

COMP 282

Advanced Data Structures

Lecture 06

Graphs

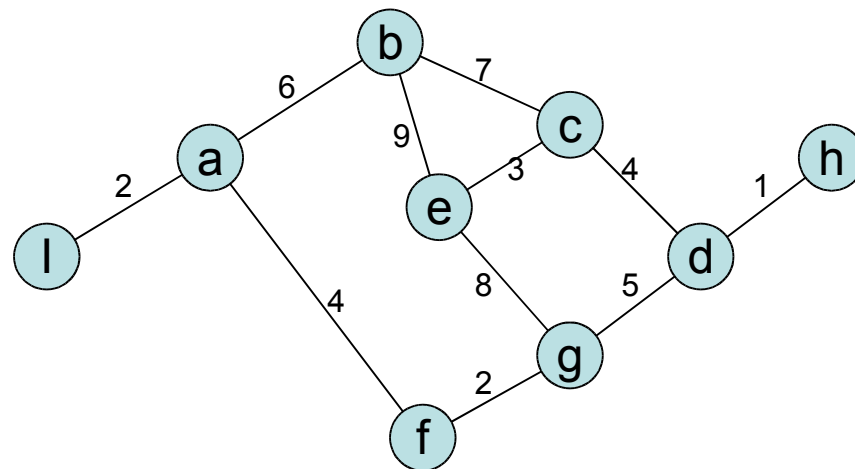
Spanning Trees

Minimum Spanning Trees

- If the edges in the graph have weights then we can consider “minimum” spanning trees.
- A minimum spanning tree is the set of edges that have the smallest cumulative weight yet produce a valid spanning tree.

An Example

- Assume the graph is cities with distances between cities...

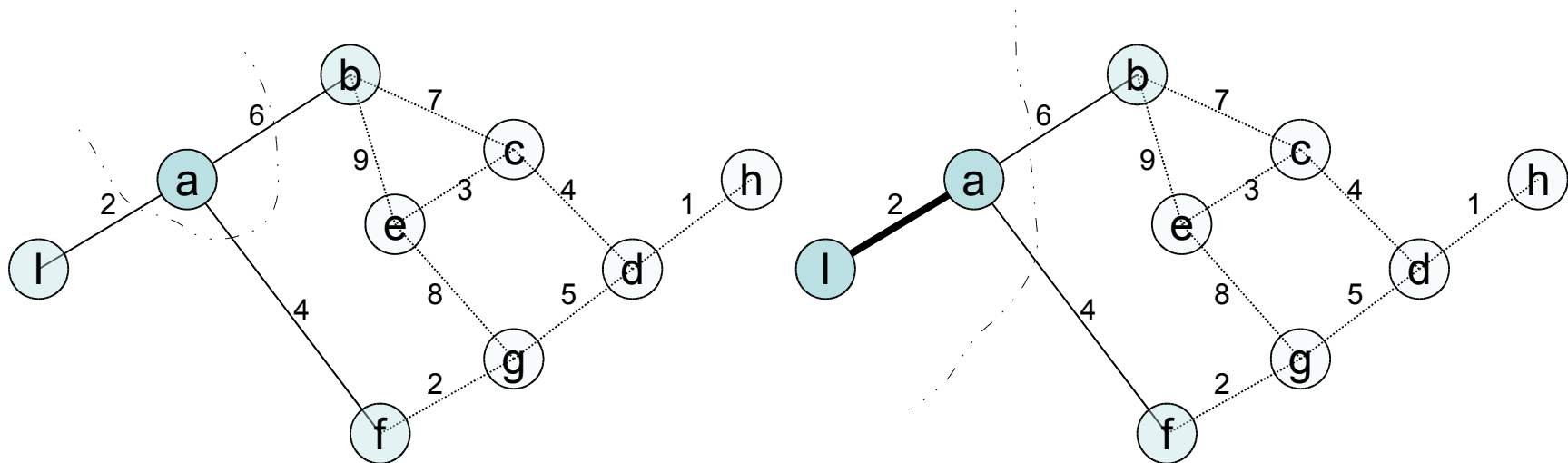


Prim's algorithm

- Start with any vertex v , mark it and include it in the minimum spanning tree.
- While there are unmarked vertices:
 - Find the least cost edge (v,u) such that v is marked [marked = “in spanning tree”] and u is not.
 - Add u and edge (v,u) to the minimum spanning tree.

Example

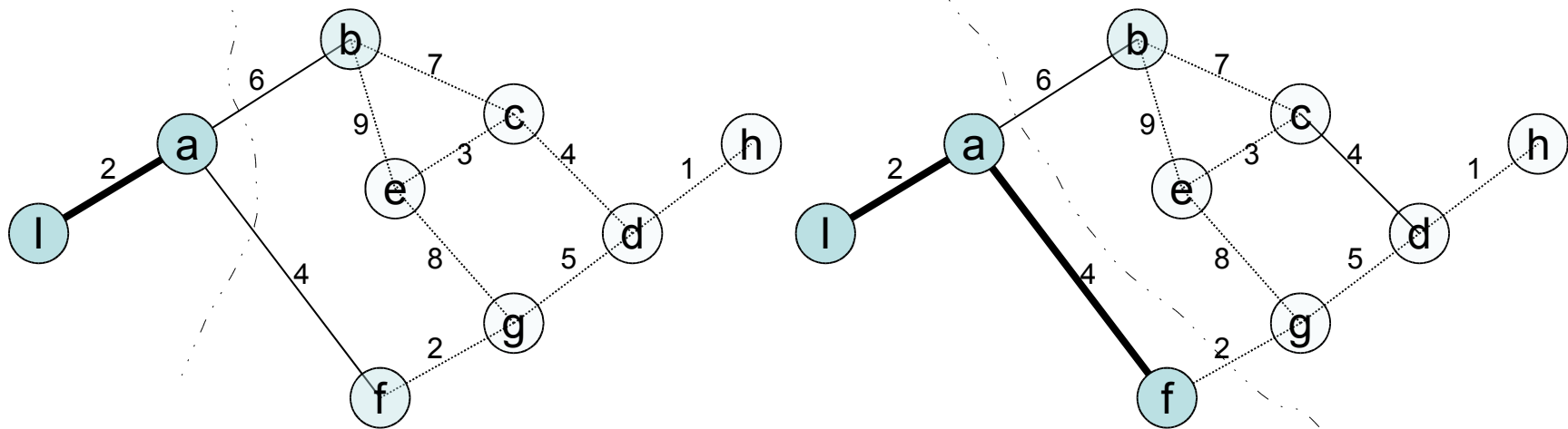
- Start with “a” consider edges (a,b) , (a,f) and (a,i) because they are connected to nodes already selected for spanning tree.



- (a,i) is smallest so we add that.

Prim's - example (cont)

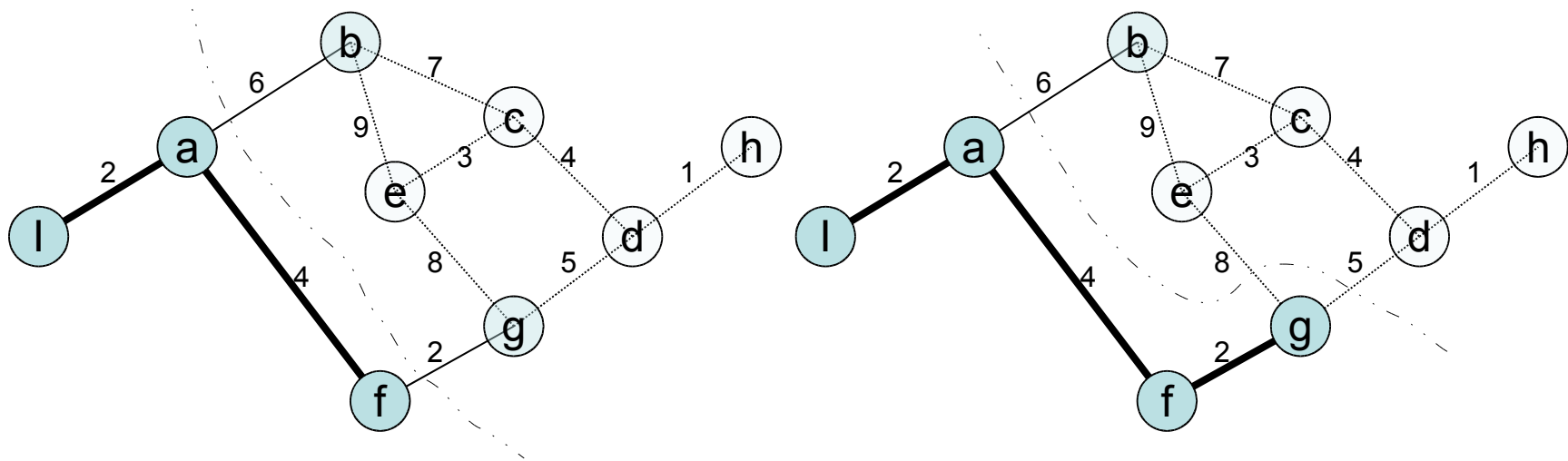
- Now we consider (a,b) and (a,f) because they are the only two edges connected to the spanning tree.



- (a,f) is smallest so add that.

Prim's – example (cont)

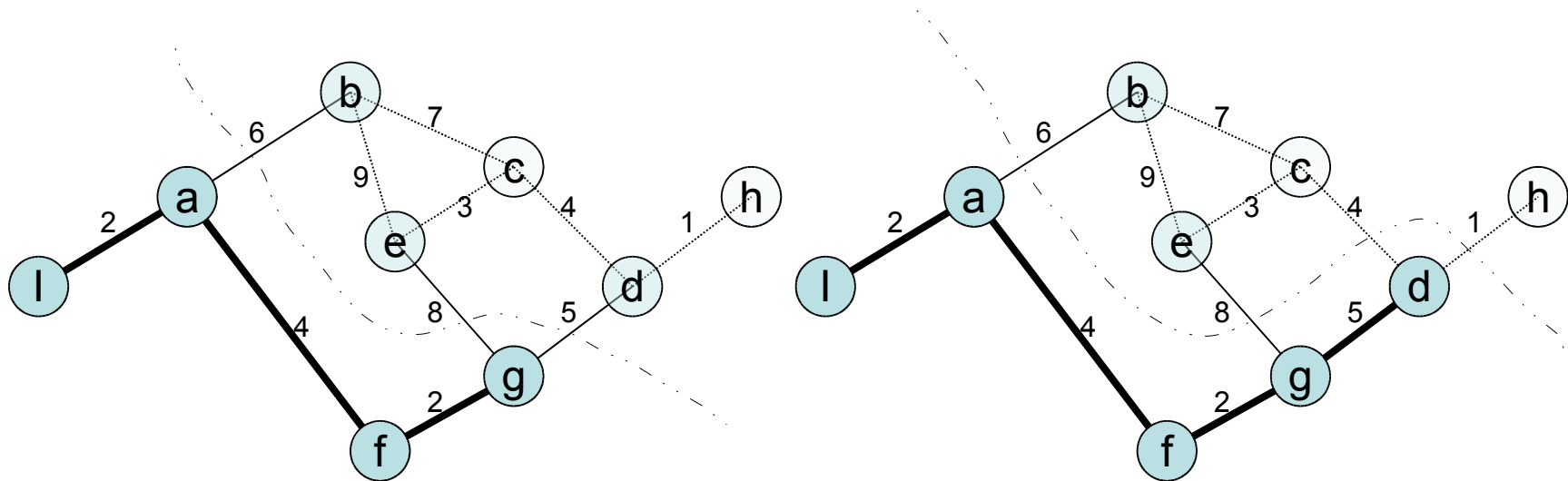
- Now (a,b) and (f,g) are candidates for u .



- (f,g) is smallest so add that

Prim's – example (cont)

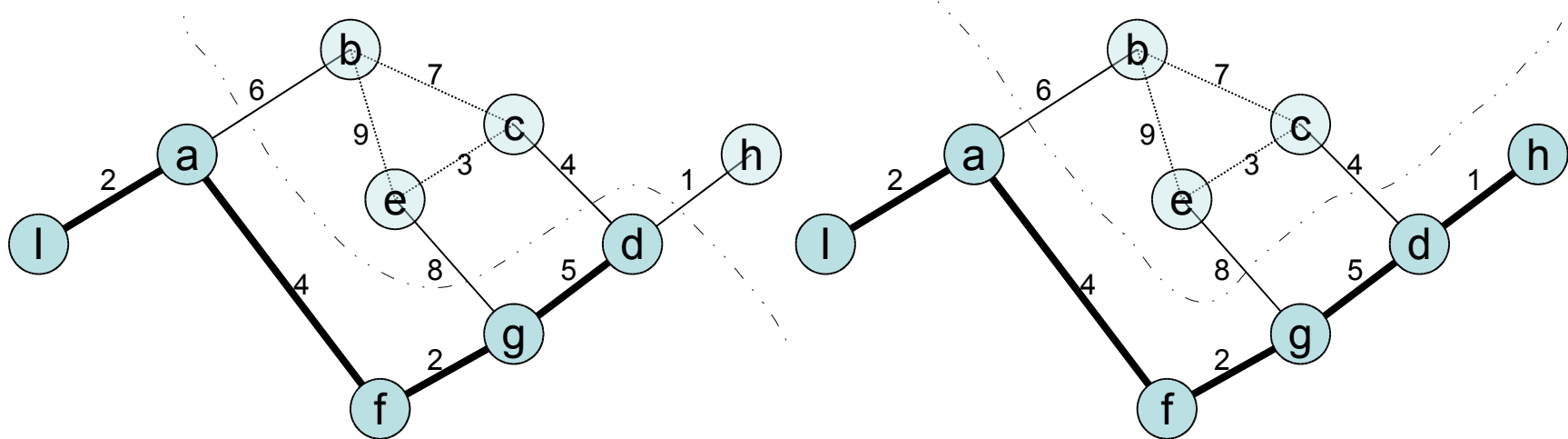
- Now, (a,b) , (e,g) and (d,g) are candidates



- (d,g) is smallest so add that

Prim's – example (cont)

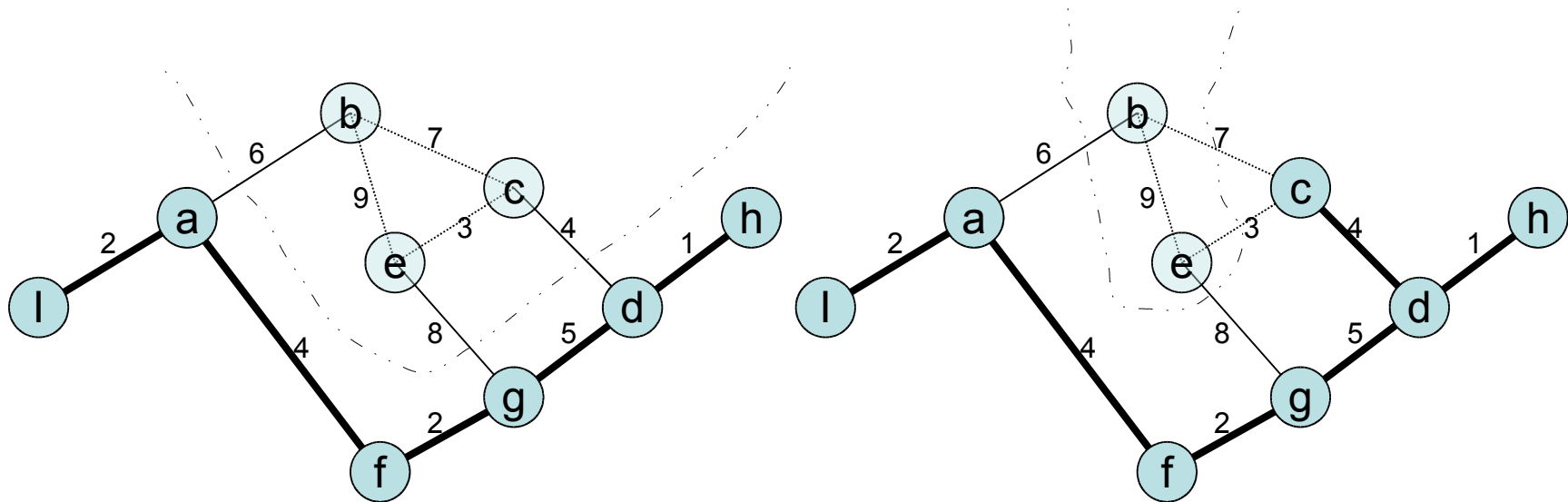
- Now (a,b) , (c,d) , (e,g) and (d,h) are candidates



- (d,h) is smallest so add that.

Prim's – example (cont)

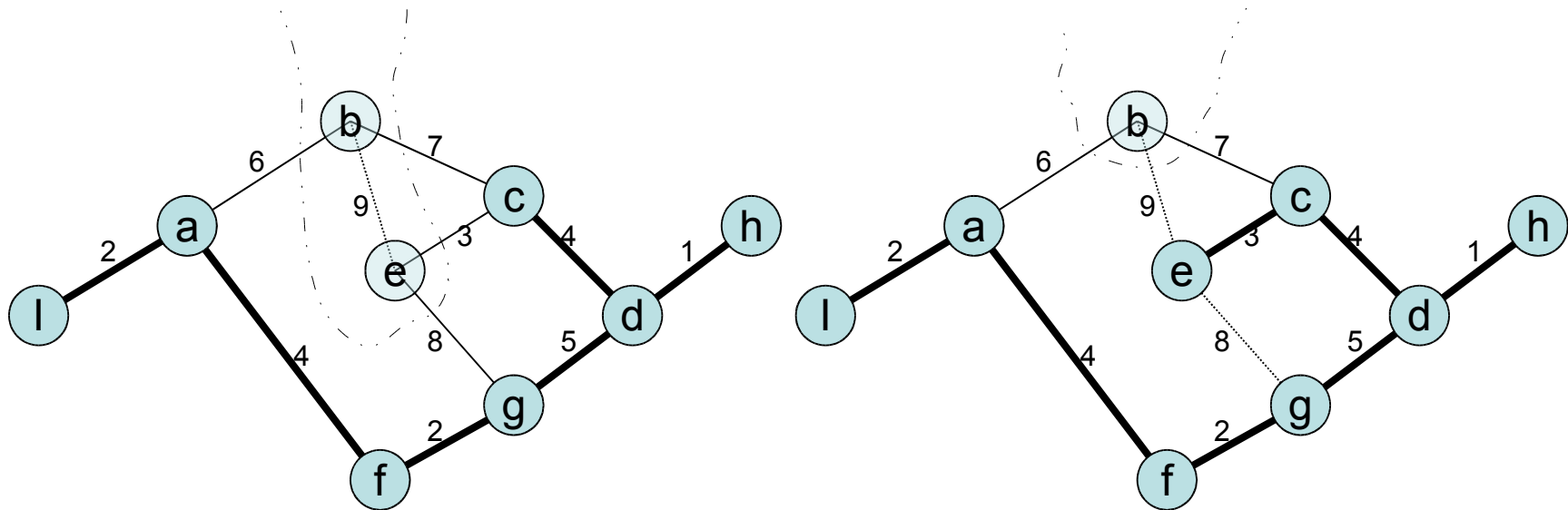
- Candidates are (a,b) , (c,d) and (e,g)



- (c,d) is smallest so add that.

Prim's – example (cont)

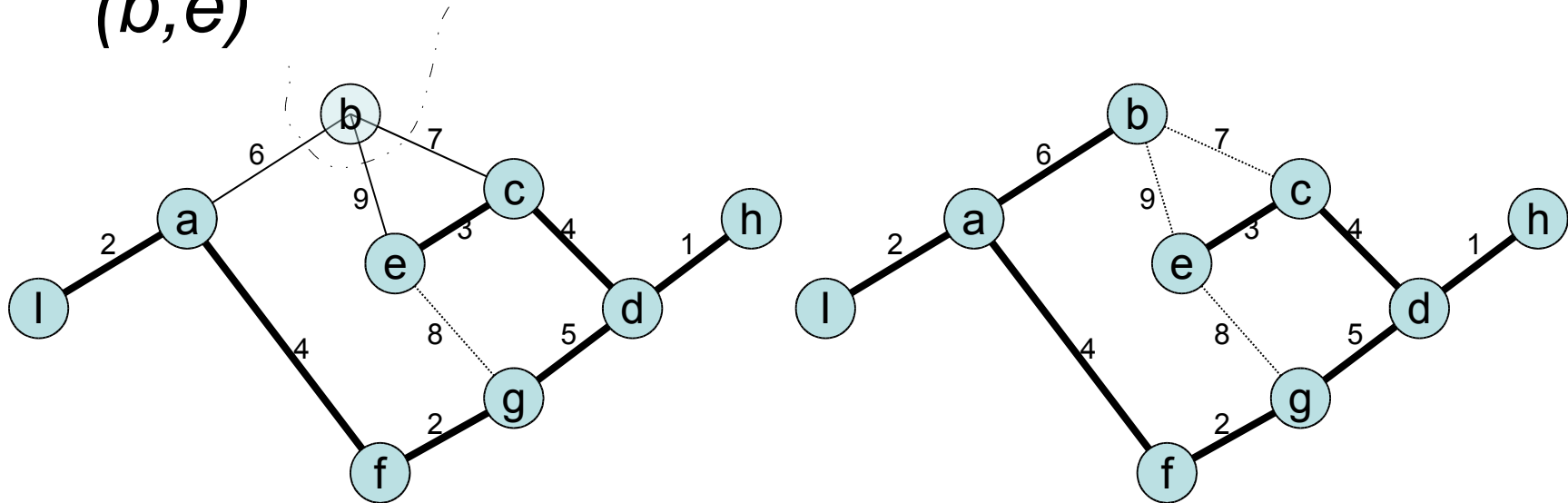
- Candidates are (a,b) , (b,c) , (c,e) and (g,e)



- (c,e) is smallest so add that.

Prim's – example (cont)

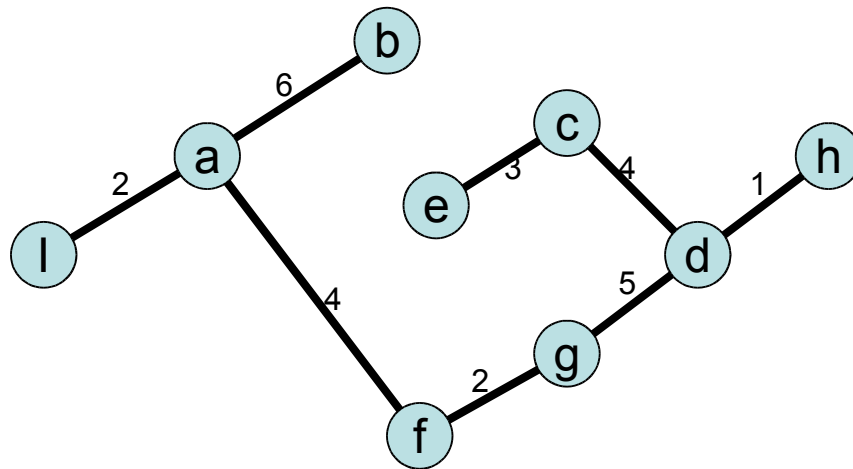
- Now the candidates are (a,b) , (b,c) and (b,e)



- (a,b) is smallest so that is added completing the minimum spanning tree.

Prim's result

- Resulting Spanning Tree



Implementation details

- Prim's algorithm is $O(E \lg V)$
- By using fibonacci heaps to implement a priority queue this can be improved to $O(E+V \lg V)$
- Where E is the number of edges in the graph and V is the number of vertices.