Minimum cost spanning trees

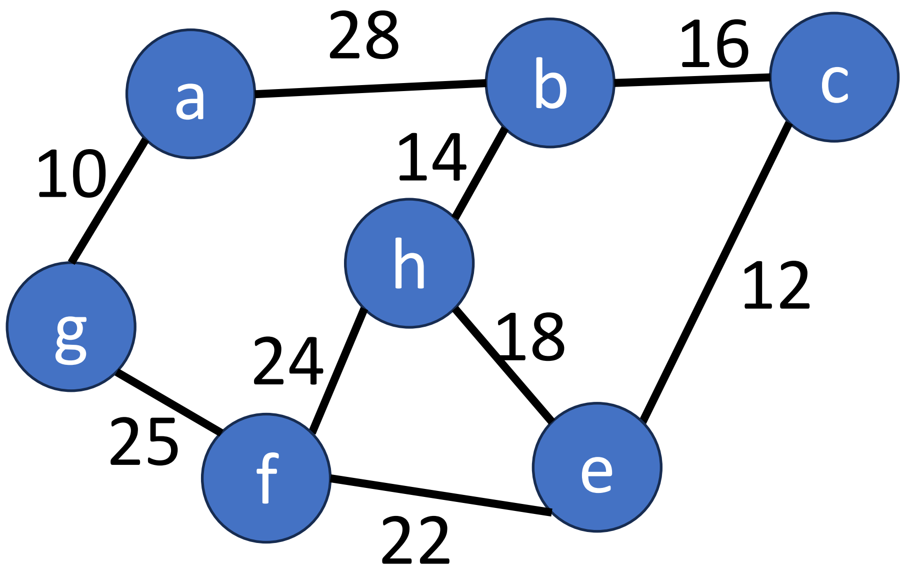
Ch. 6.3

Minimum cost spanning tree

- In a weighted <u>connected</u> <u>undirected</u> graph G
  - N: number of vertices

- A spanning tree of least cost
  - Cost = Sum(weights of edges in the spanning tree)
  - Edges within the graph G
  - Number of edges = $N - 1$

Example



- Network has 9 edges and 7 vertices.
- Spanning tree should have $N - 1 = 6$ edges.
    - Strategy 1: select 6 edges.
    - Strategy 2: remove 3 edges.

Edge selection

To build a minimum spanning tree
- N vertices, N – 1 edges
- Minimum costs

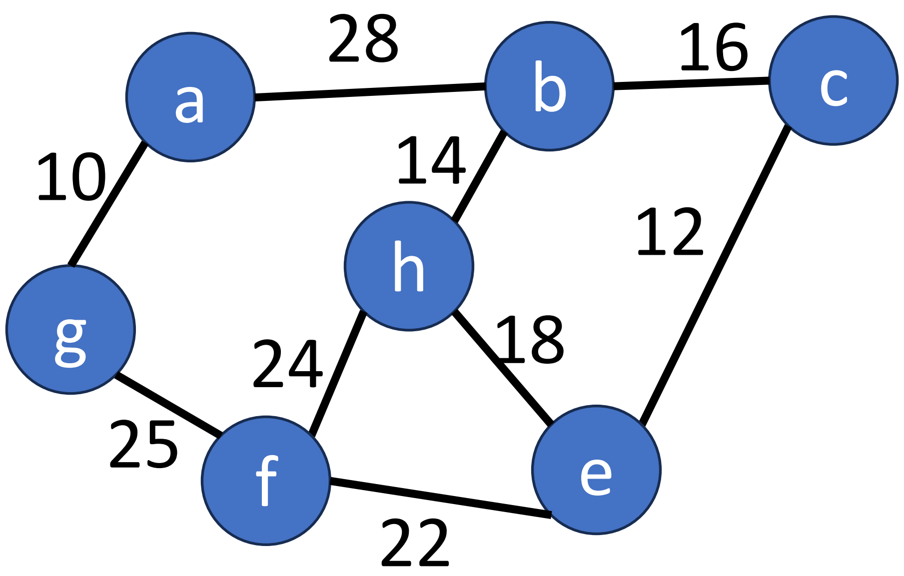
- Method 1:
  - Start with an *N*-vertex 0-edge forest.
  - Select edges in nondecreasing order of cost.
    - If not form a cycle with the edges that are already selected.

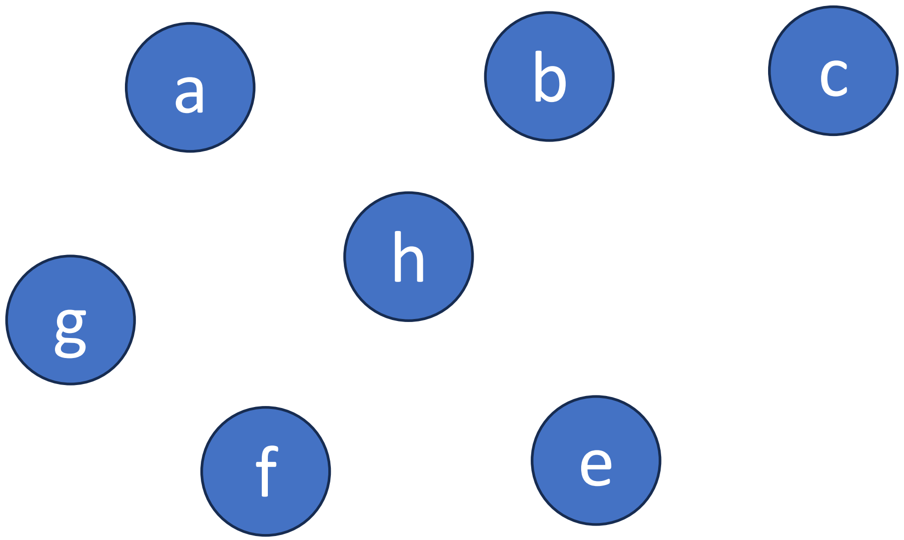
Kruskal's method

- Method 2:
  - Start with a 1-vertex tree T.
  - Grow the tree T by repeatedly adding a least cost edge (*u*, *v*).
    - Only one of *u* or *v* is in T.

Prim's method

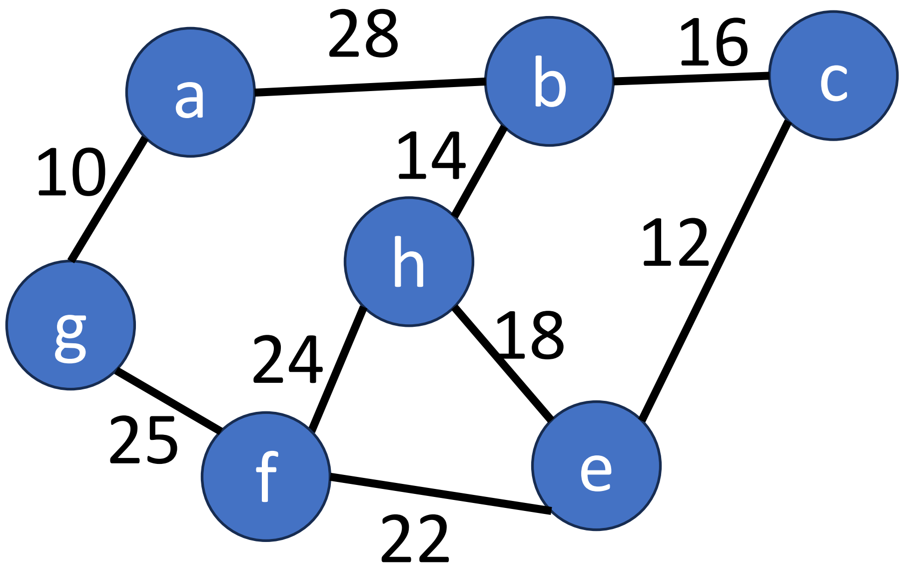
Kruskal’s method

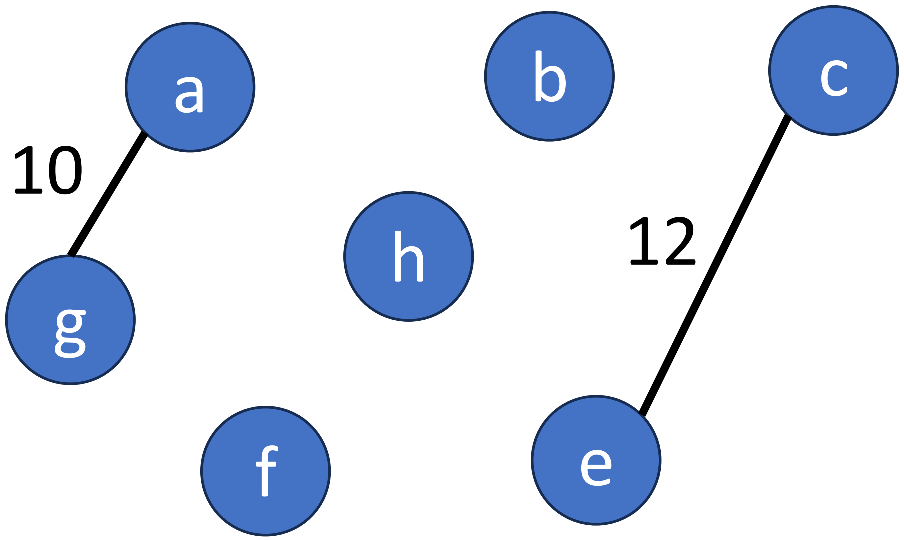




- Start with a 7-vertex 0-edge forest.
- Examine the edges in nondecreasing order
  - 10, 12, 14, 16, 18, 22, 24, 25, 28
- Edge (a,g) is the first considered for inclusion.

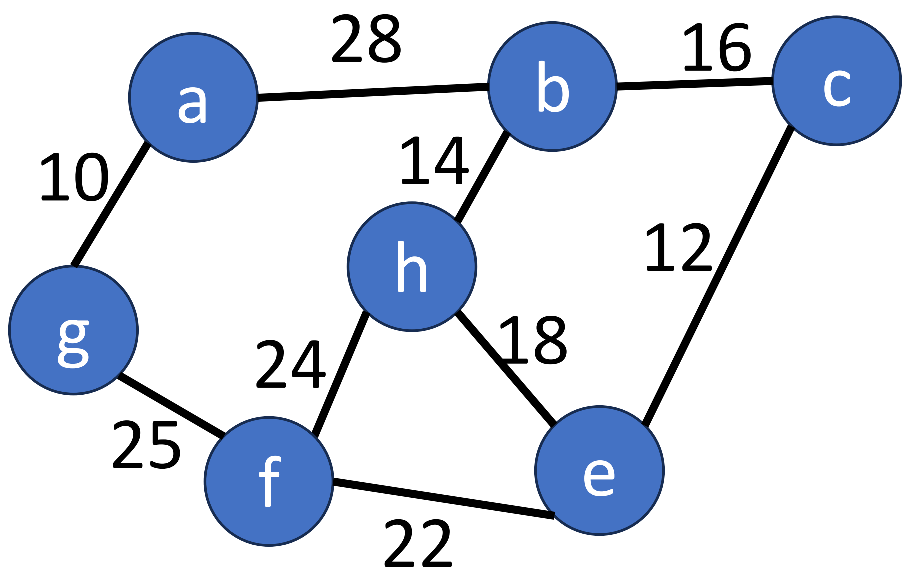
Kruskal’s method

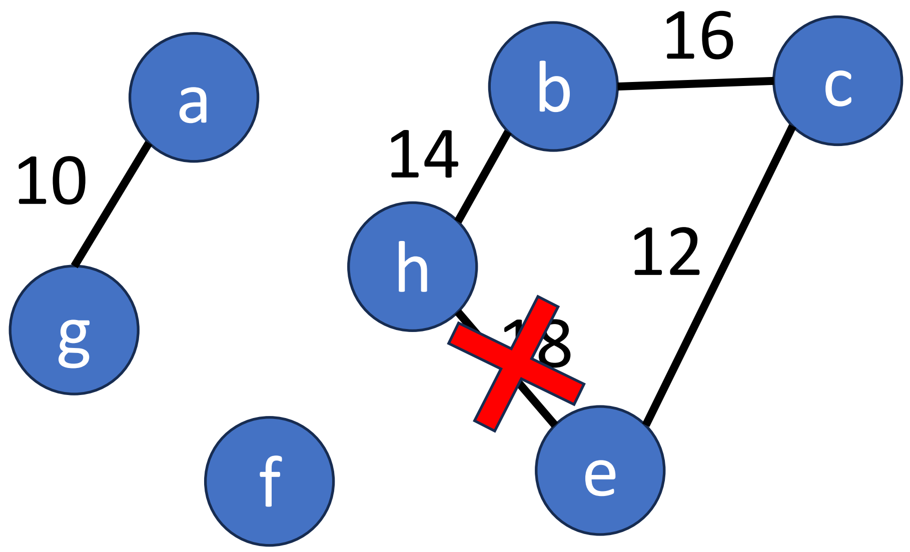




- Examine the remaining edges in nondecreasing order
  - ~~10,~~ 12, 14, 16, 18, 22, 24, 25, 28
- Edge (c,e) is considered next and added.

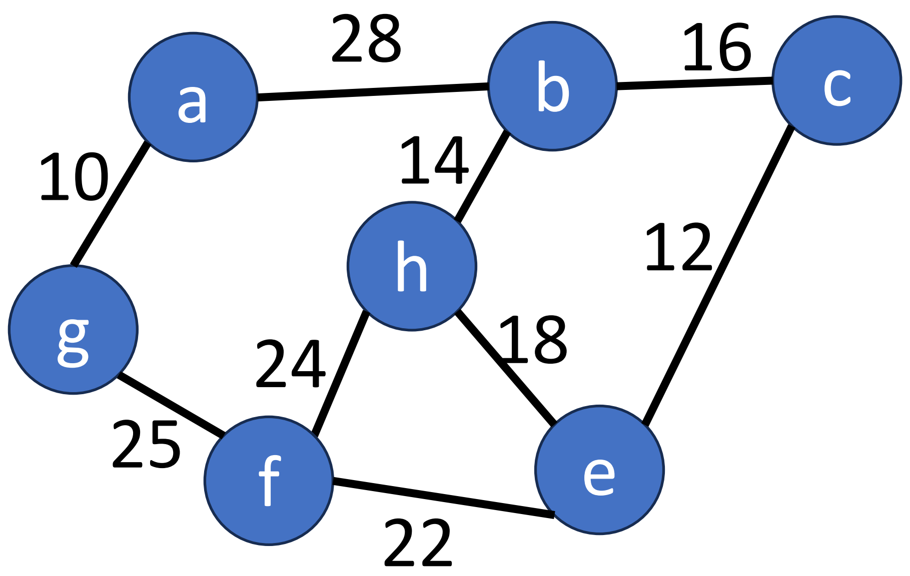
Kruskal’s method

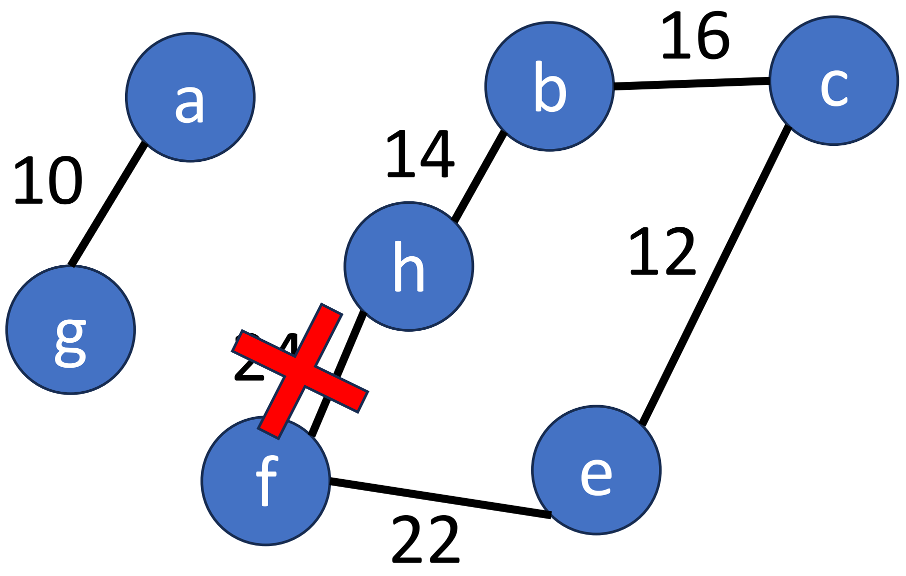




- Edge (b,h) is considered next and added.
- Edge (b,c) is considered next and added.
- Edge (h,e) is considered next but rejected because it creates a cycle.

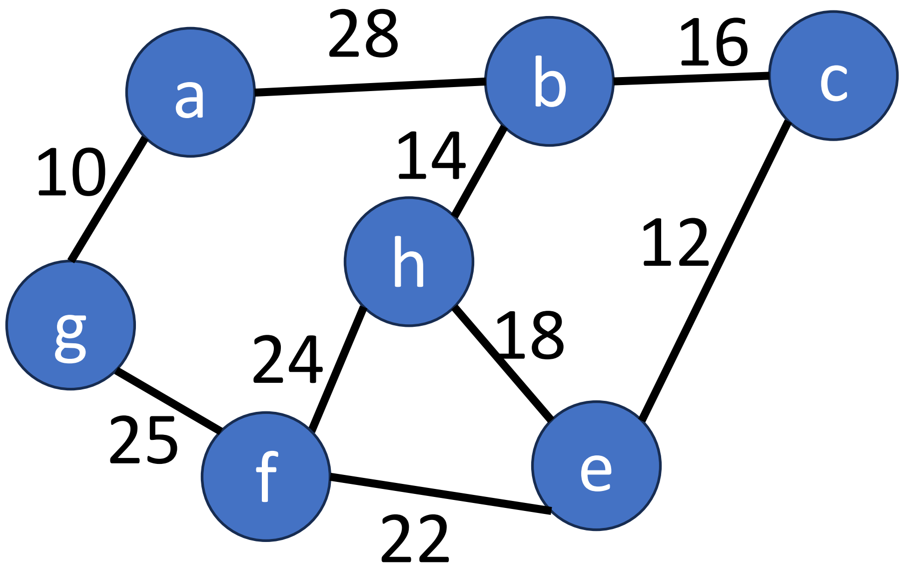
Kruskal's method

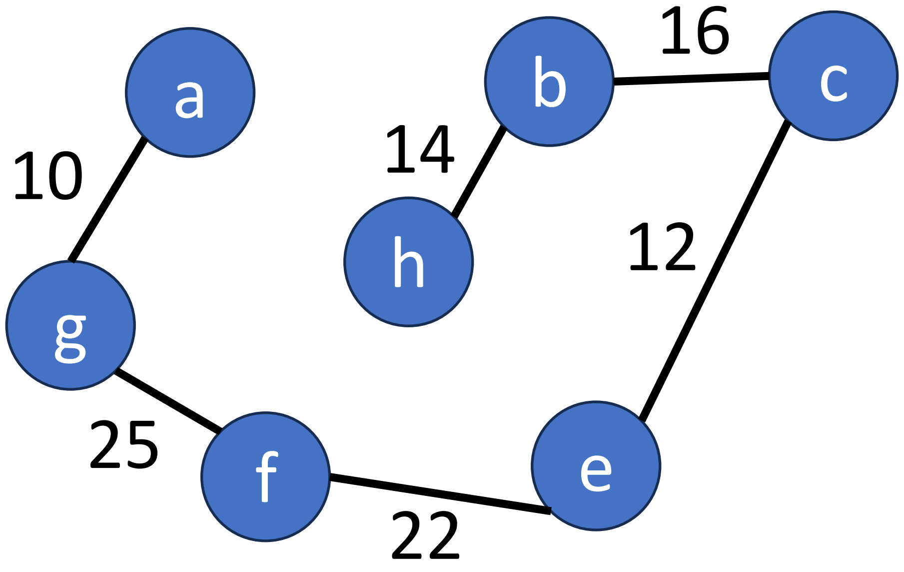




- Edge (e,f) is considered next and added.
- Edge (f,h) is considered next but rejected because it creates a cycle.

Kruskal’s method





- 6 edges are selected and no cycle forms.
  - Number of edges = $N - 1$
  - It’s a spanning tree.
- Cost = 99

Pseudocode for Kruskal's method

For these two operations, what data structure will you use?

```
T = {};
while (|T| < N-1 and E is not empty){
    choose a least cost edge (v,w) from E;
    E = E - {(v,w)}; /*delete edge from E*/

    if (adding (v,w) doesn't create a cycle in T)
       T = T + {(v,w)}; /*add edge to T*/
}
if (|T| == N - 1) T is a minimum cost spanning tree.
else There is no spanning tree.
```

Data structure for Kruskal's method (1)

- Operations related to E:
  - Check whether edge set E is empty.
  - Select and remove a least-cost edge.

- Use a min heap or leftist for edges.

  Time complexity:
  - Initialize: O(e)
  - Remove and return least-cost edge: O(log e)

Pseudocode for Kruskal’s method

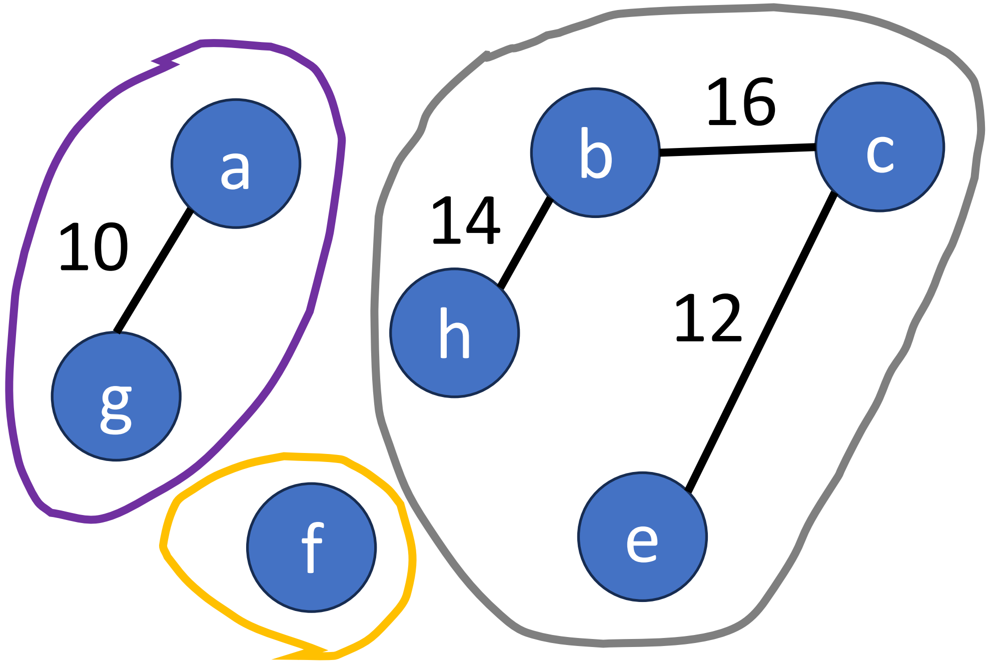
Operations related to selected edges T

```
T = {};
while (|T| < N-1 and E is not empty){
    choose a least cost edge (v,w) from E
    E = E - {(v,w)}; /*delete edge from E*/

    if (adding (v,w) doesn’t create a cycle in T)
        T = T + {(v,w)}; /*add edge to T*/
}
if (|T| == N - 1) T is a minimum cost spanning tree.
else There is no spanning tree.
```

Data structure for Kruskal's method (2)

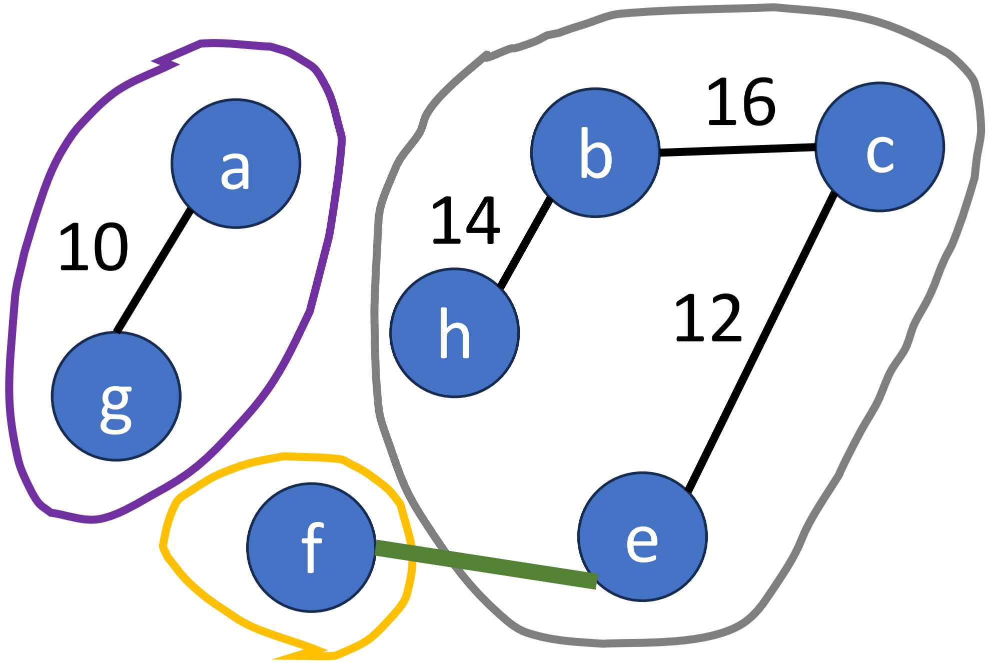
- Operations related to T:
  - Check whether T has *N*-1 edges.
  - Examine whether adding (v,w) to T creates a cycle.
  - Add an edge (v,w) to T.



- Each connected component in T is a set containing the vertices.
  - {a,g}, {f}, {h,b,c,e}
- Adding two vertices that are already connected creates a cycle.
  → Using ***find* operation** to determine whether u and w are in the same set.

Data structure for Kruskal's method (2)

- Operations related to T:
  - Check whether T has *N*-1 edges.
  - Examine whether adding (v,w) to T creates a cycle.
  - Add an edge (v,w) to T.

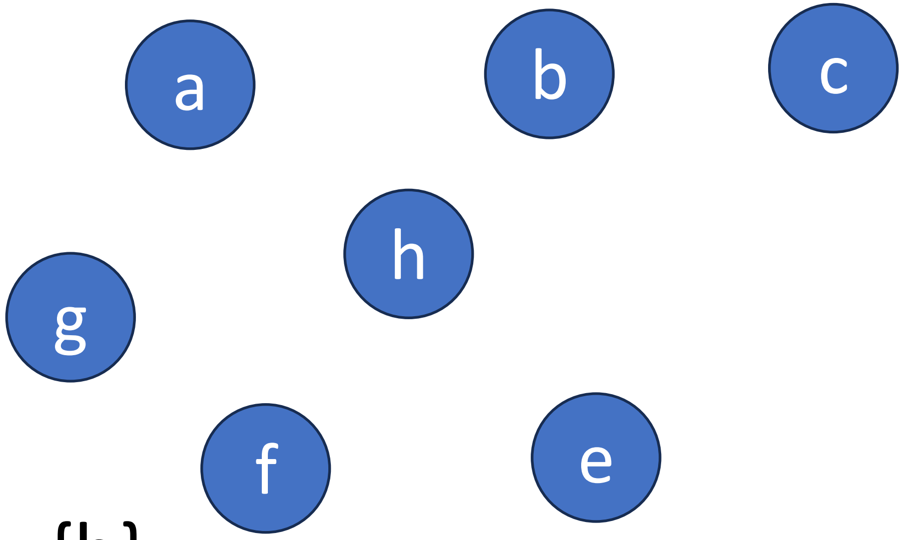


- If an edge (v,w) is added to T, the two connected components that have vertices v and w should be merged.
  → Using ***union operations*** to merge the set containing u and the set containing v.
  - {f} + {h,b,c,e} = {f,h,b,c,e}

Data structure for Kruskal’s method (3)

Use disjoint sets to process T

- Initially, T is empty.



- Initial sets are:
  - {a}, {b}, {c}, {e}, {f}, {g}, {h}
- Does add {u, v} to T create a cycle?
  If no, add edge to T.

Union-find operations

```
s1 = find(u);
s2 = find(v);
if (s1 != s2) union(s1,s2);
```

Time complexity of Kruskal's method

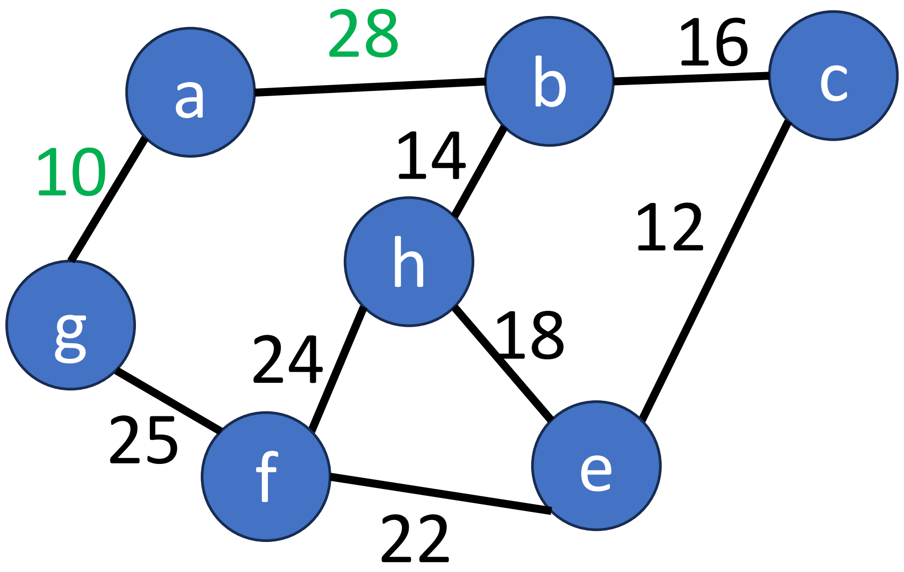
- Operations for edge set E:
  - Initialize min heap or leftist: O(e)
  - Operations to get minimum cost edge: O(log e)
    - At most e times of operation: O(e log e)

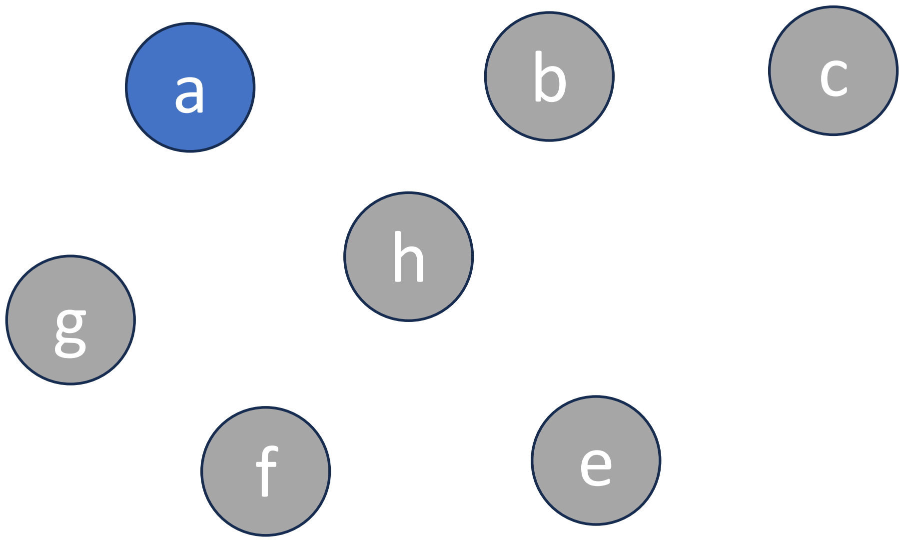
- Operations for vertices:
  - Initialize disjoint sets : O(n)
  - At most 2e finds and n-1 unions: close to O(e + n)

- Overall: O(e + e log e + n + e) = O(e log e)

For each iteration, time for union find is less than that for obtaining minimum cost edge.

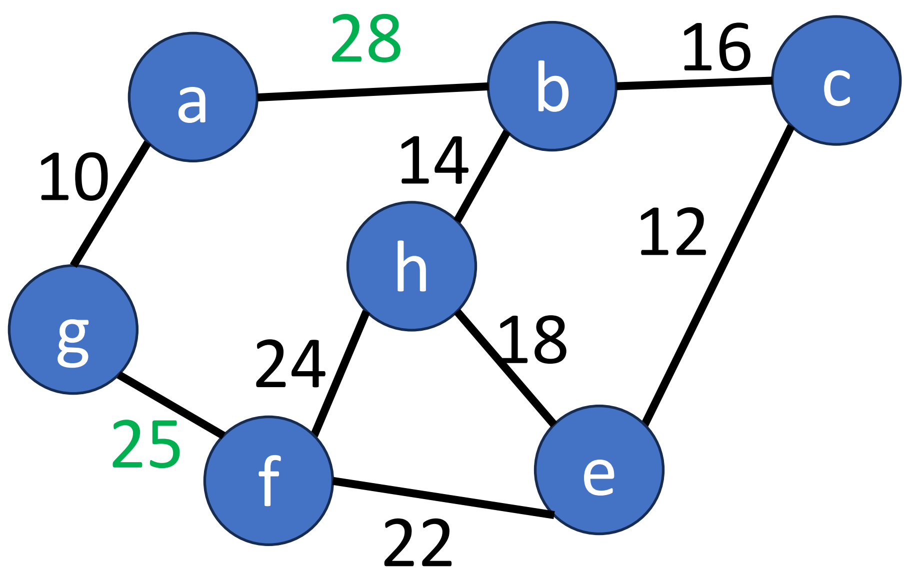
Prim’s method

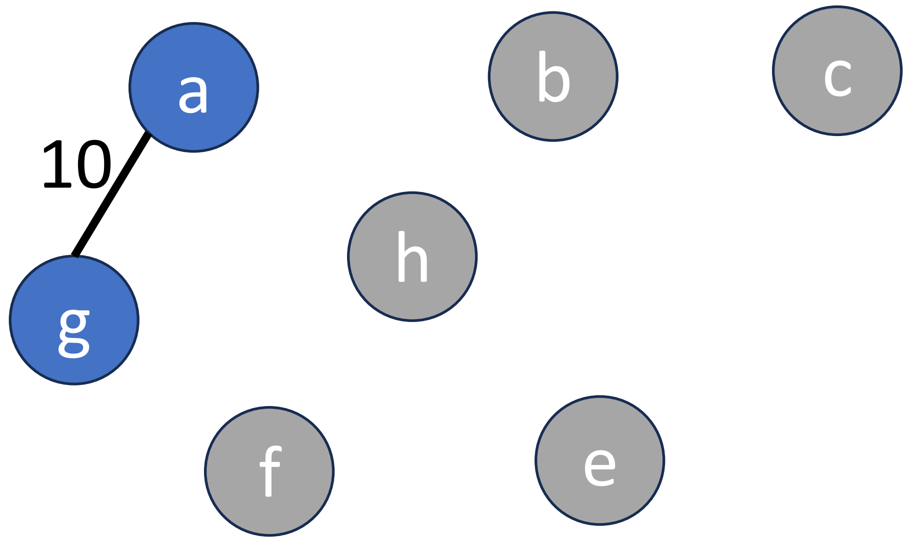




- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.

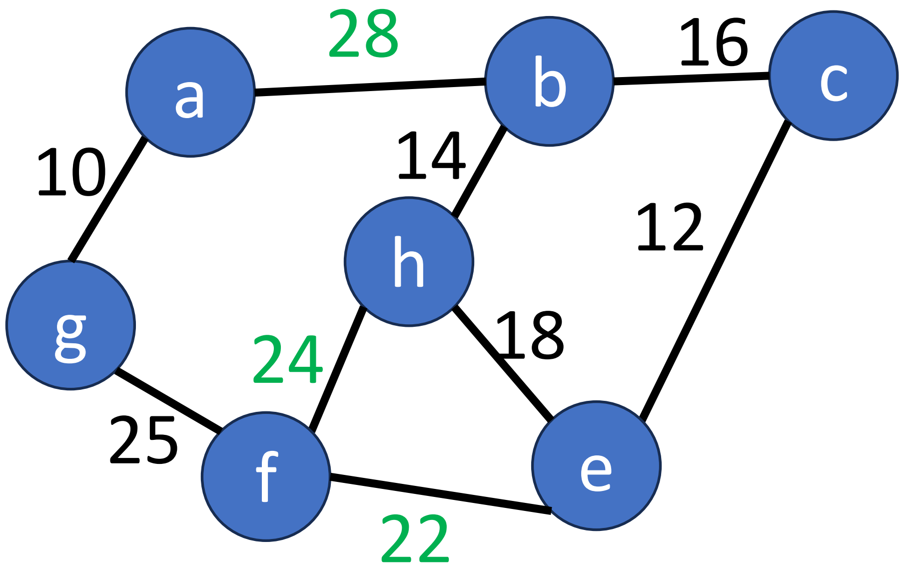
Prim’s method

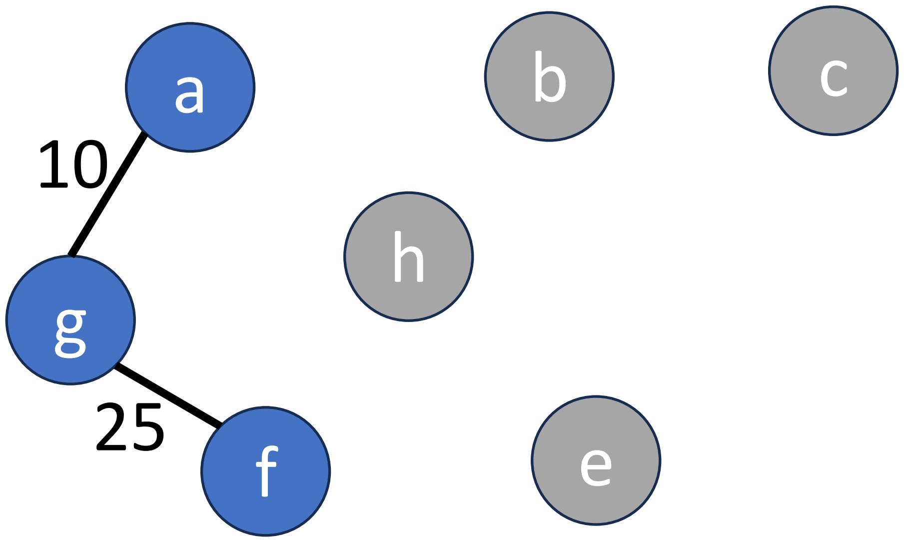




- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

Prim’s method

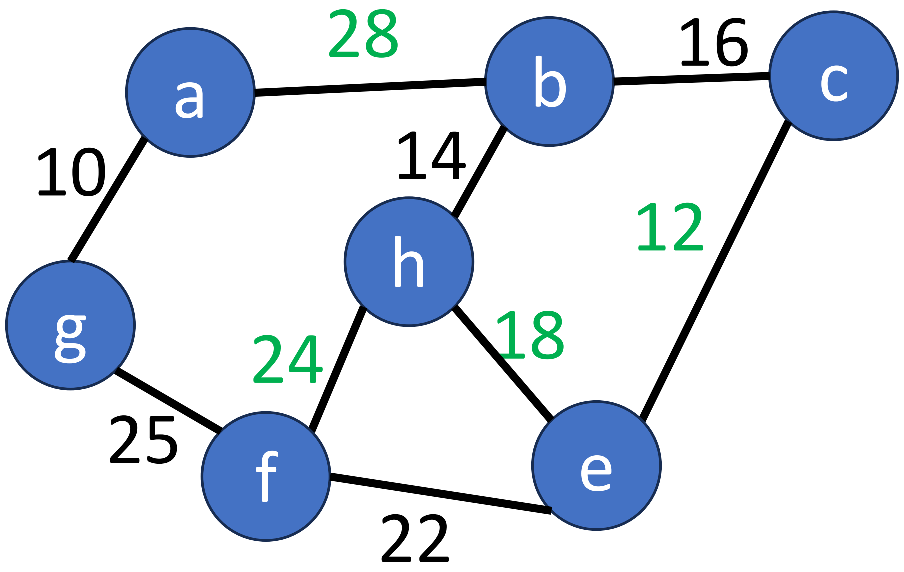


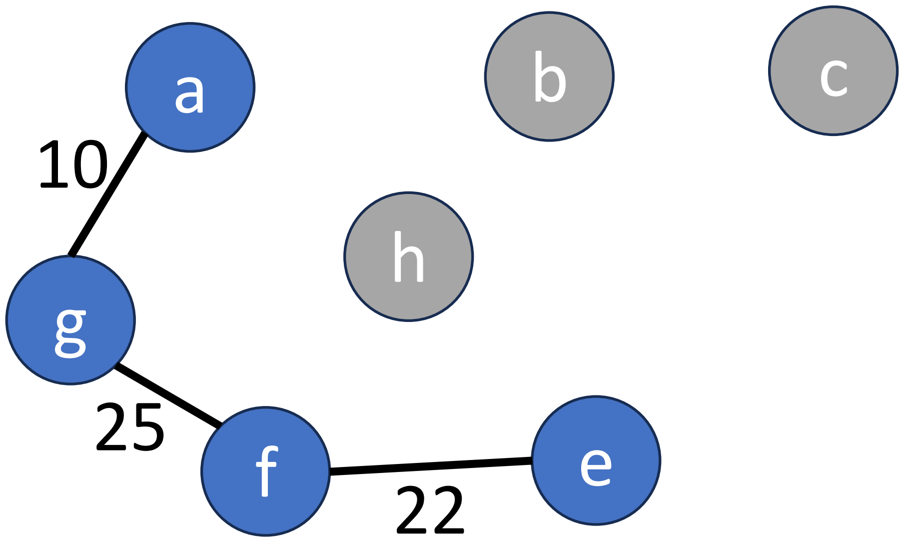


- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

….

Prim’s method

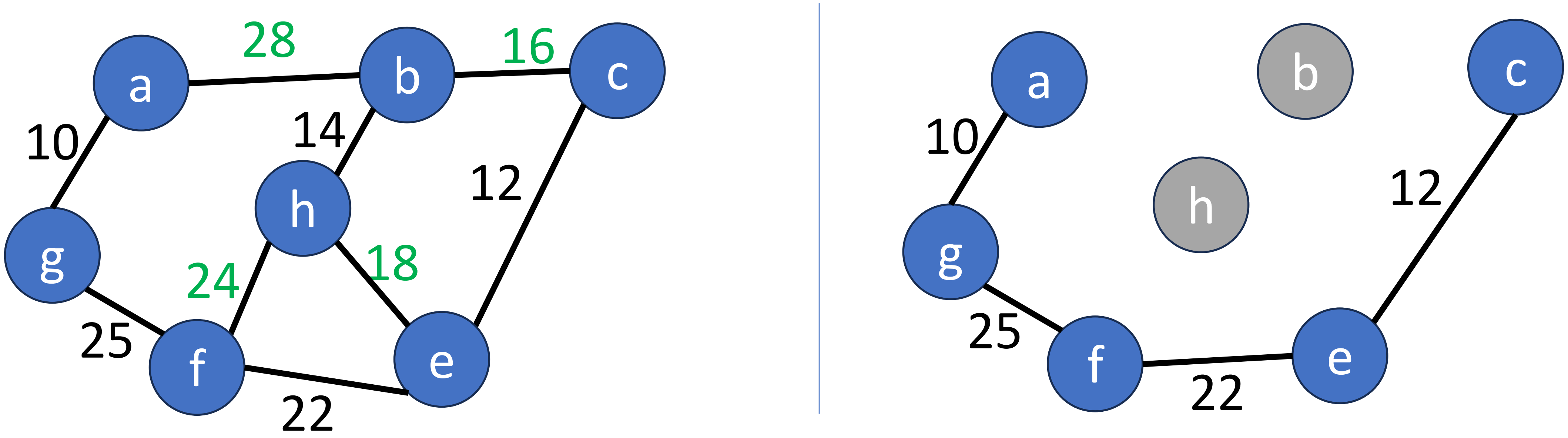




- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

  ….

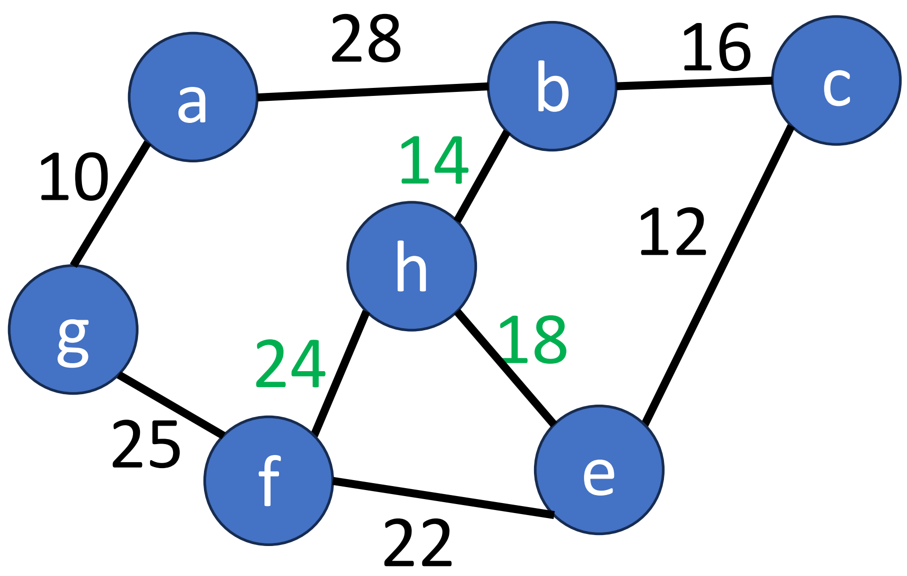
Prim’s method

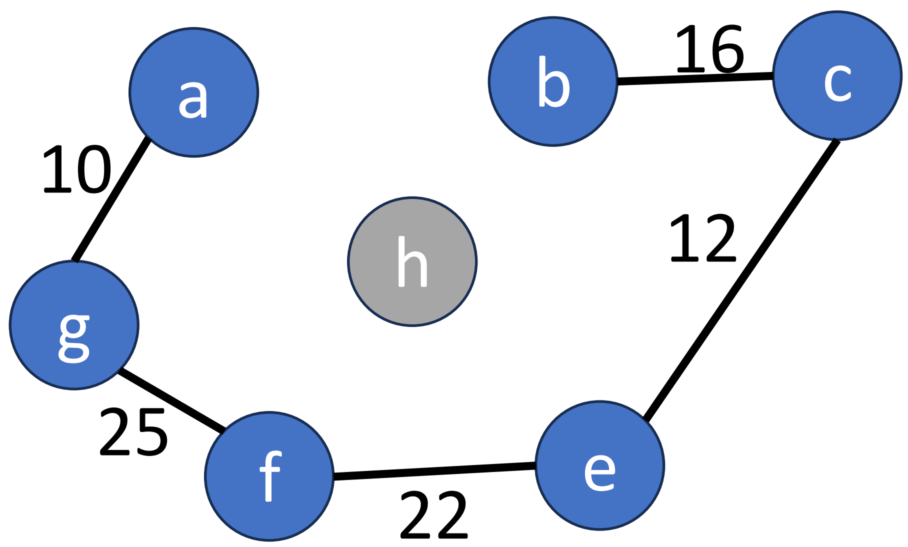


- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

  ….

Prim’s method



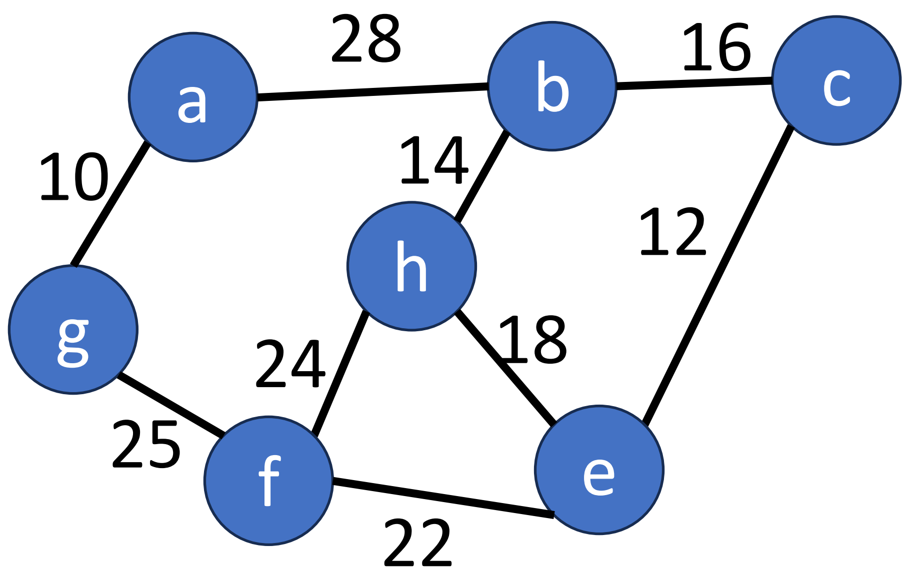


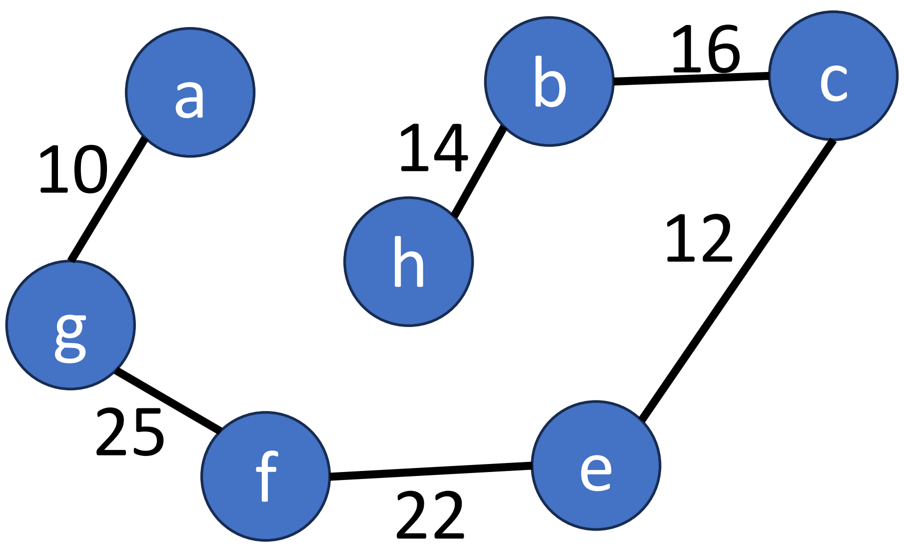
- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

  ....

- Grow the tree until it has n – 1 edges (or n vertices).

Prim’s method





- Start with a single-vertex tree.
- Grow the tree to two vertices by adding a least cost edge.
- Grow the tree to three vertices by adding a least cost edge.

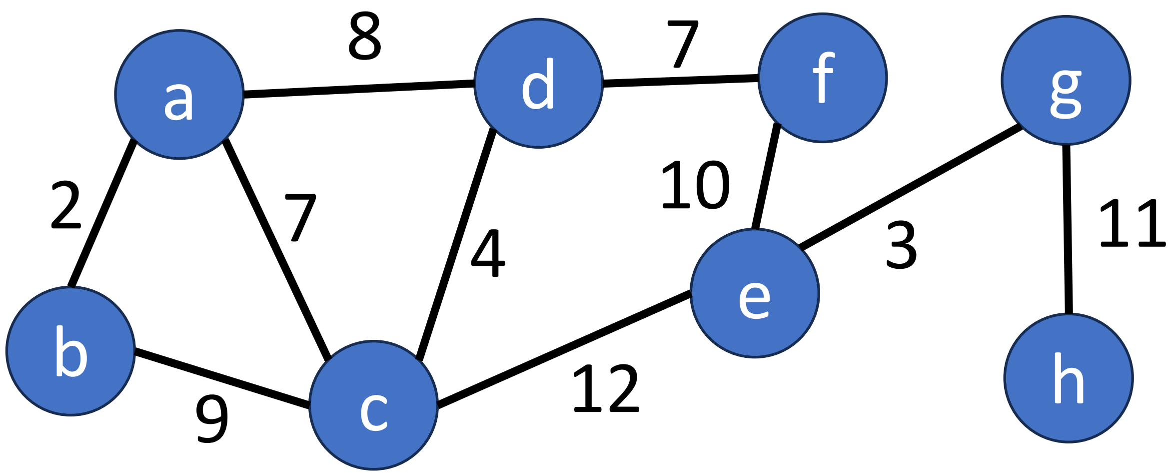
  ….

- Grow the tree until it has n – 1 edges (or n vertices).

Exercise

Please reply your answers of Q3-Q4 via the following link:

Group members: 1~3 people



Write out the sequence of edges in the generation of minimum-cost spanning tree:

- Q3: When Kruskal's algorithm is used.
- Q4: When Prim's algorithm is used. (Start from vertex a)

- Q5: The cost of minimum cost spanning tree.

Prove that Kruskal algorithm can generate minimum-cost spanning tree.

Check Theorem 6.1 in the textbook

a) Kruskal's method produces a spanning tree whenever a spanning tree exits.
   - Only the edges create cycles are discarded.
   - Deletion of a single edge from a cycle still forms connected graph.
   - Initially, the graph G is a connected graph…

b) The spanning tree generated is of minimum cost.
   - Let U be a minimum cost spanning tree.
   - Prove that the cost of Kruskal's spanning tree T equals the cost of U.
   - Assume that k edges in T are not in U, …

Summary

- Minimum cost spanning tree
- Kruskal’s algorithm
- Prim’s algorithm