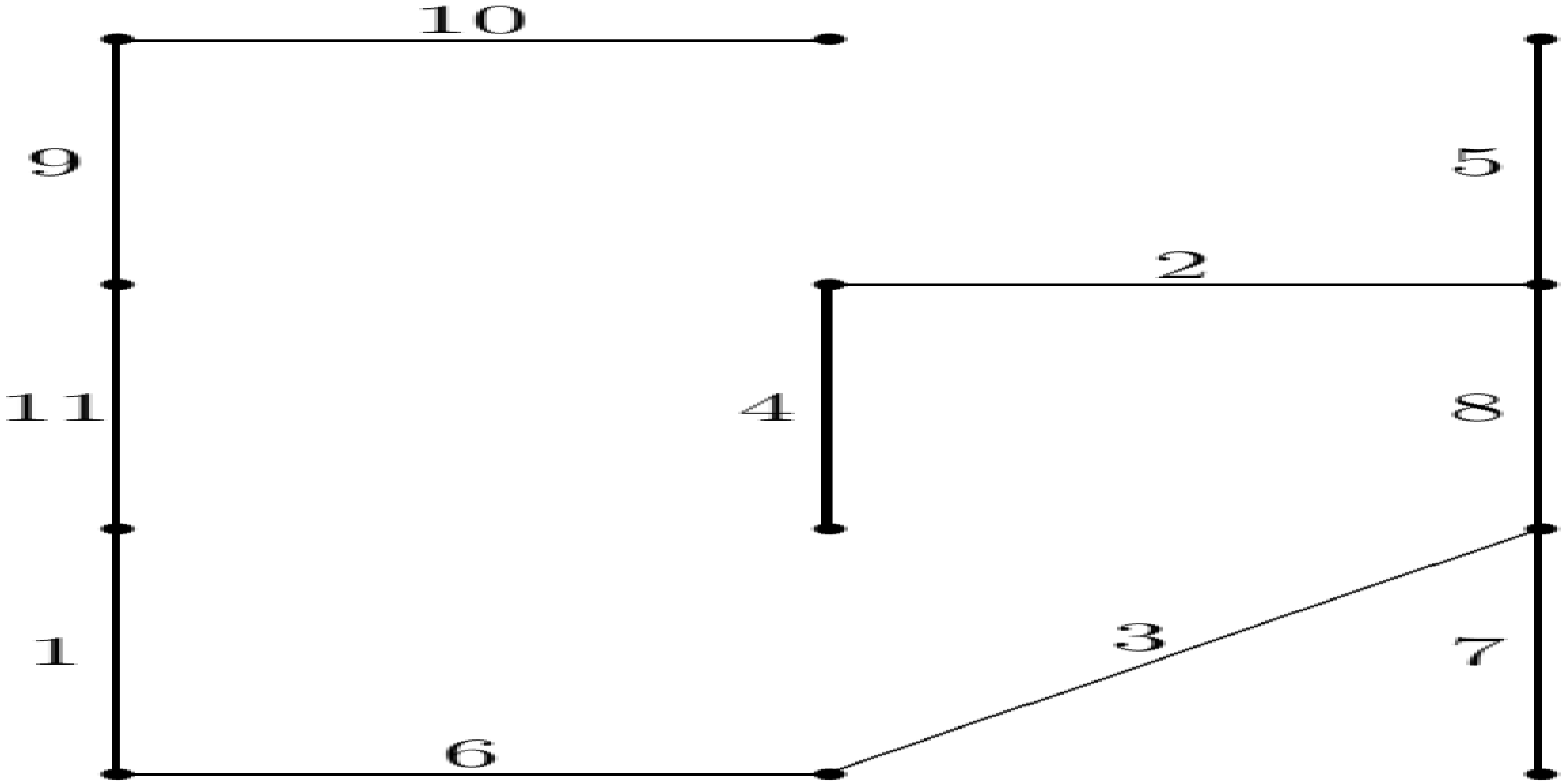
Ví dụ 2 (tự làm): Using Kruskal's Algorithm

Choice	Edge	Weight
1	$\{c, d\}$	1
2	$\{k, l\}$	1
3	$\{b, f\}$	1
4	$\{c, g\}$	2
5	$\{a, b\}$	2
6	$\{f, j\}$	2
7	$\{b, c\}$	3
8	$\{j, k\}$	3
9	$\{g, h\}$	3
10	$\{i, j\}$	3
11	$\{a, e\}$	3
	Total:	24

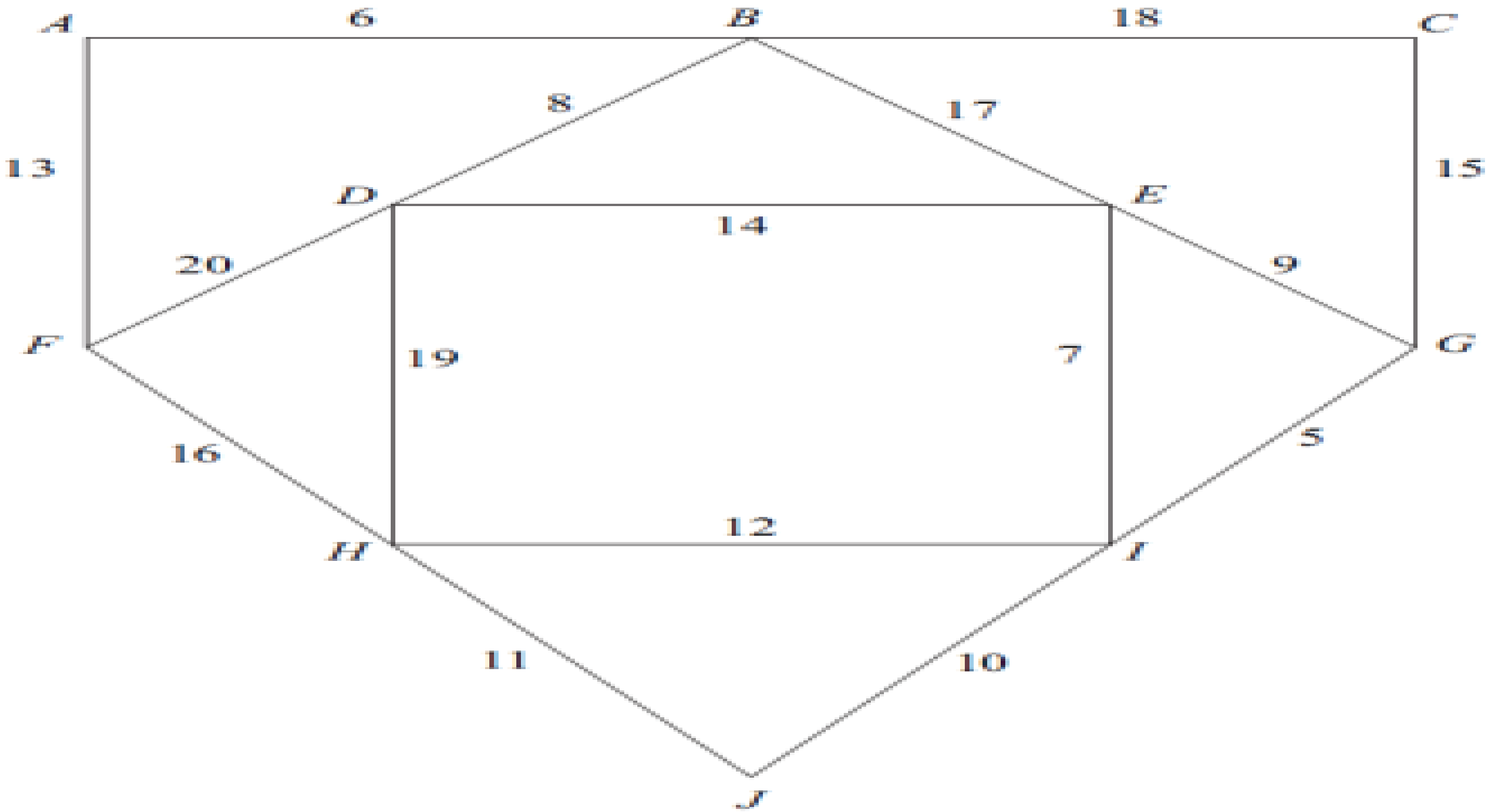
Ví dụ 3 (tự làm): Using Kruskal’s Algorithm



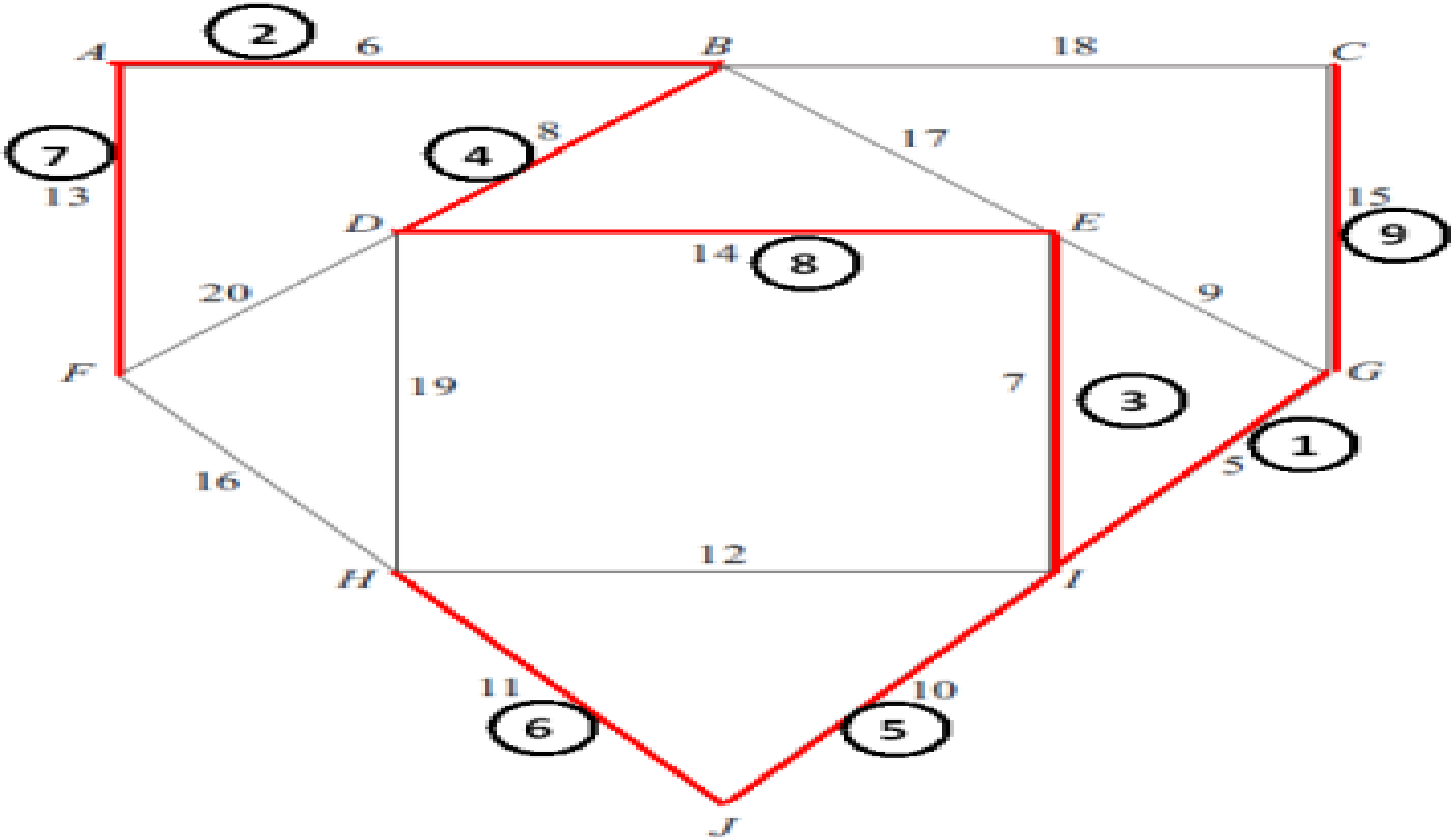
Ví dụ 3 (tự làm): Using Kruskal’s Algorithm



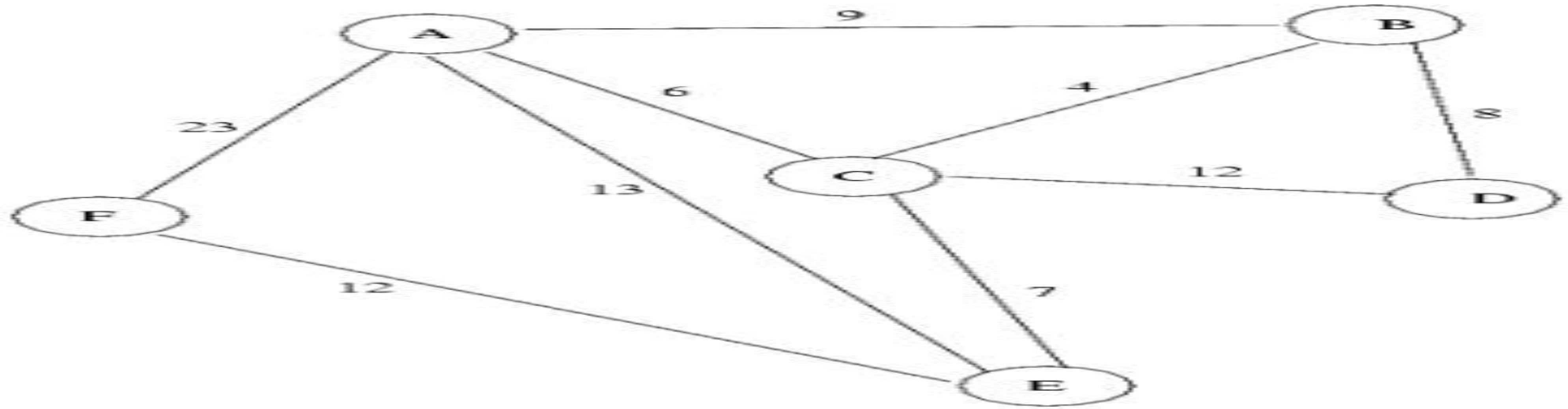
Ví dụ 4 (tự làm): Using Kruskal’s Algorithm



Ví dụ 4 (tự làm): Using Kruskal's Algorithm



Ví dụ 5 (tự làm): Using Kruskal’s Algorithm



- Suppose that it is desired to install a new fiber optic cable network between the six cities (A, B, C, D, E, and F) shown above for the least total cost. Also, suppose that the fiber optic cable can only be installed along the roadways shown above. The weighted graph above shows the cost (in millions of dollars) of installing the fiber optic cable along each roadway. We want to find the minimum spanning tree for this graph using Kruskal’s Algorithm.

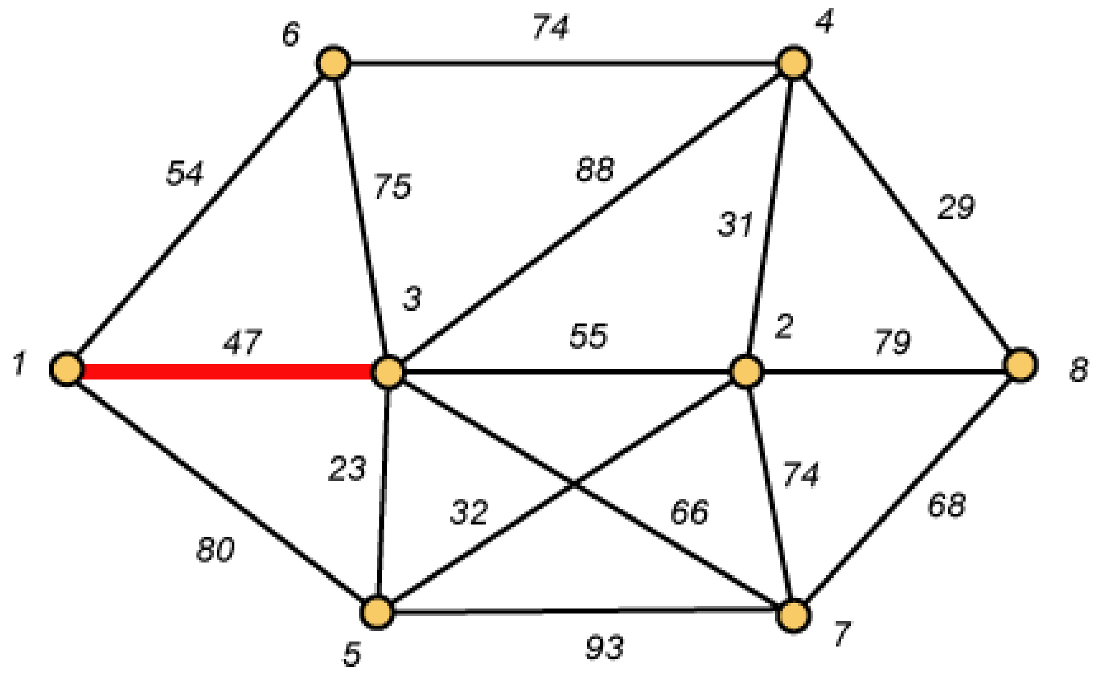
Prim's Algorithm (Build Tree)

Steps of Prim's Algorithm

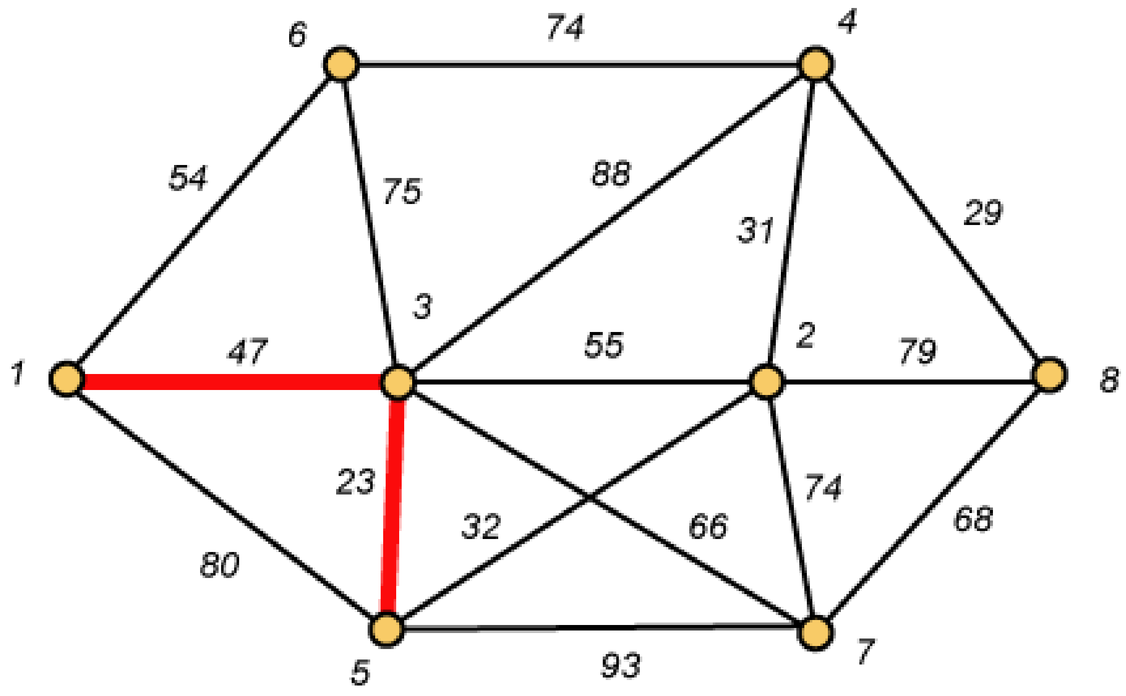
1. Start at any node
2. Join this node to the nearest node.

   If 2 or more nodes are equally far away, choose one at random
3. Join any of the connected nodes to the nearest unconnected node
4. Continue joining connected nodes to unconnected nodes until all nodes are connected

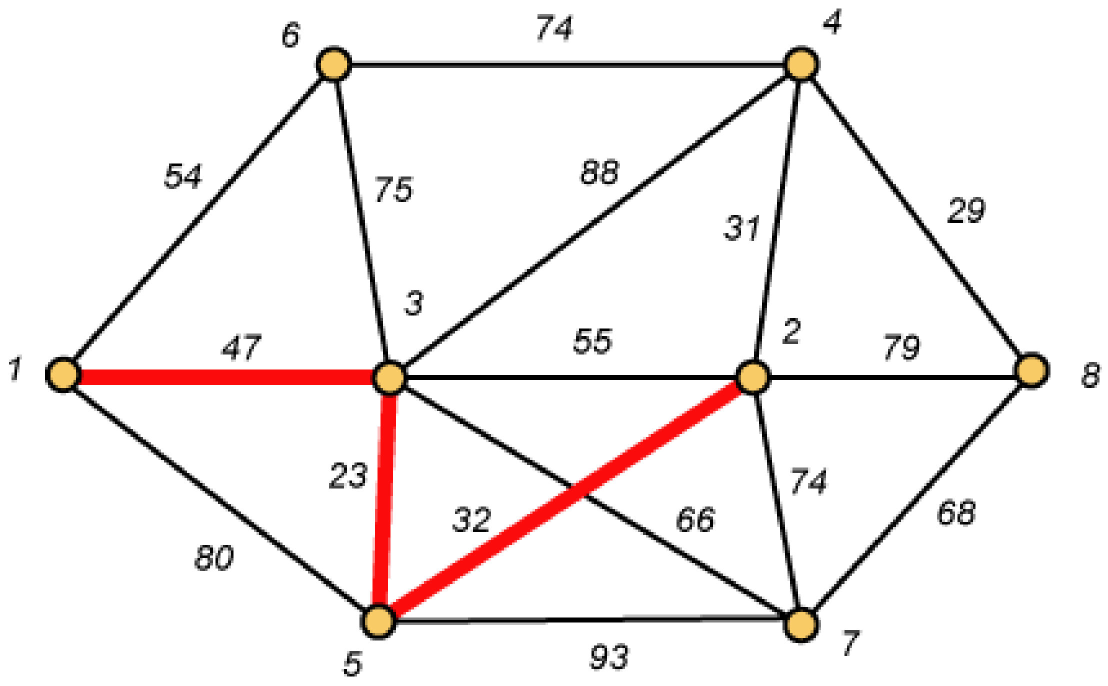
Prim – Step 1



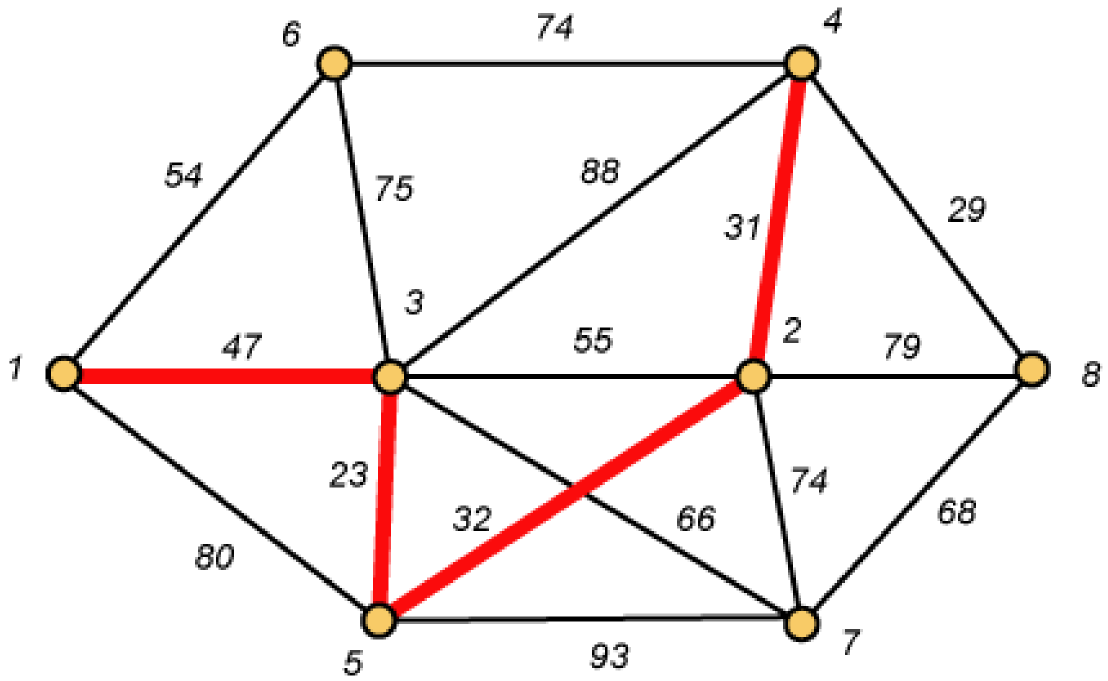
Prim – Step 2



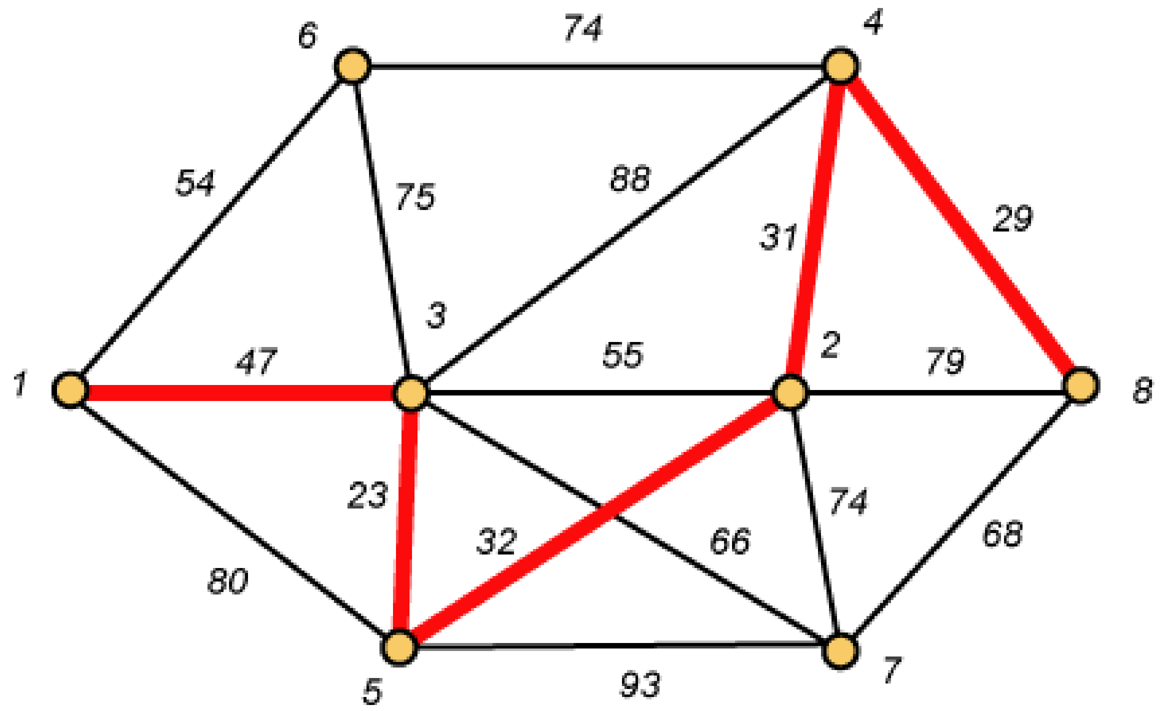
Prim – Step 3



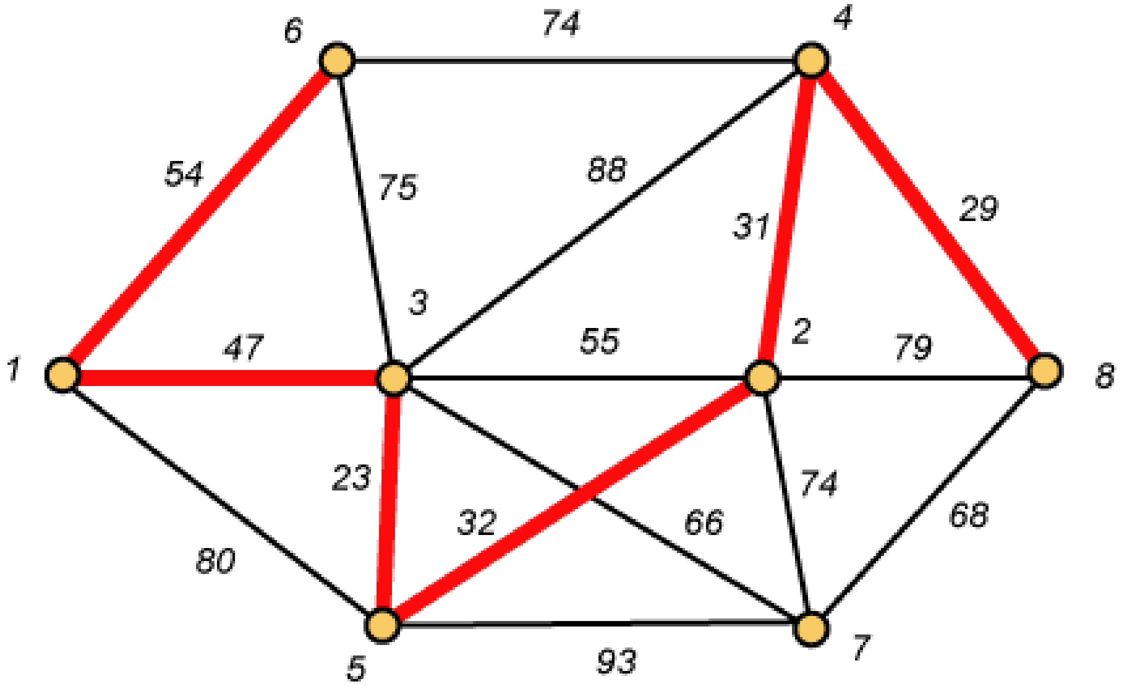
Prim – Step 4



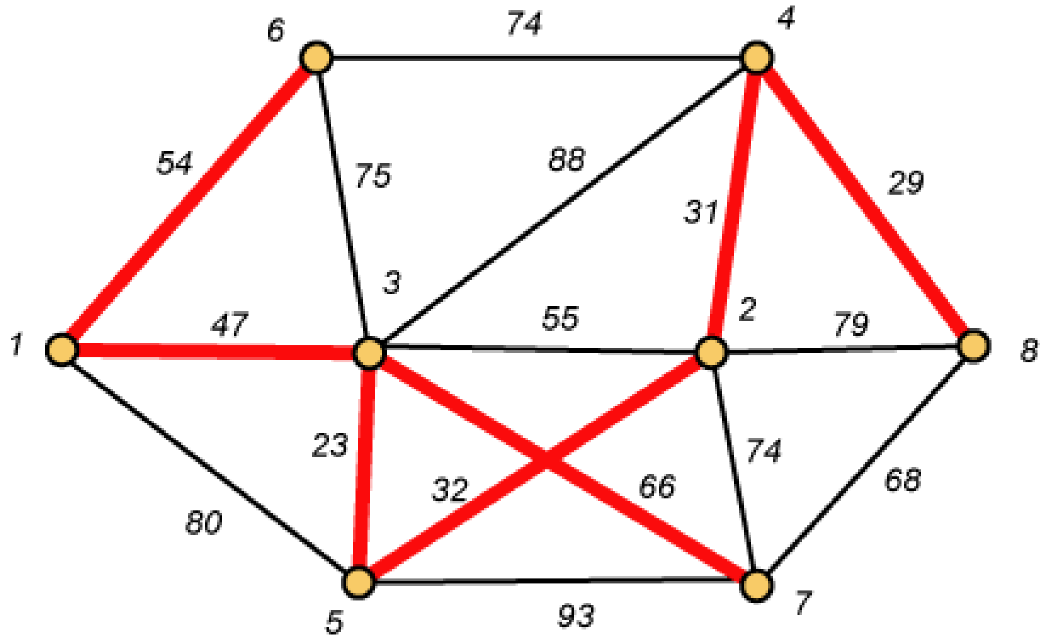
Prim – Step 5



Prim – Step 6

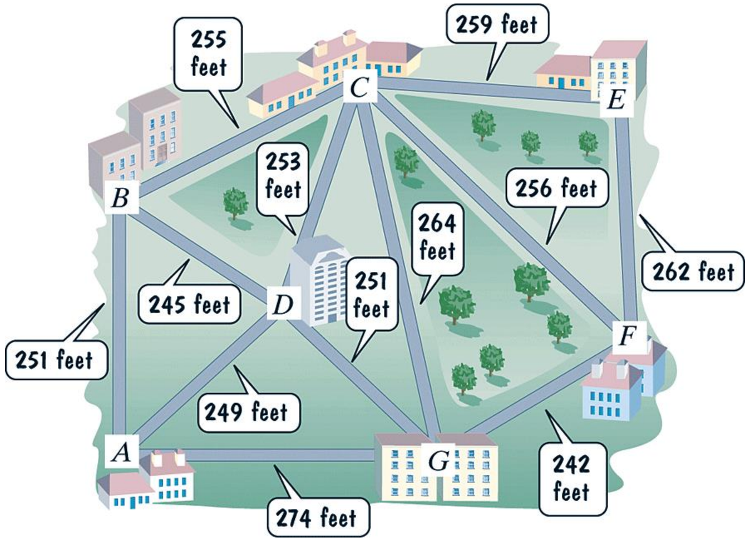


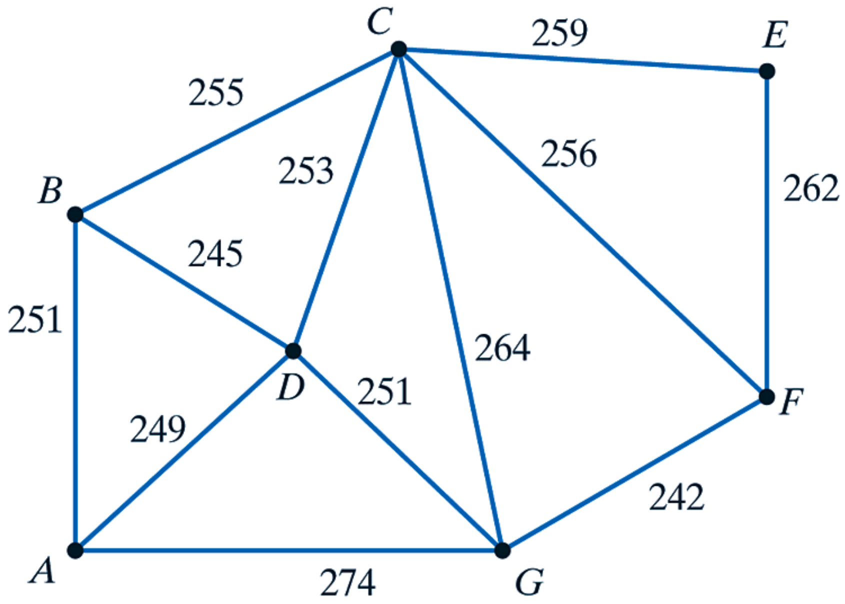
Prim – Step 7 *Done!!*



Weight (T) = 23 + 29 + 31 + 32 + 47 + 54 + 66 = **282**

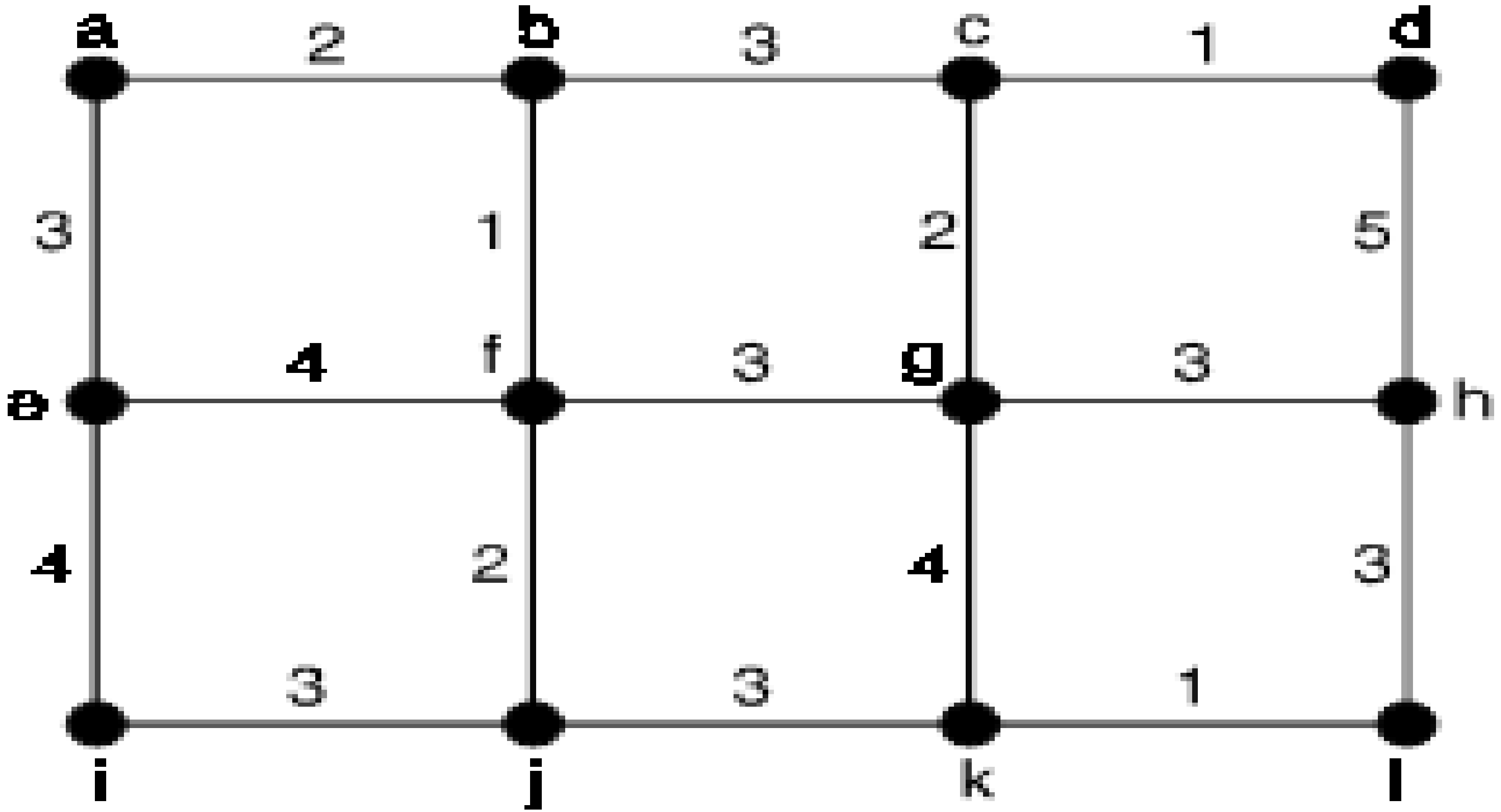
Ví dụ 1 (tự làm): Using Prim’s Algorithm (continued)



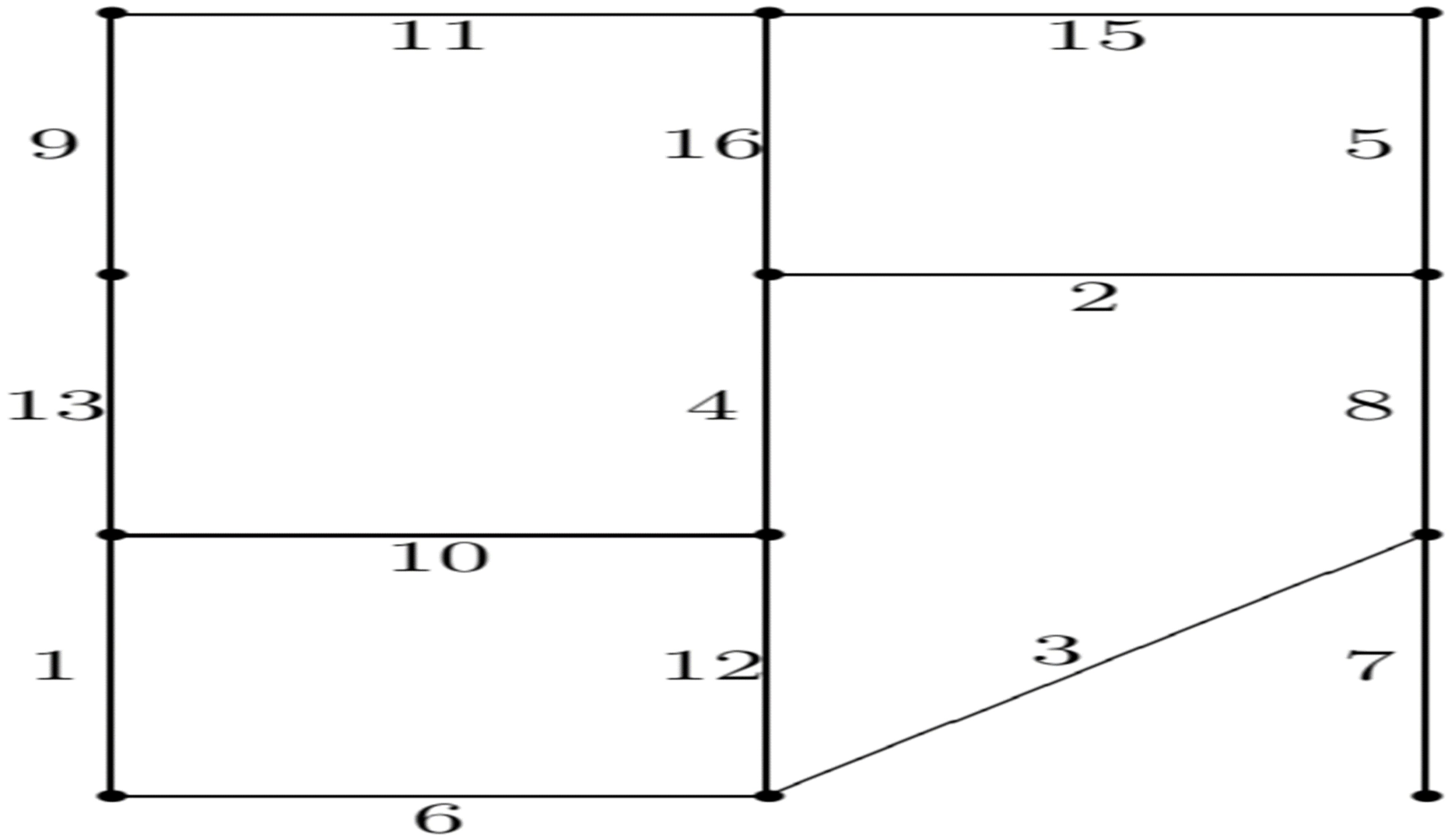


- Using Prim’s Algorithm, we complete minimizing the spanning tree in a series of steps given below. Refer to the next graph coinciding with the steps.

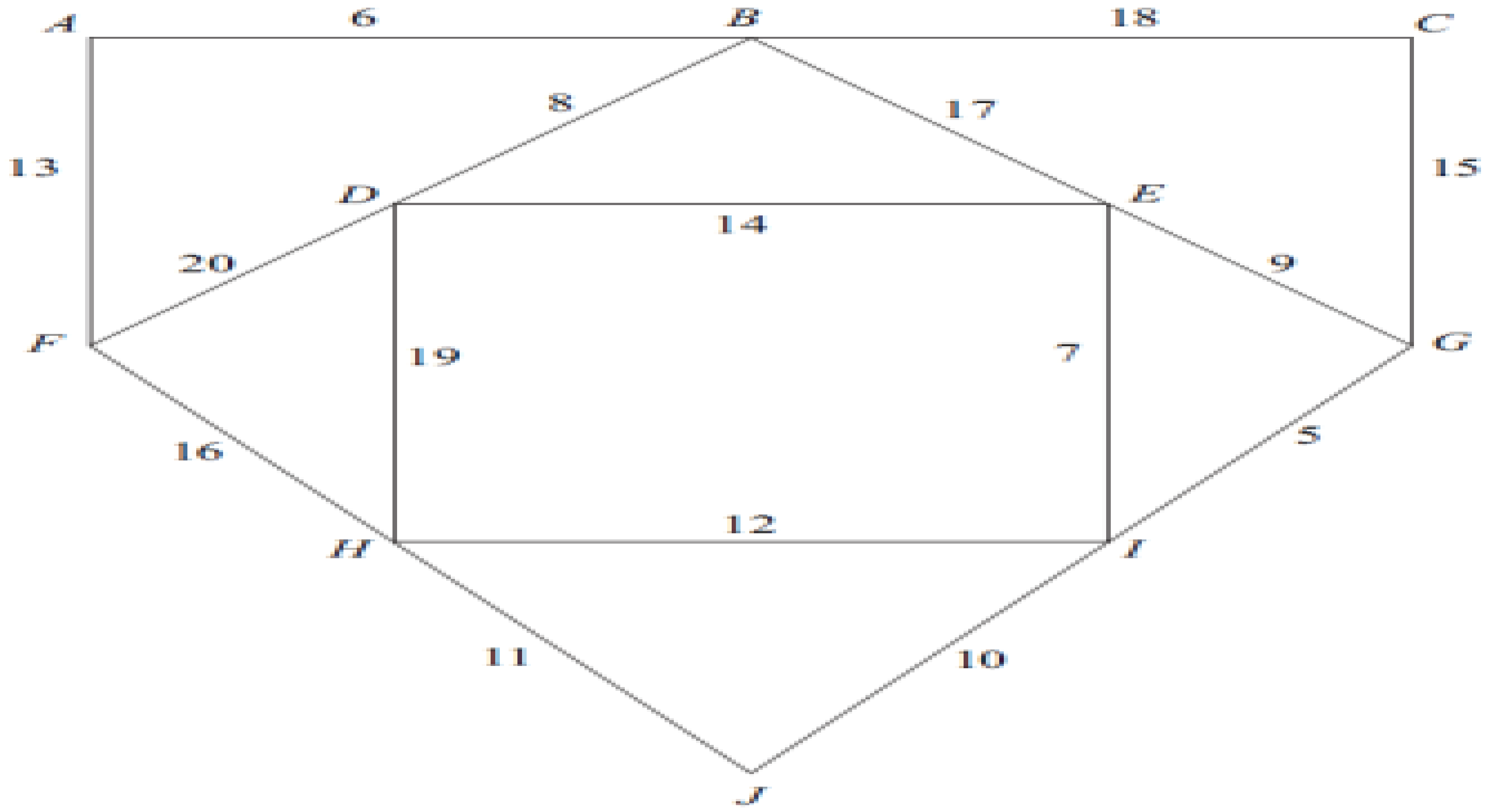
Ví dụ 2 (tự làm): Using Prim's Algorithm



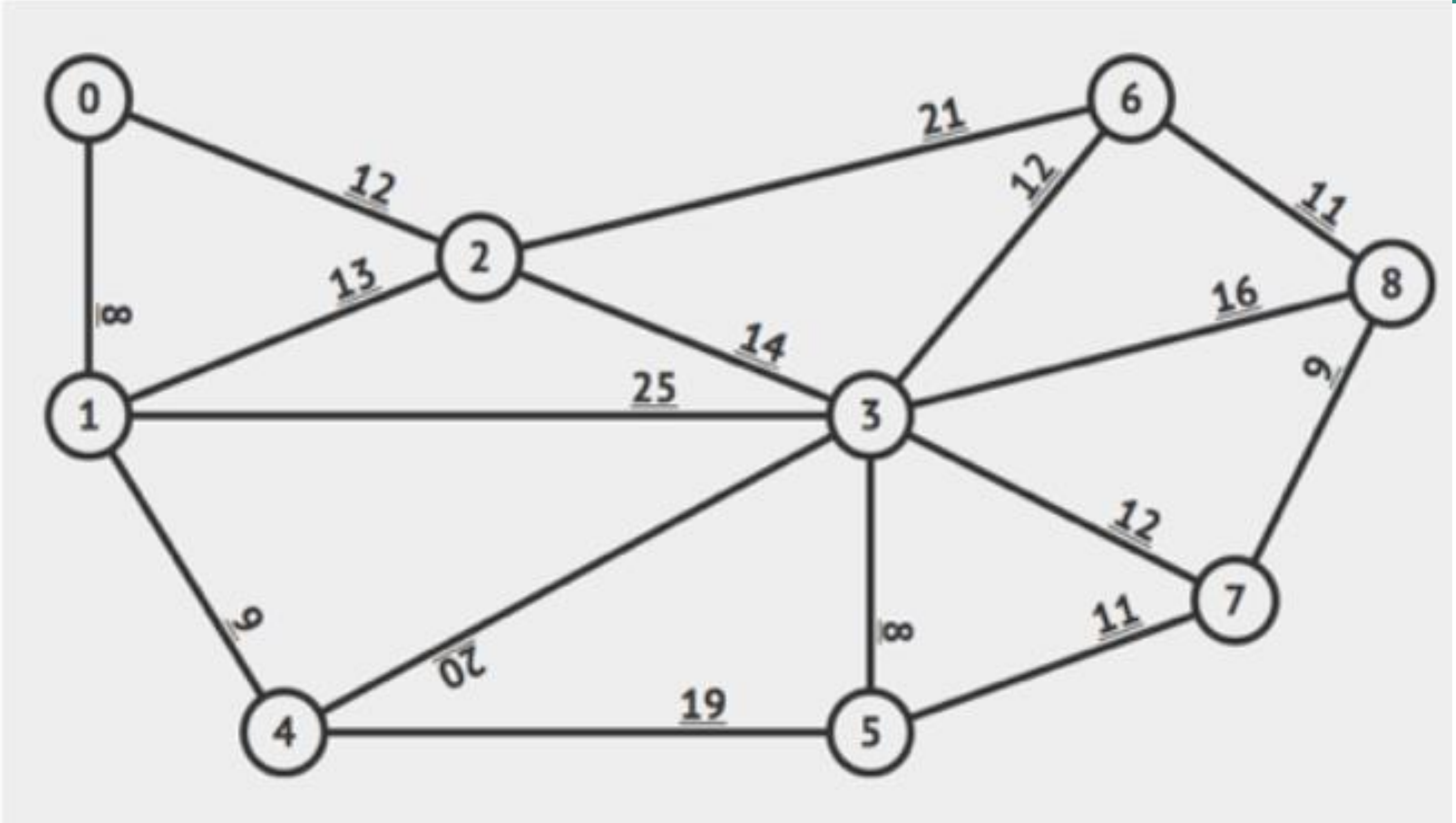
Ví dụ 3 (tự làm): Using Prim’s Algorithm



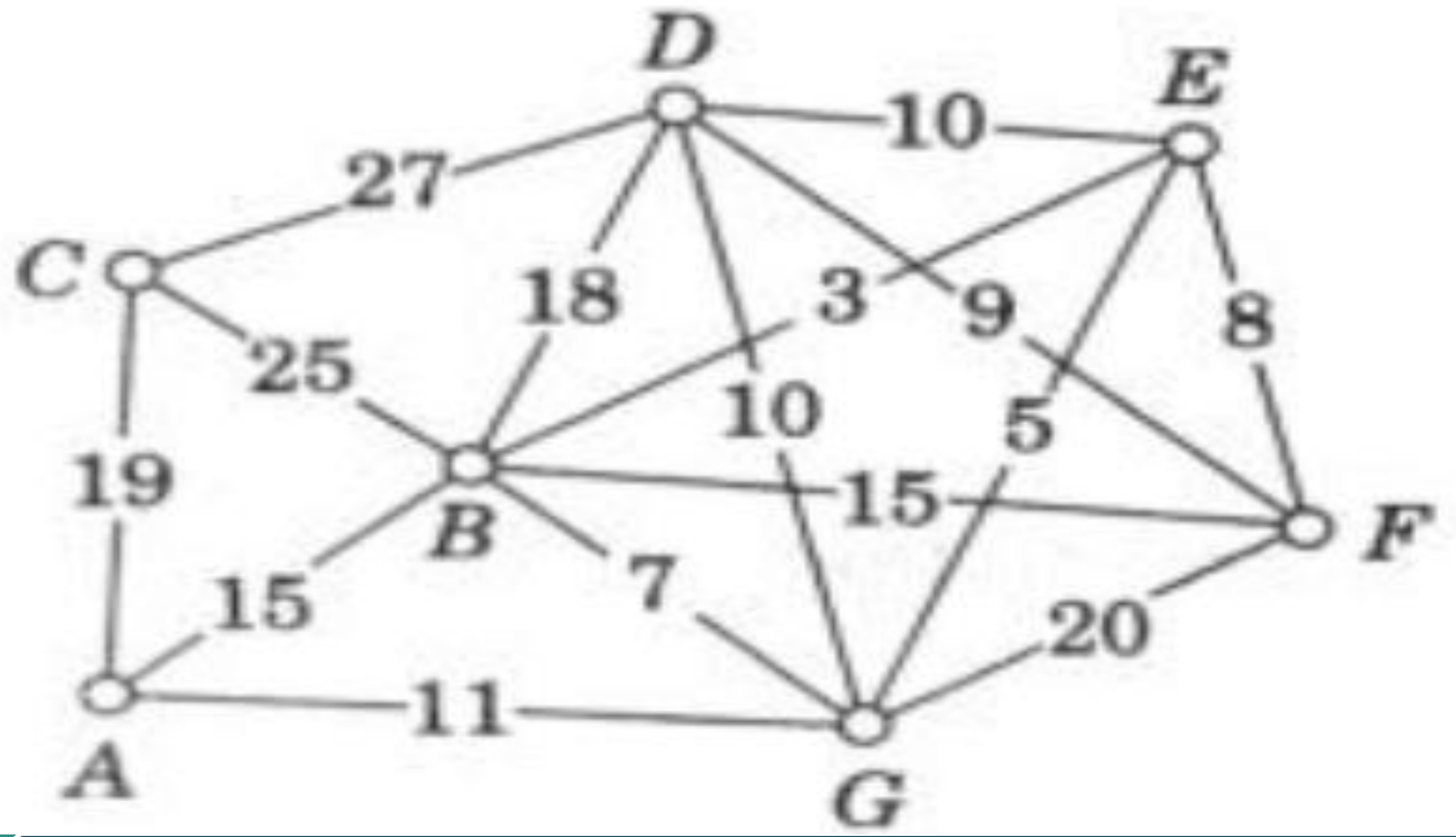
Ví dụ 4 (tự làm): Using Prim’s Algorithm



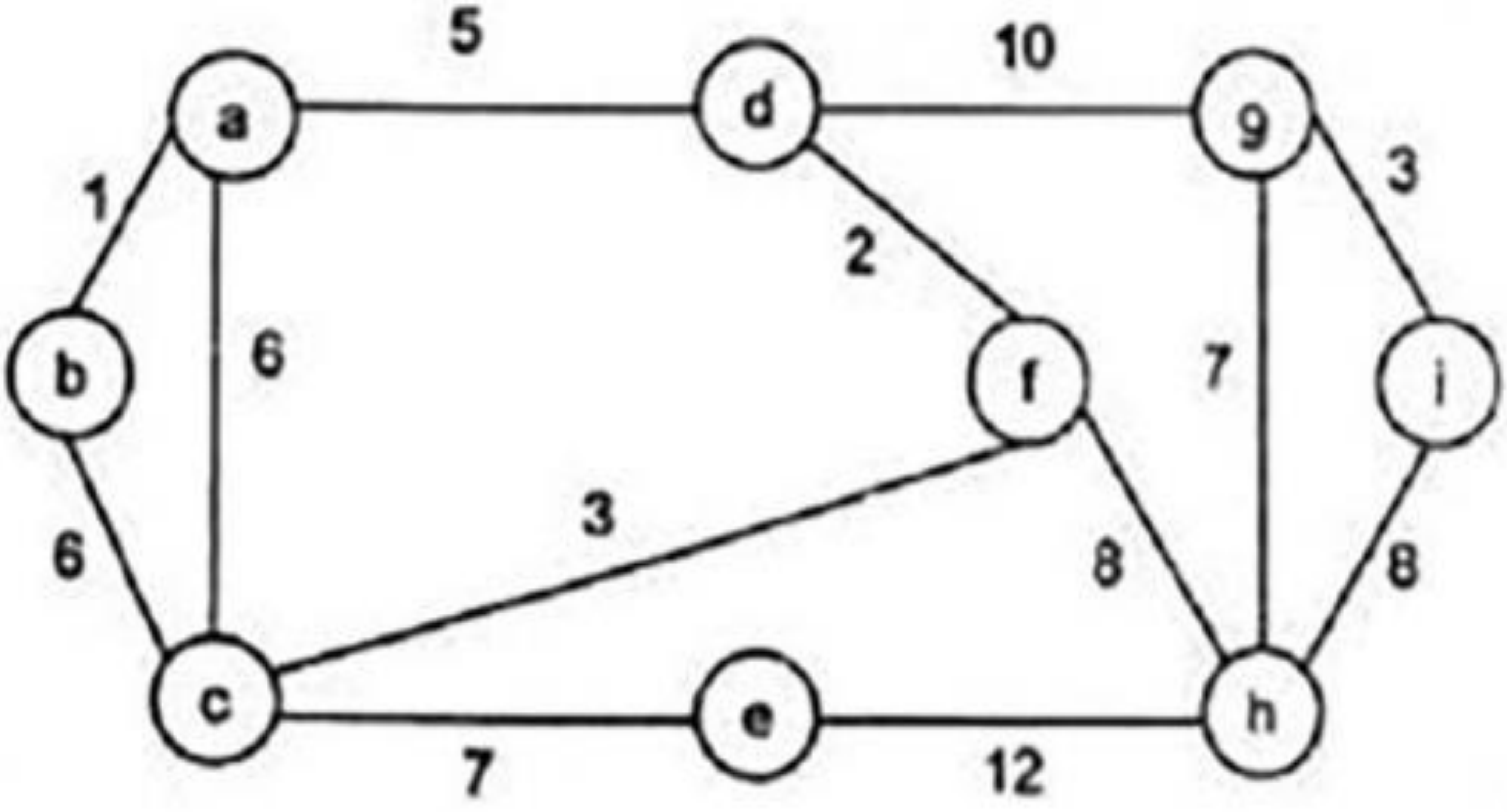
Ví dụ 5 (tự làm): Using Prim’s Algorithm



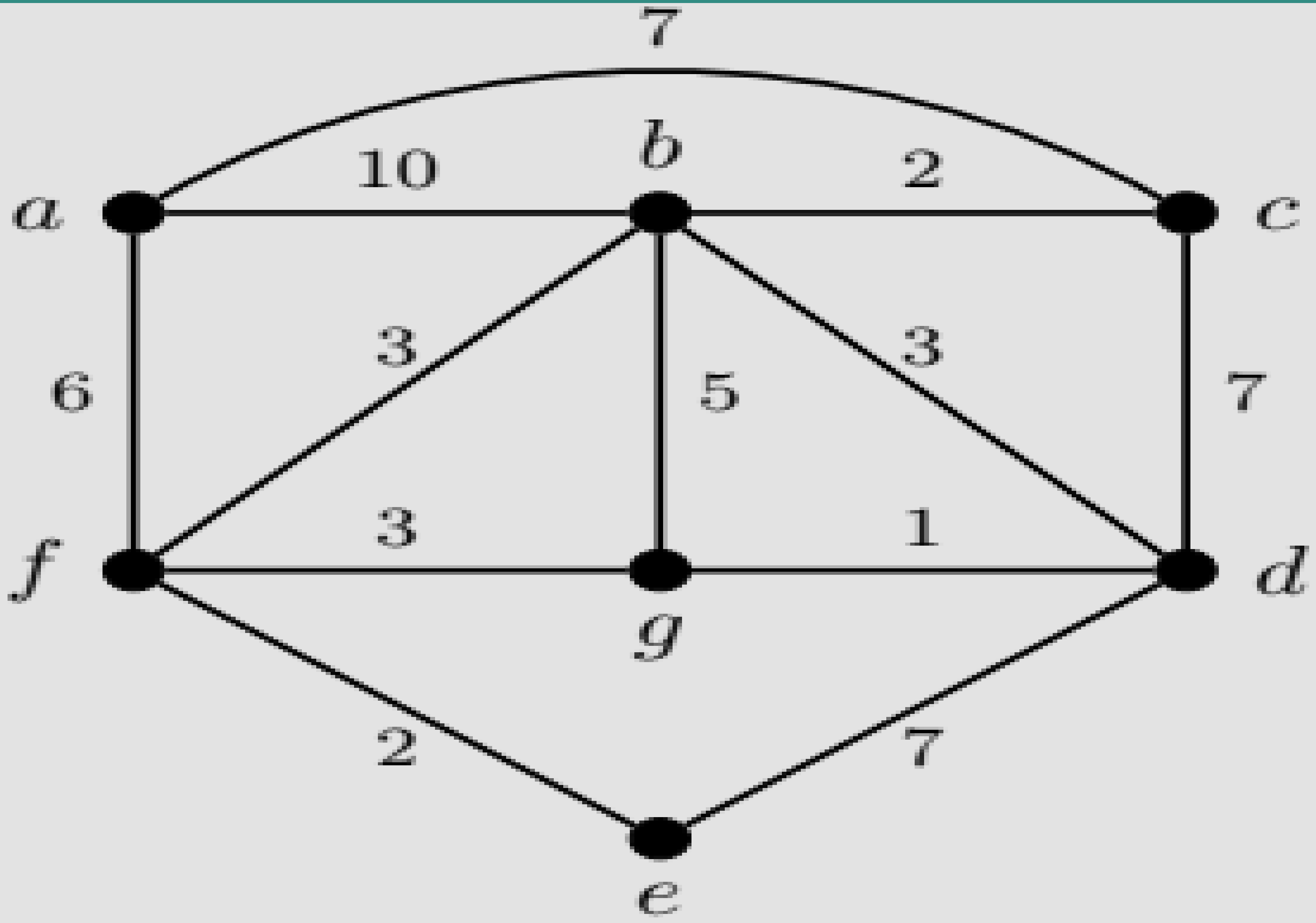
Ví dụ 6 (tự làm): Using Prim's Algorithm



Ví dụ 7 (tự làm): Using Prim’s Algorithm



Ví dụ 8 (tự làm): Using Prim's Algorithm



Thảo luận & bài tập (1/1)

❖ **Một số bài toán:**

1. Thuật toán Kruskal có thể sử dụng để kiểm tra tính liên thông hay không?
2. Làm thế nào để tìm cây khung cực tiểu với điều kiện nó phải chứa (các) cạnh cho trước nào đó?
3. Nên biểu diễn đồ thị dưới dạng nào (ma trận trọng số, danh sách kề, danh sách cạnh) để hỗ trợ tốt cho các thuật toán Prim và Kruskal?
4. Độ phức tạp tính toán của 2 thuật toán này?
5. Xây dựng thuật toán kiểm tra đồ thị có chứa chu trình (đơn) hay không?
6. Cài đặt các thuật toán trên máy tính?