Graph theory

Trees and Arborescence

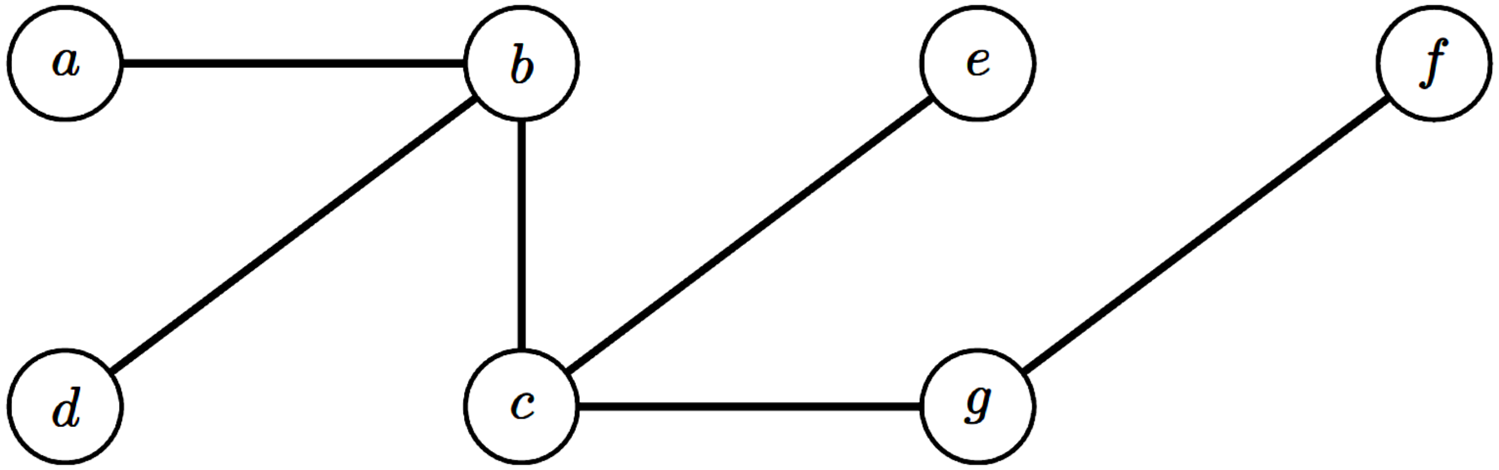
Z.GUELLIL

Trees and Arborescence

- Introduction to Tree
- Trees and Spanning Trees
- Minimum Spanning Trees (MST)
- MST Algorithms
    - Prim's Algorithm
    - Kruskal's Algorithm

Trees

- Definition: A tree is a connected undirected graph without a cycle (acyclic).
- A forest is a graph in which each connected component is a tree.

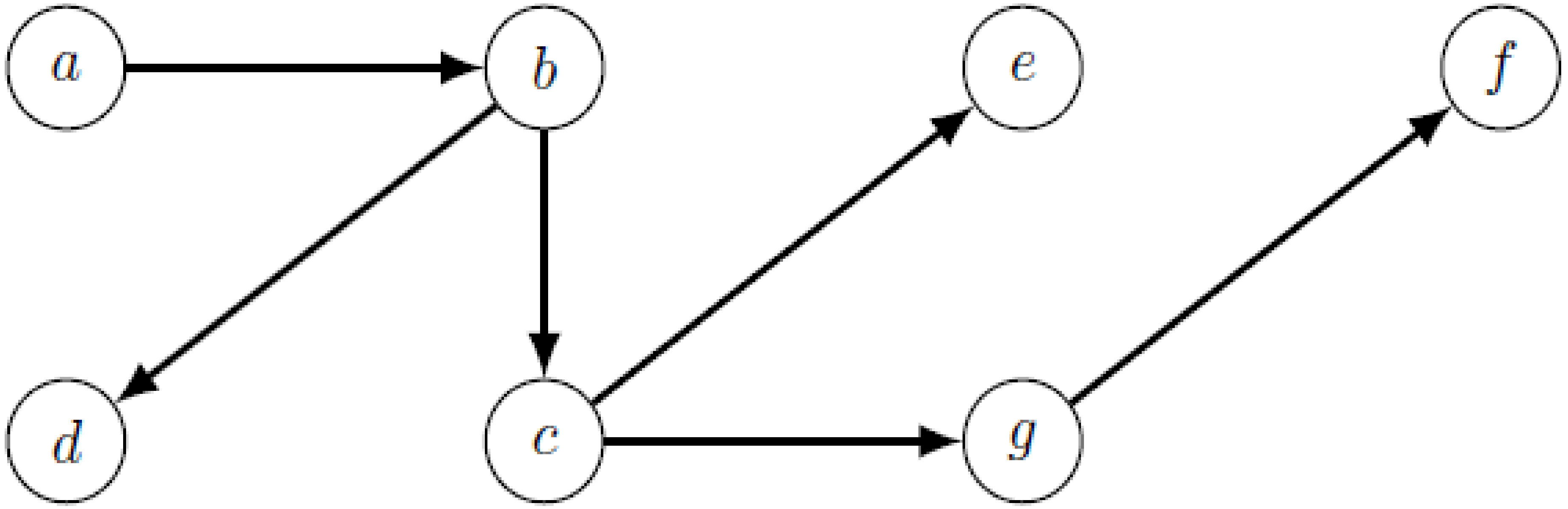


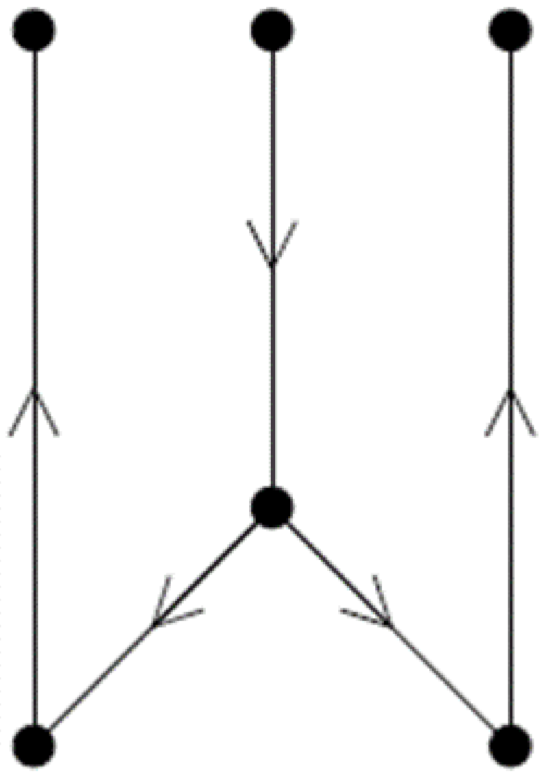
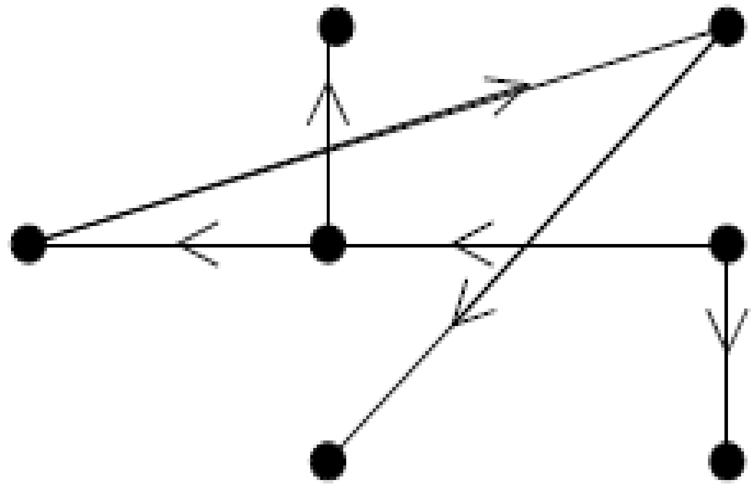
Properties and characterizations of trees

- G is cycle-free and has n-1 edges
- G is connected and has n-1 edges
- G is cycle-free, and by adding an edge, we create one and only one elementary cycle,
- G is connected, and by removing any edge, it is no longer connected,
- There exists one and only one chain between any 2 vertices of G.

Arborescence

- An arborescence is a directed acyclic graph with a root vertex $s_0 \in S$ such that for any other vertex $s_i \in S$, there exists a unique path from s_0 to s_i .
- If the arborescence has n vertices, then it has exactly n-1 arcs.



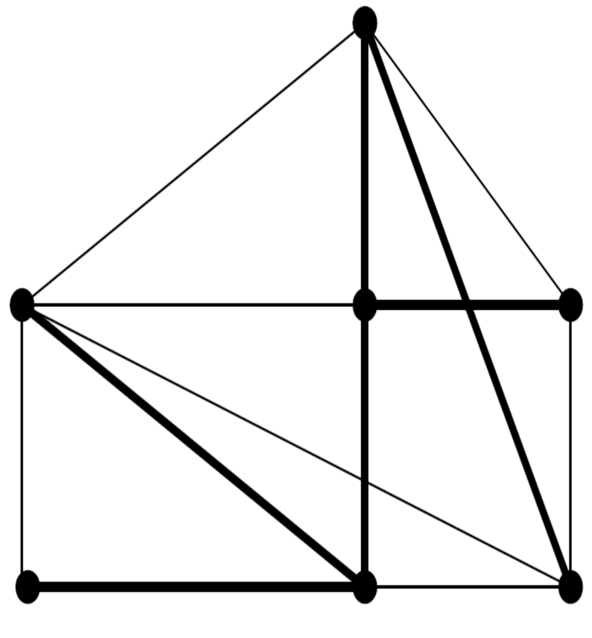



Spanning tree

- A spanning tree or covering tree is any tree containing all the vertices of the graph.
- Given a graph G = (V,E), a spanning tree T = (V,E') is a subgraph of G where:
  - T contains all vertices of G (V)
  - E' ⊆ E (edges of T are subset of G's edges)
  - T is a tree (connected and acyclic)
- A spanning Tree T is:
  - Minimal connected subgraph (removing any edge breaks connectivity)
  - Maximal acyclic subgraph (adding any edge creates a cycle)

Properties

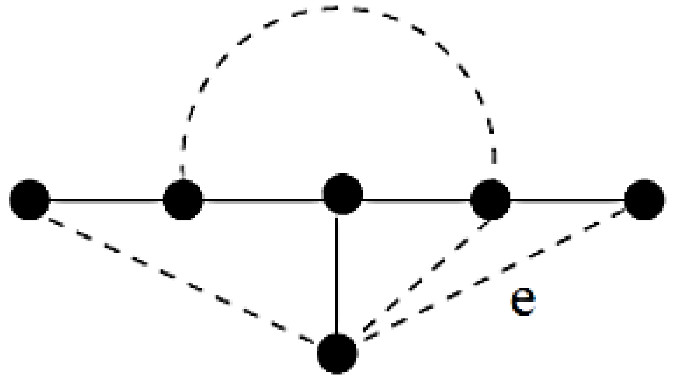
- Every connected graph has a spanning tree.
- A connected graph, there can be several non-isomorphic spanning trees.
- A complete graph with *n* vertices has = $n^{(n-2)}$ spanning trees (Cayley's Formula)
- For an unconnected graph, we speak of a spanning forest.
- Spanning trees can be obtained by DFS and BFS searches.



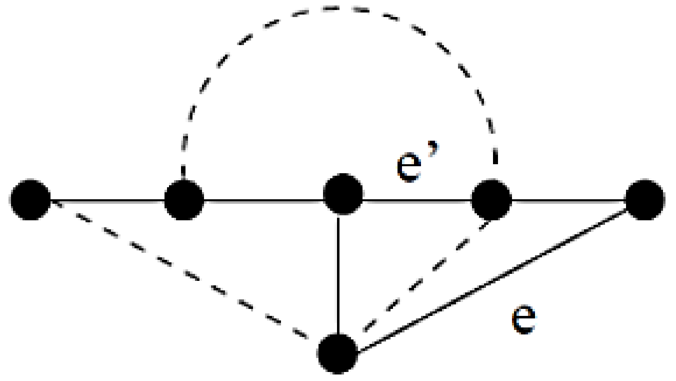
Properties

- A partial graph of a connected graph G is a spanning tree of G if and only if it is connected and minimal with respect to this property concerning the removal of edges.
- A partial graph of a connected graph G is a spanning tree of G if and only if it is acyclic and maximal with respect to this property concerning the addition of edges.

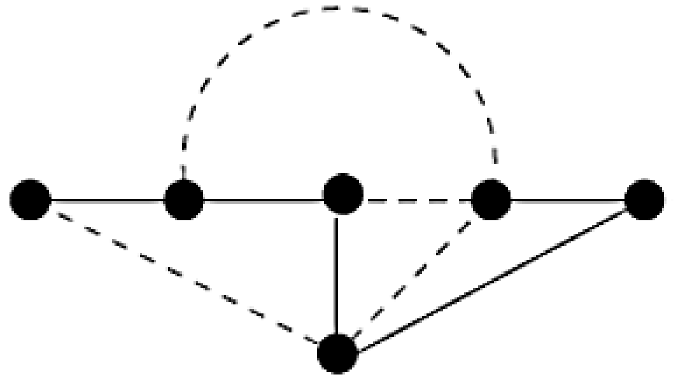
- Let T be a spanning tree of G and *e* an edge of G not belonging to T. The partial graph T + *e* contains a unique elementary cycle.
- Let T be a spanning tree of G and *e* be an edge of G not belonging to T.
- Let *e*′ be an edge of the cycle of T + *e*. Then T + *e* − *e*′ is a spanning tree of G not necessarily isomorphic to T.



T



T+e



T+e-e'

Minimum spanning tree

- A Minimum Spanning Tree (MST) is a spanning tree T of a weighted, connected graph G where the sum of edge weights in T is minimal among all possible spanning trees.
- If all edge weights in a graph are distinct, the MST is unique.
- If weights are not distinct, multiple MSTs may exist, but they will share the same set of edge weights.

cost function

- A cost function on the edges of G,
  - C : E(G) → R, is the assignment of a value c(<u>e</u>) to each edge <u>e</u> of G.
- Let F be any subset of edges of G.
- We define the <u>capacity</u> of F as the minimum cost of an edge of F.
- <u>Cost</u> of F, denoted c(F), as the sum of the costs of the edges of F:

$$C(F) = \sum_{e \in F} C(e)$$

- <u>Cost</u> of graph G, denoted c(G), is equal to the cost of the set E of its edges.

cost function

- Given: Graph $G = (V, E, w)$ where w are edge weights MST T
- Total weight $W(T) = \sum w(e)$ for $e \in T$ is minimal

Kruskal’s Algorithm

- A greedy algorithm that builds the MST by adding edges in increasing order of weight while avoiding cycles.

1. **Sort all edges**: Arrange edges in ascending order of weight.
2. **Initialize subsets**: Use a disjoint set (union-find) data structure to track connected components for each vertex.
3. **Iterate through edges**:
   - For each edge, check if it connects two vertices in different subsets (i.e., no cycle is formed).
   - If it does, add this edge to the MST.
4. **Repeat until** complete: Continue adding edges until the MST has $n-1$ edges, where n is the number of vertices.

Kruskal’s Algorithm

```
procedure Kruskal (G: graph)
Begin
    $T \leftarrow \emptyset$
    $F \leftarrow E$
    Wihle ($|T| < n - 1$) do
        find an edge $e \in F$ of minimal weight
        $F \leftarrow F - e$
        if $T + e$ is acyclic then $T \leftarrow T \cup e$ End if
    End While
End.
```

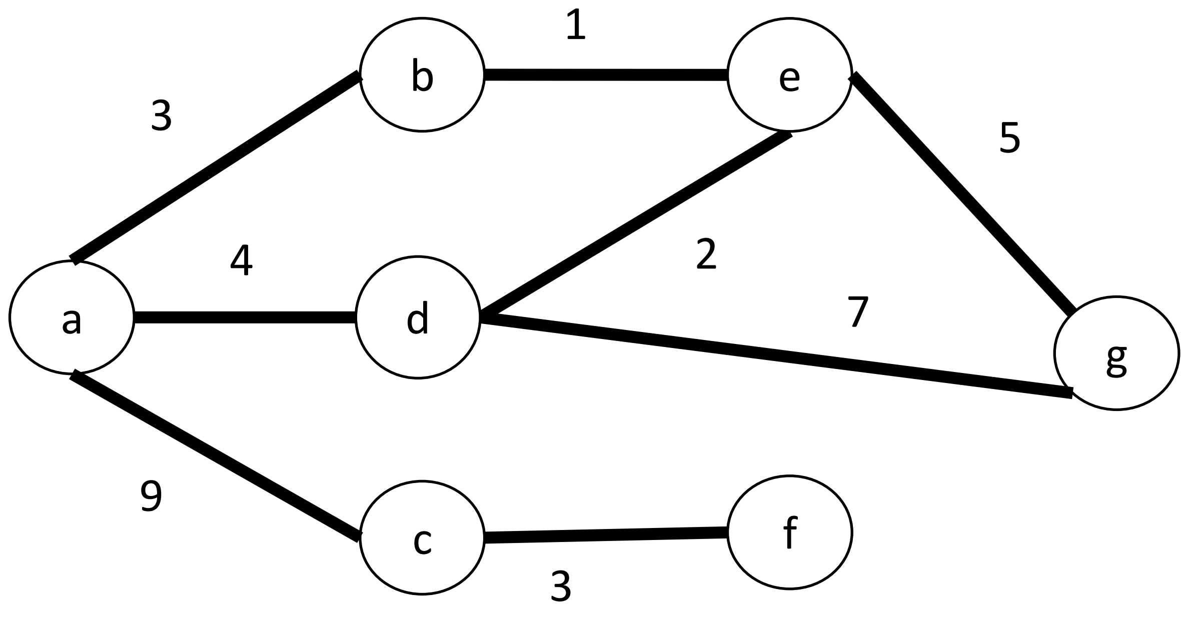
Remark:

1. This algorithm gradually "consumes" the selected edges: it is called a greedy algorithm.
2. When several edges have the same weight, the order in which they are taken is indifferent.
3. The same algorithm applies to finding a maximally weighted spanning tree: We sort the edges in order of decreasing weight.

Example

1. Sort arcs in ascending order of weight

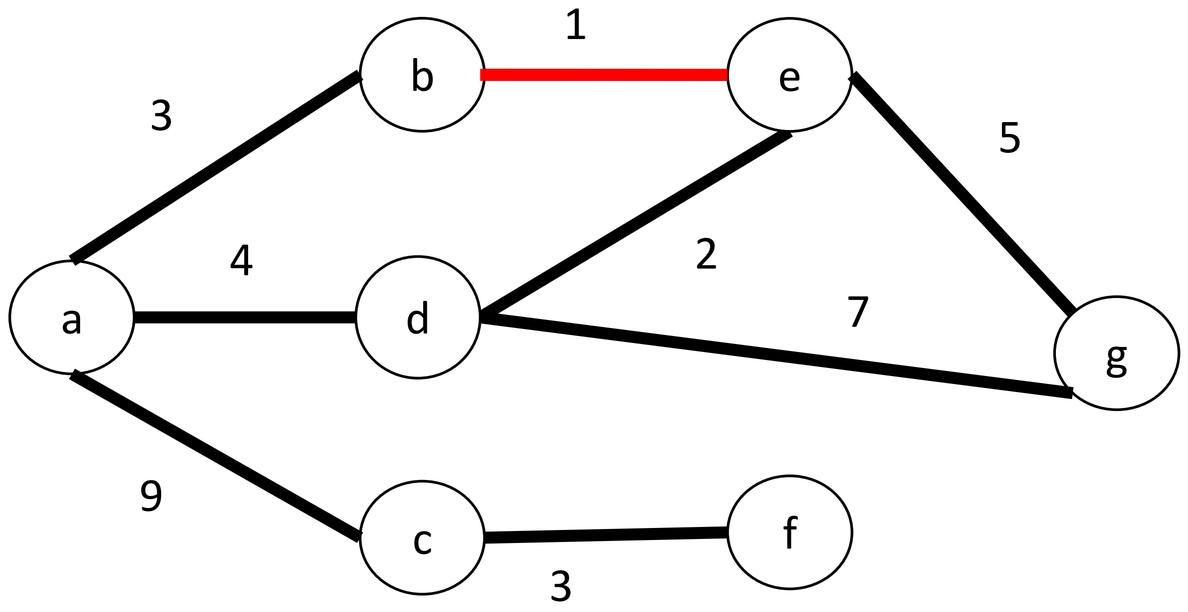
F	T
(b, e) 1	
(d, e) 2	
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

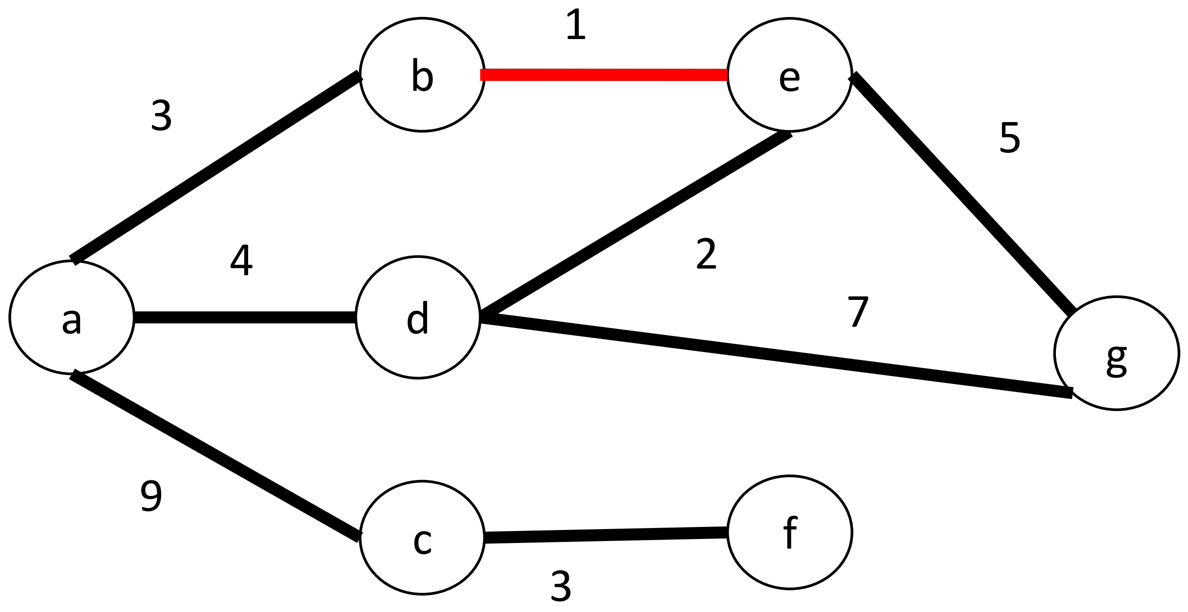
F	T
(b, e) 1	
(d, e) 2	
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

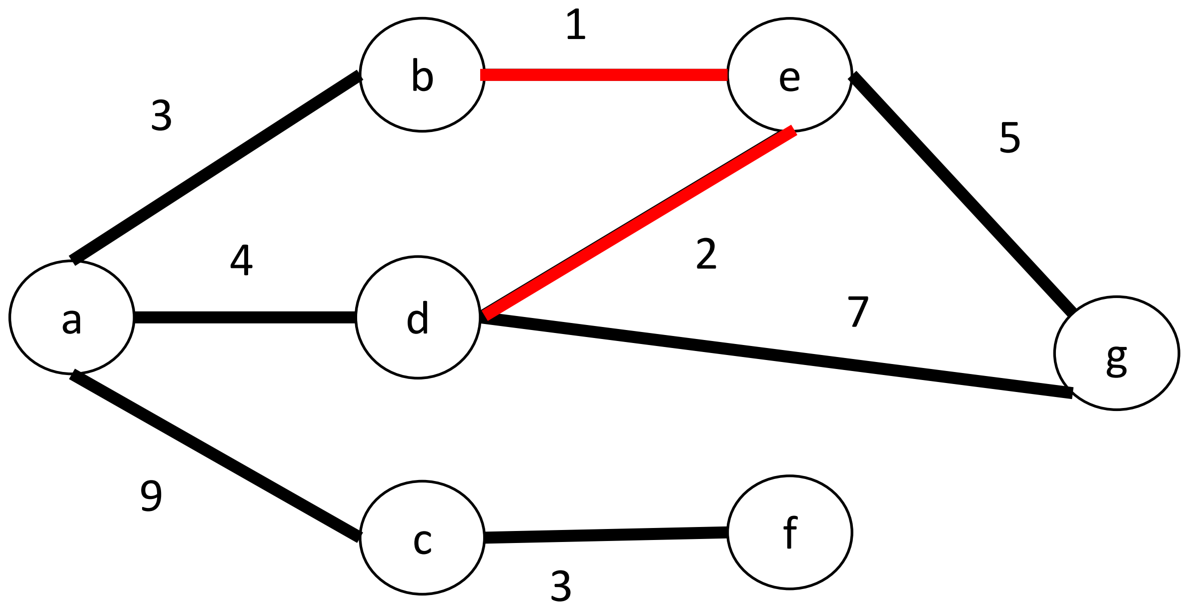
F	**T**
	(b, e) 1
(d, e) 2	
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

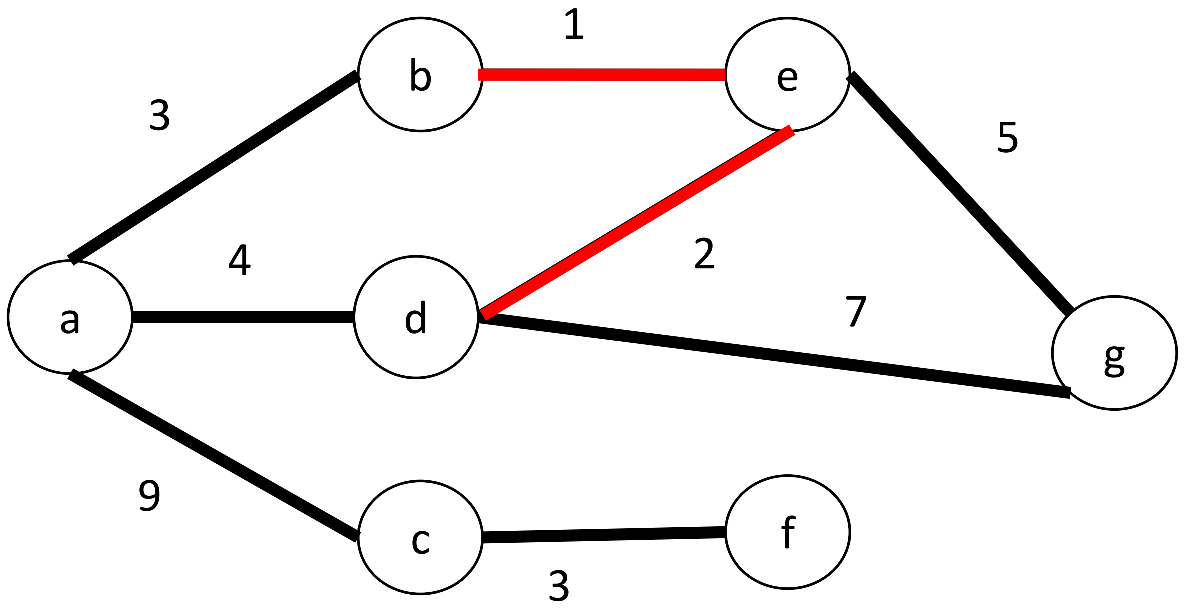
F	T
	(b, e) 1
(d, e) 2	
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

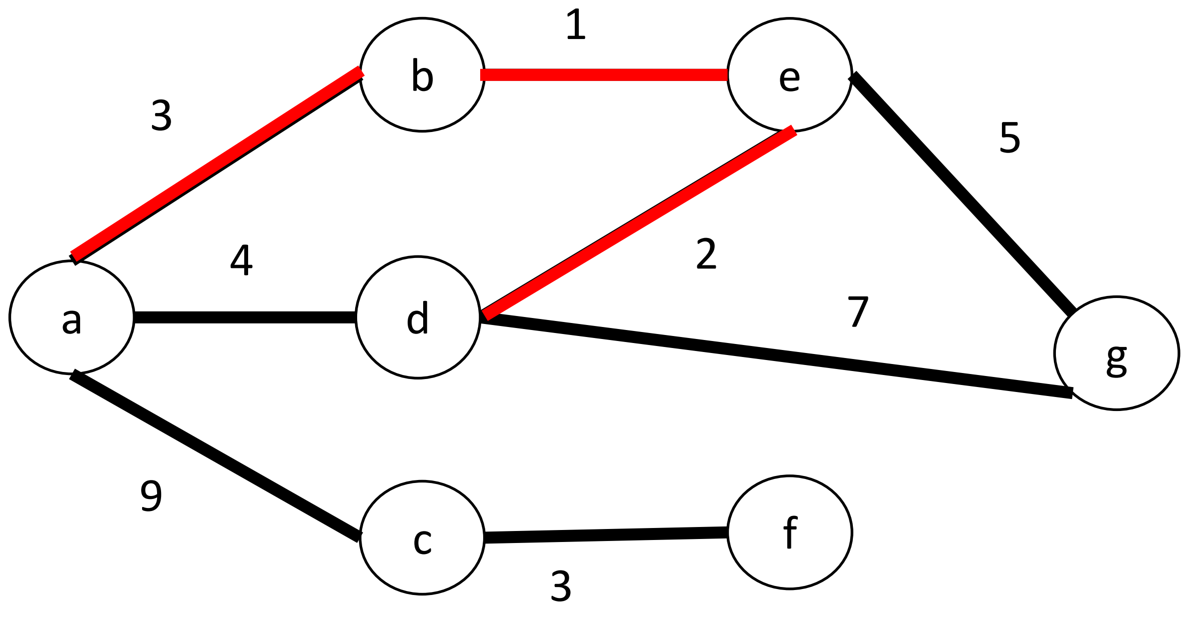
F	T
	(b, e) 1
	(d, e) 2
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

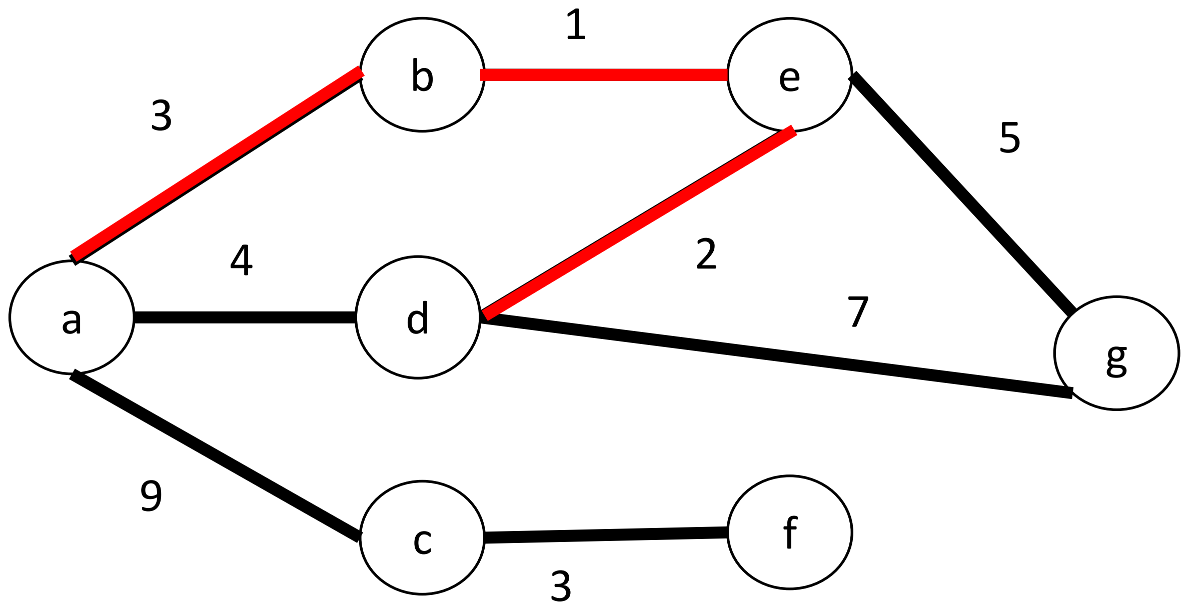
F	T
	(b, e) 1
	(d, e) 2
(a, b) 3	
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

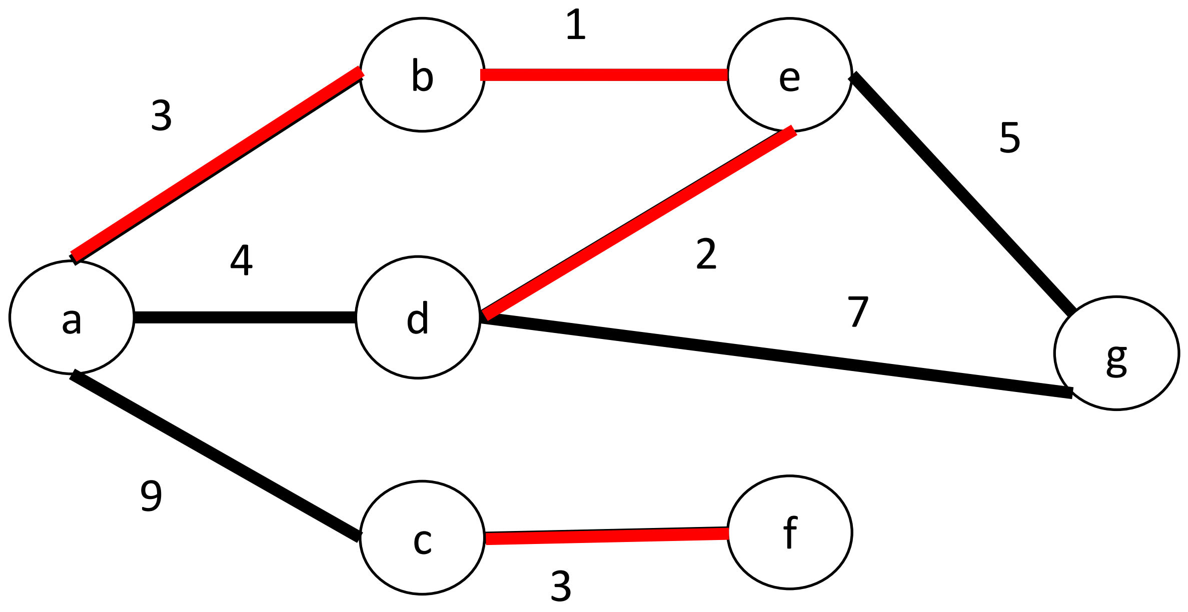
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

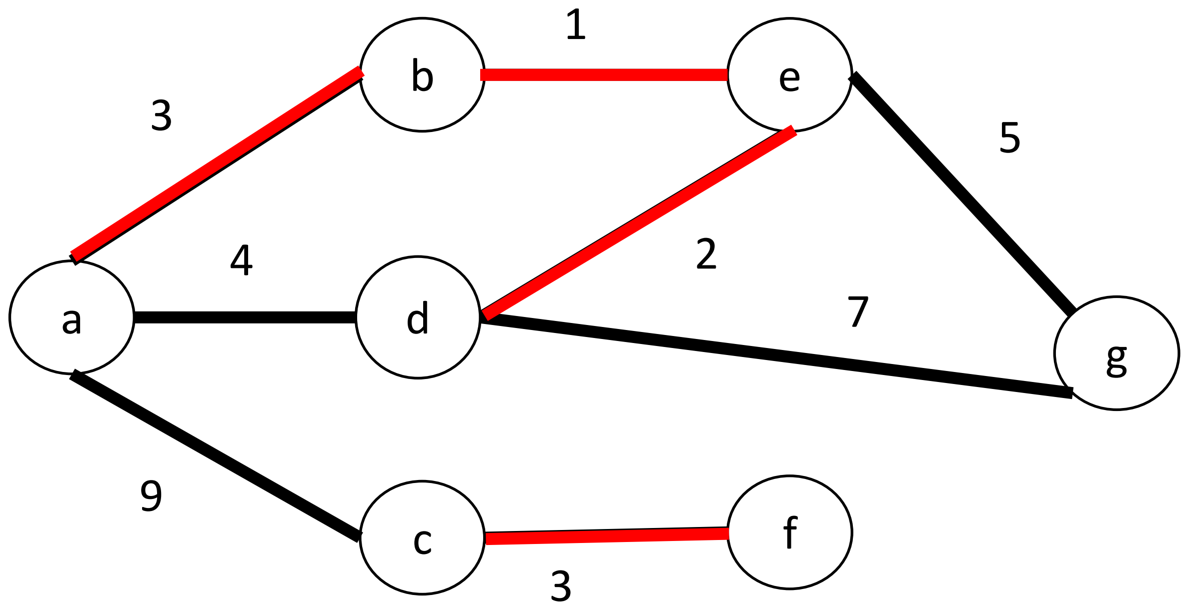
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
(c, f) 3	
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

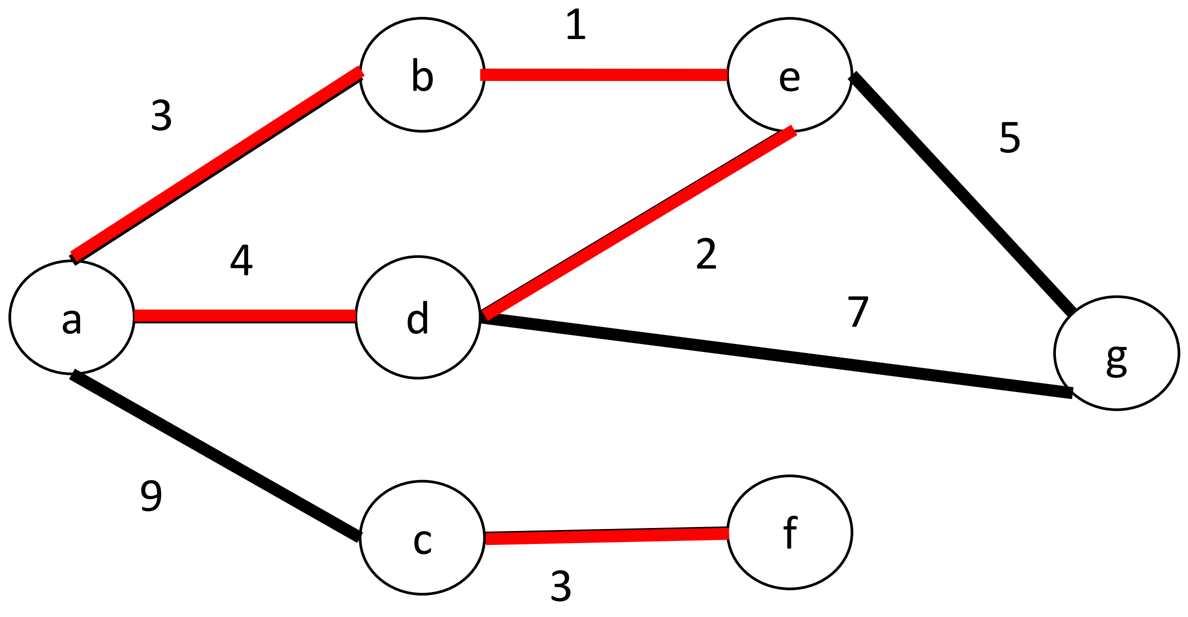
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

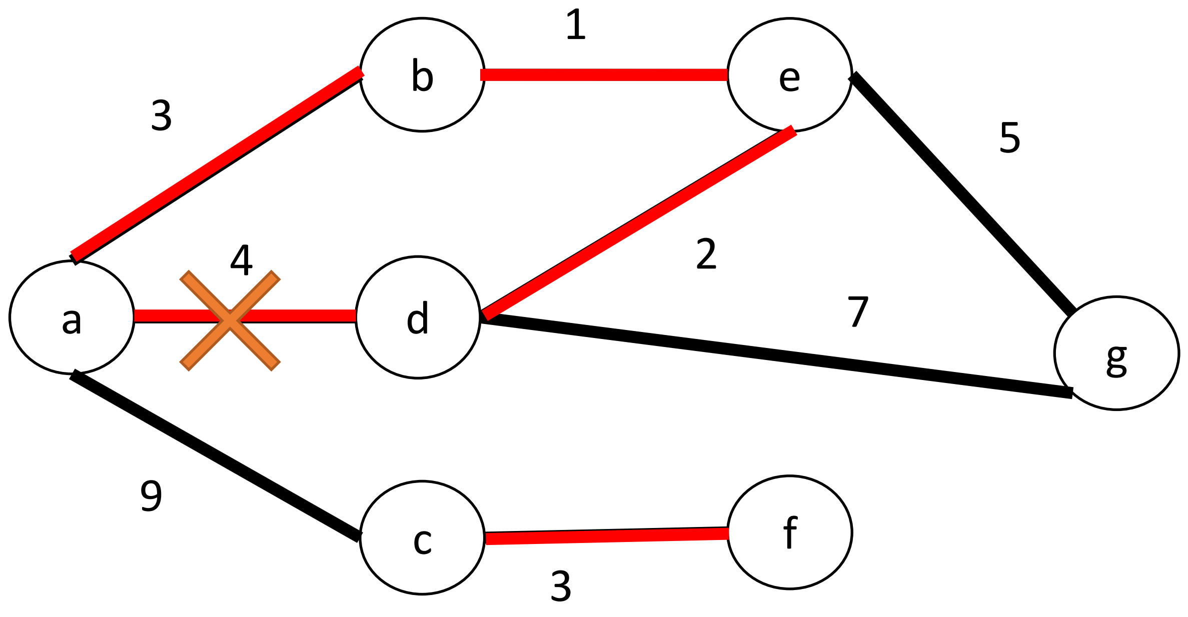
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
(a, d) 4	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

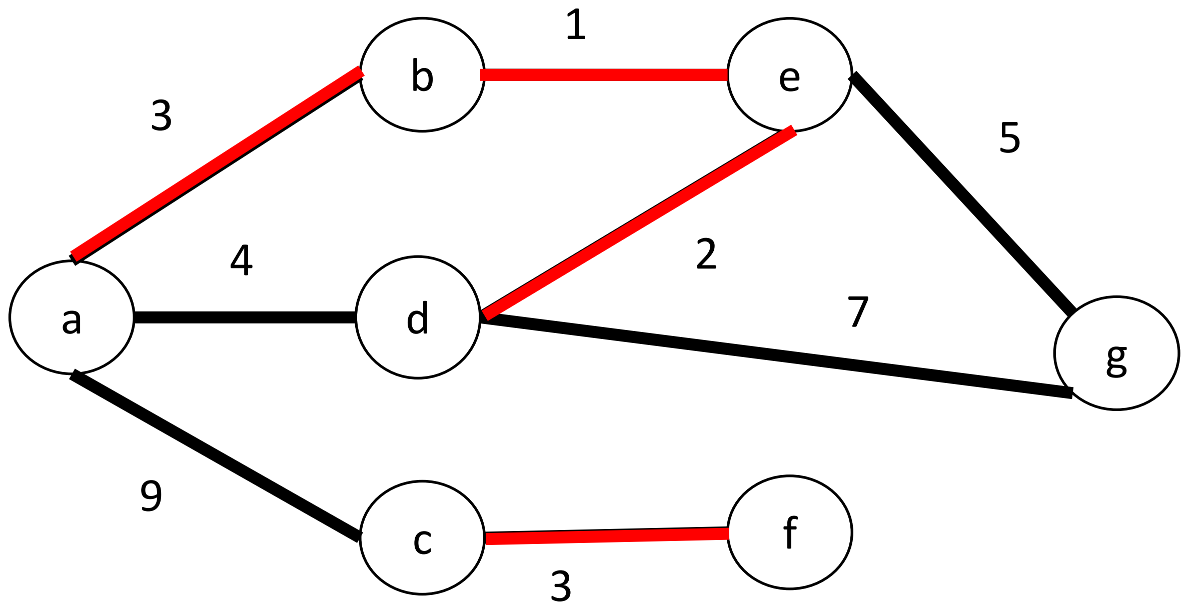
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

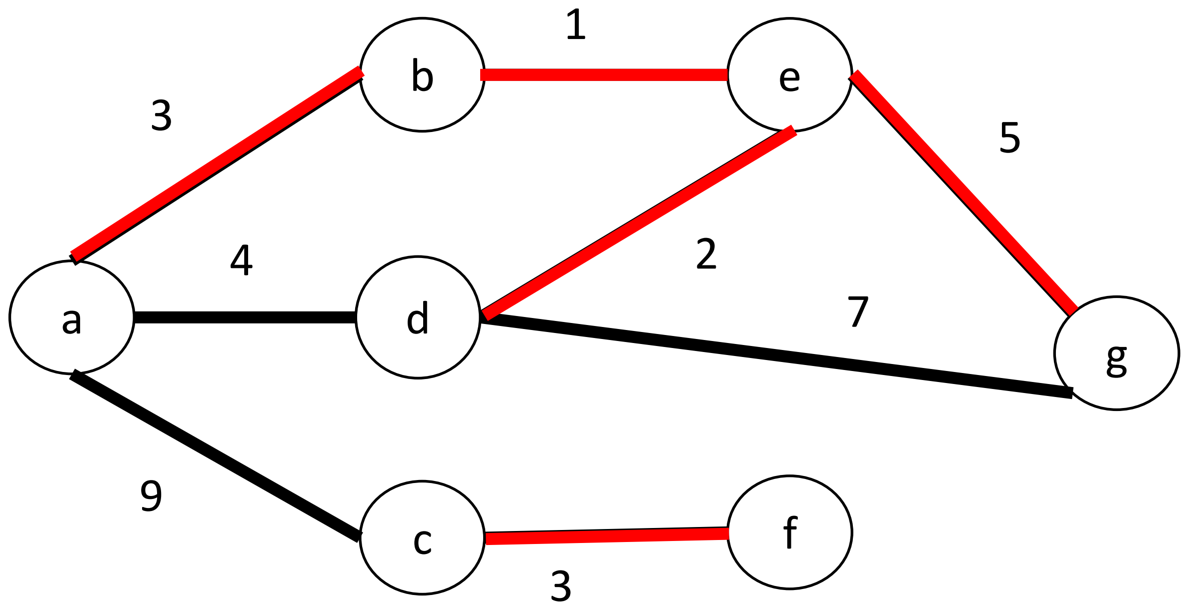
F	**T**
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

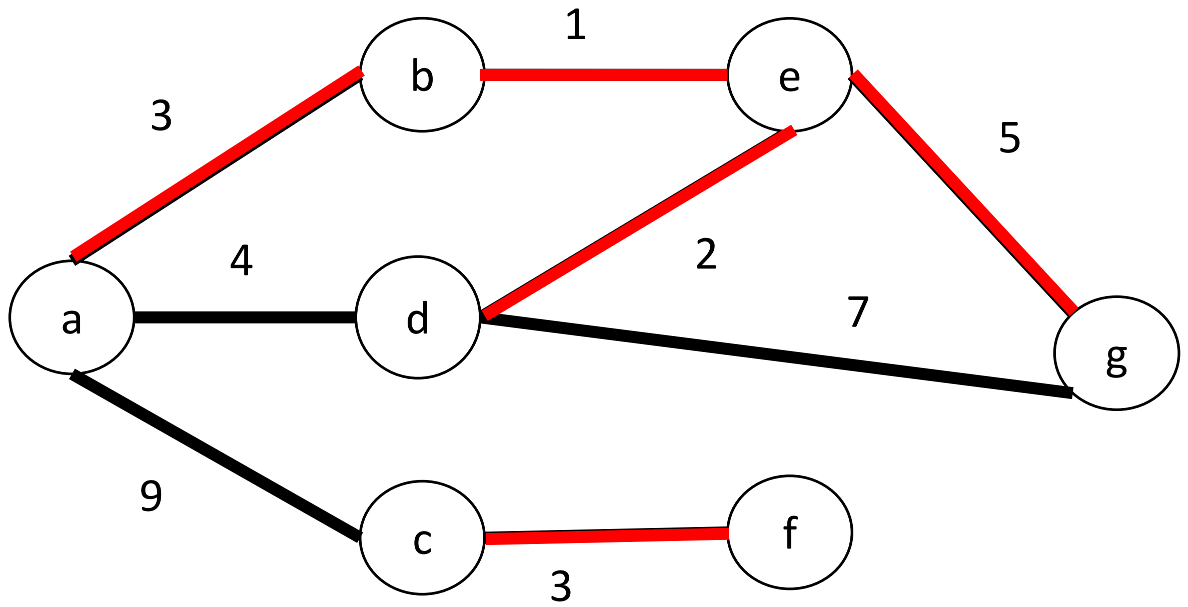
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
(e, g) 5	
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

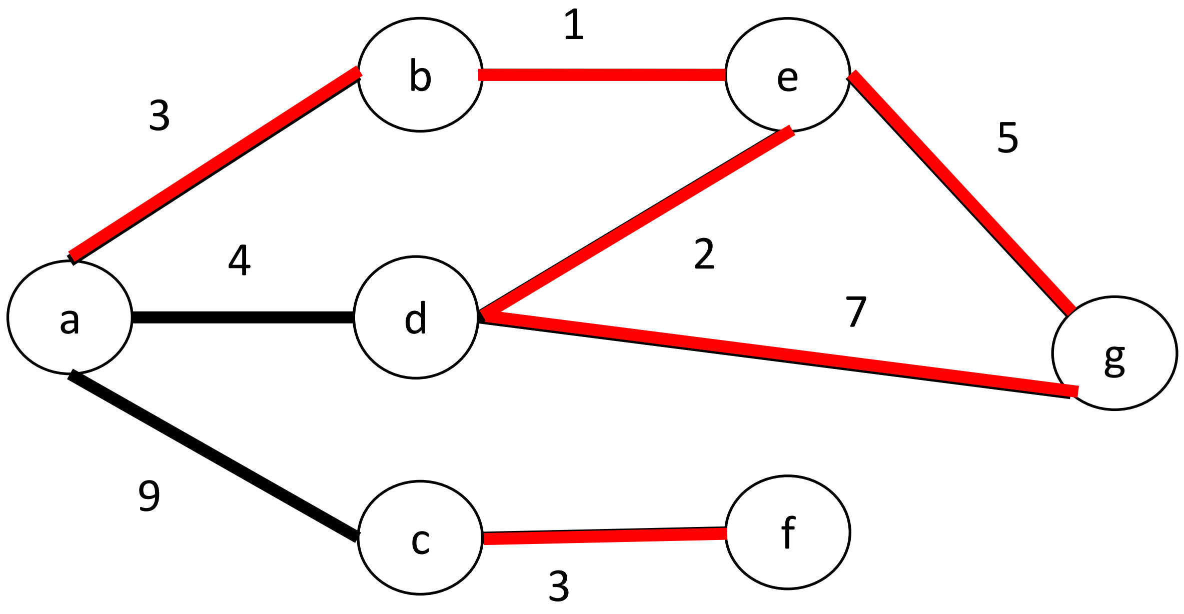
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
	(e, g) 5
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

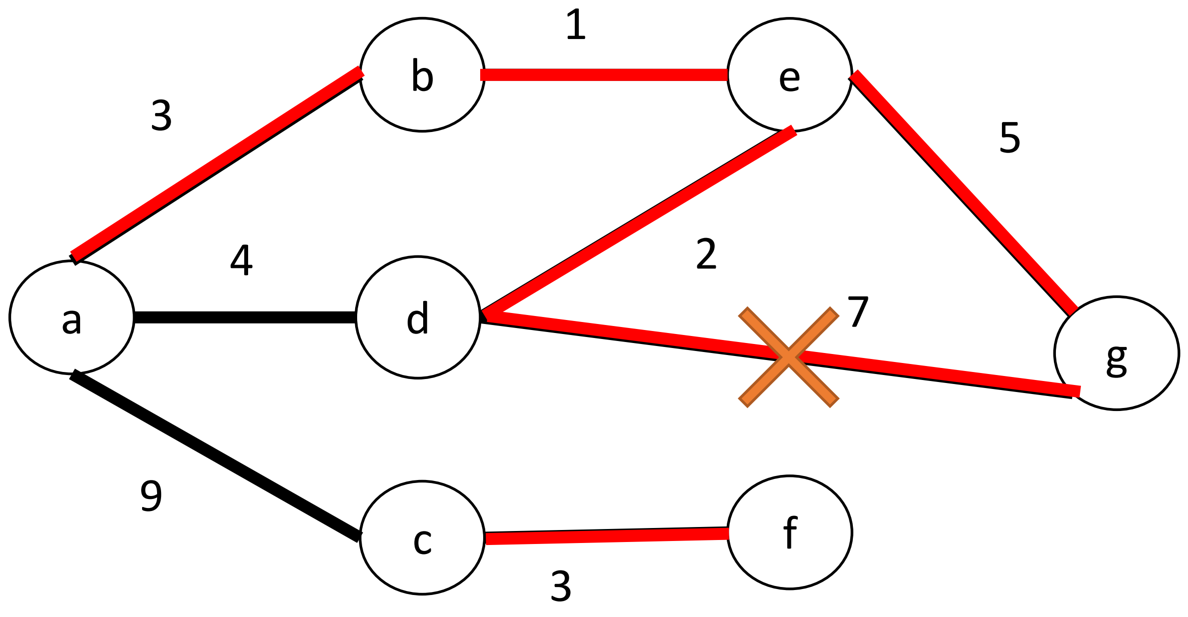
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
	(e, g) 5
(d, g) 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

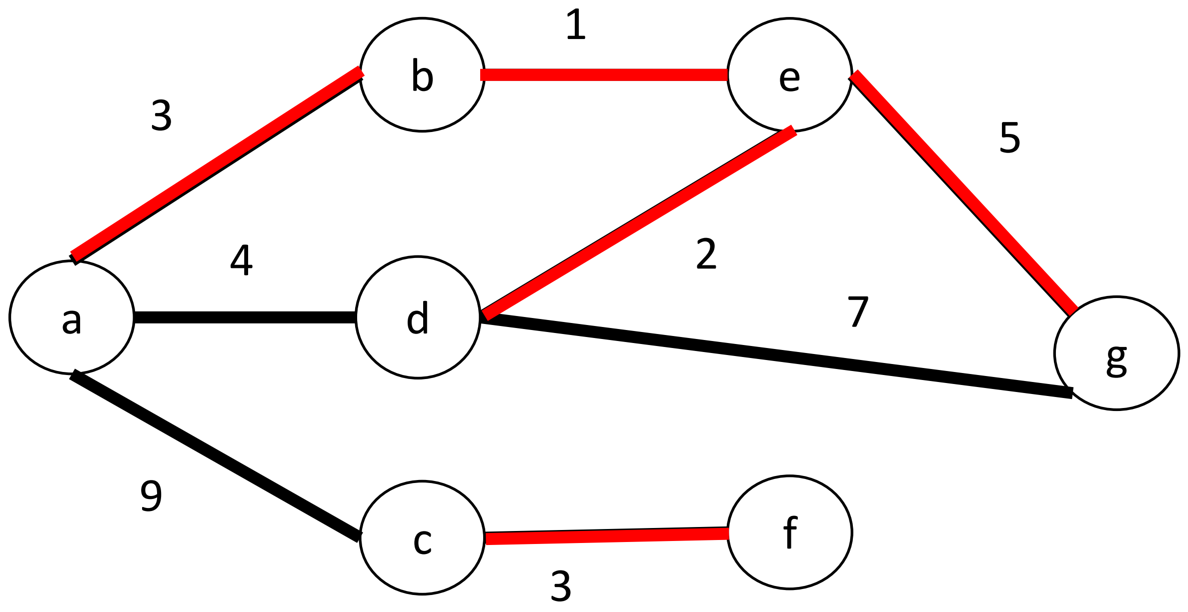
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
	(e, g) 5
~~(d, g) 7~~	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

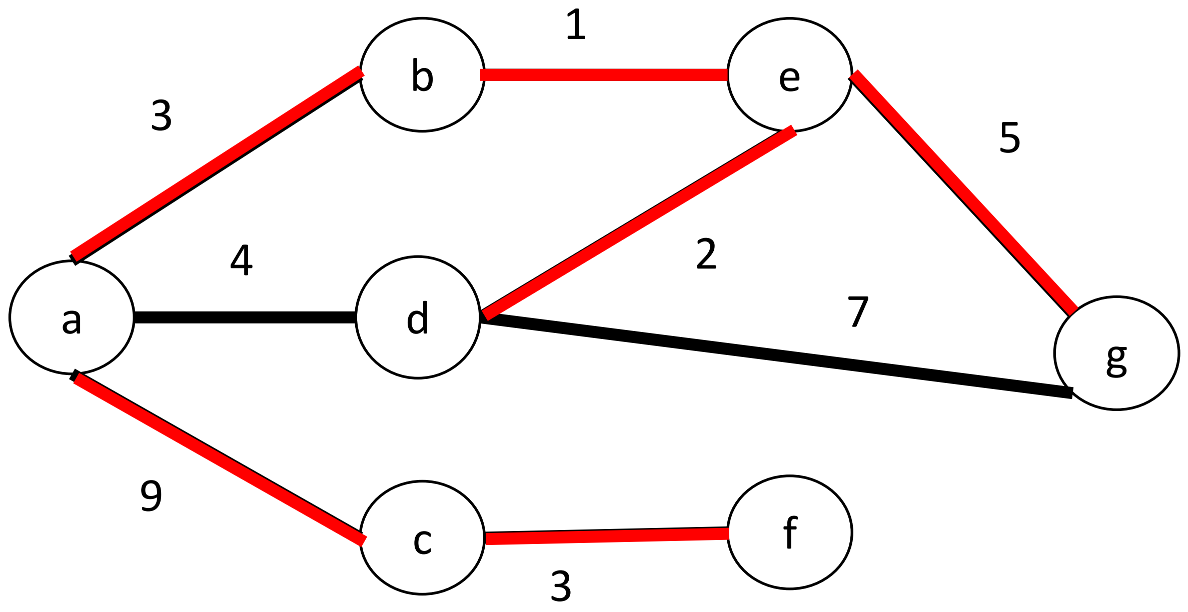
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d)~~ 4	
	(e, g) 5
~~(d, g)~~ 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

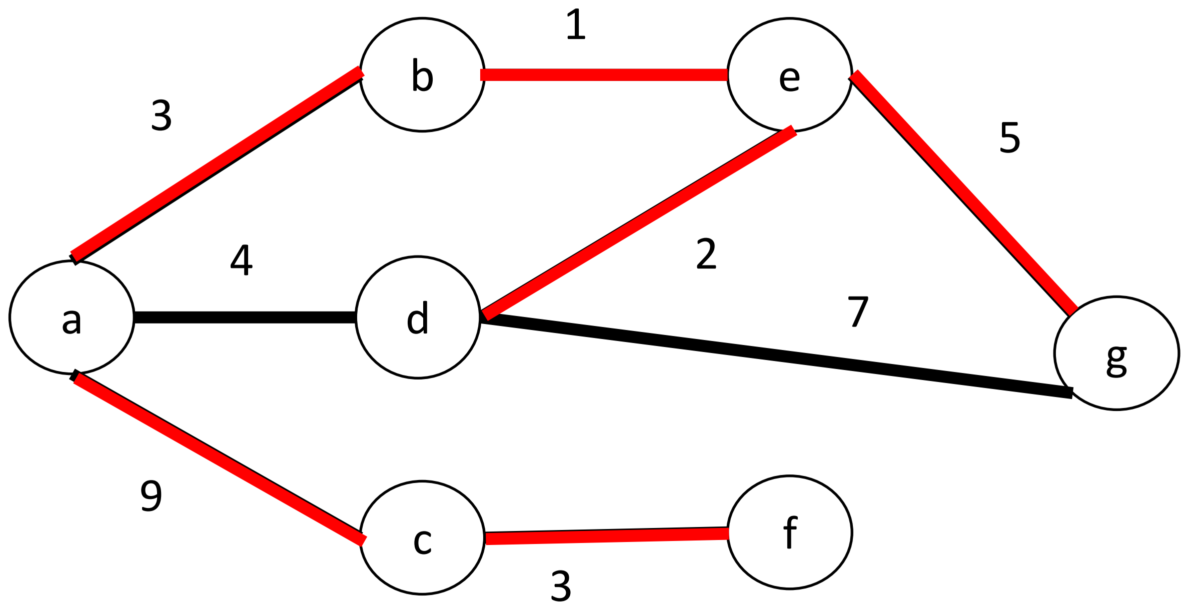
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d)~~ 4	
	(e, g) 5
~~(d, g)~~ 7	
(a, c) 9	



Example

1. Sort arcs in ascending order of weight

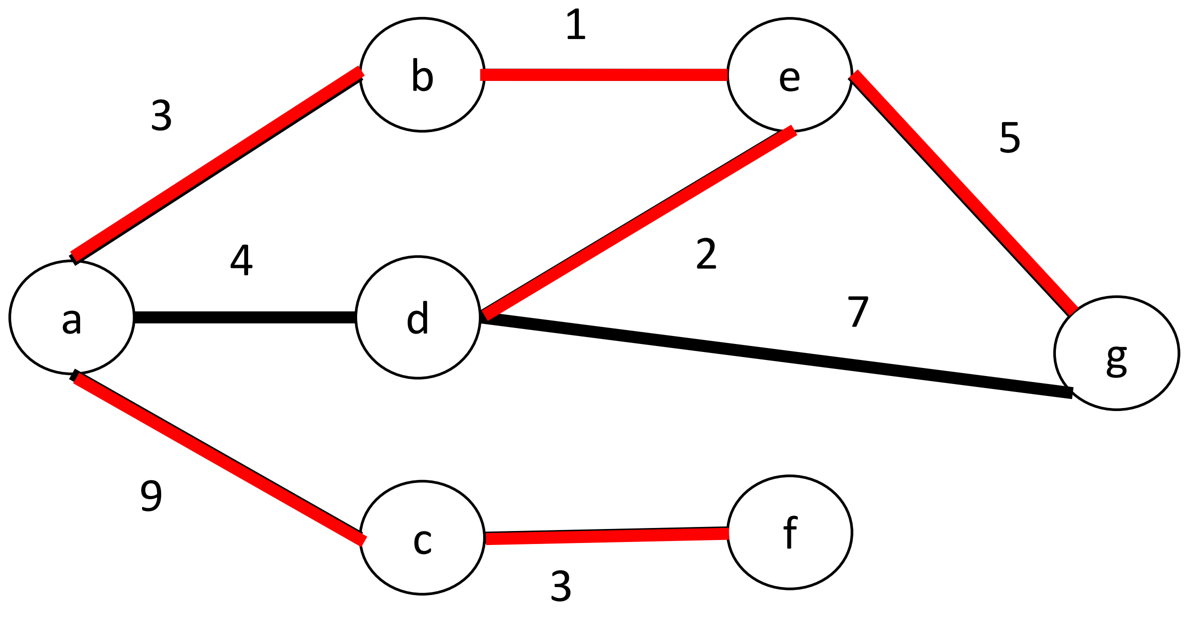
F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
	(e, g) 5
~~(d, g) 7~~	
	(a, c) 9



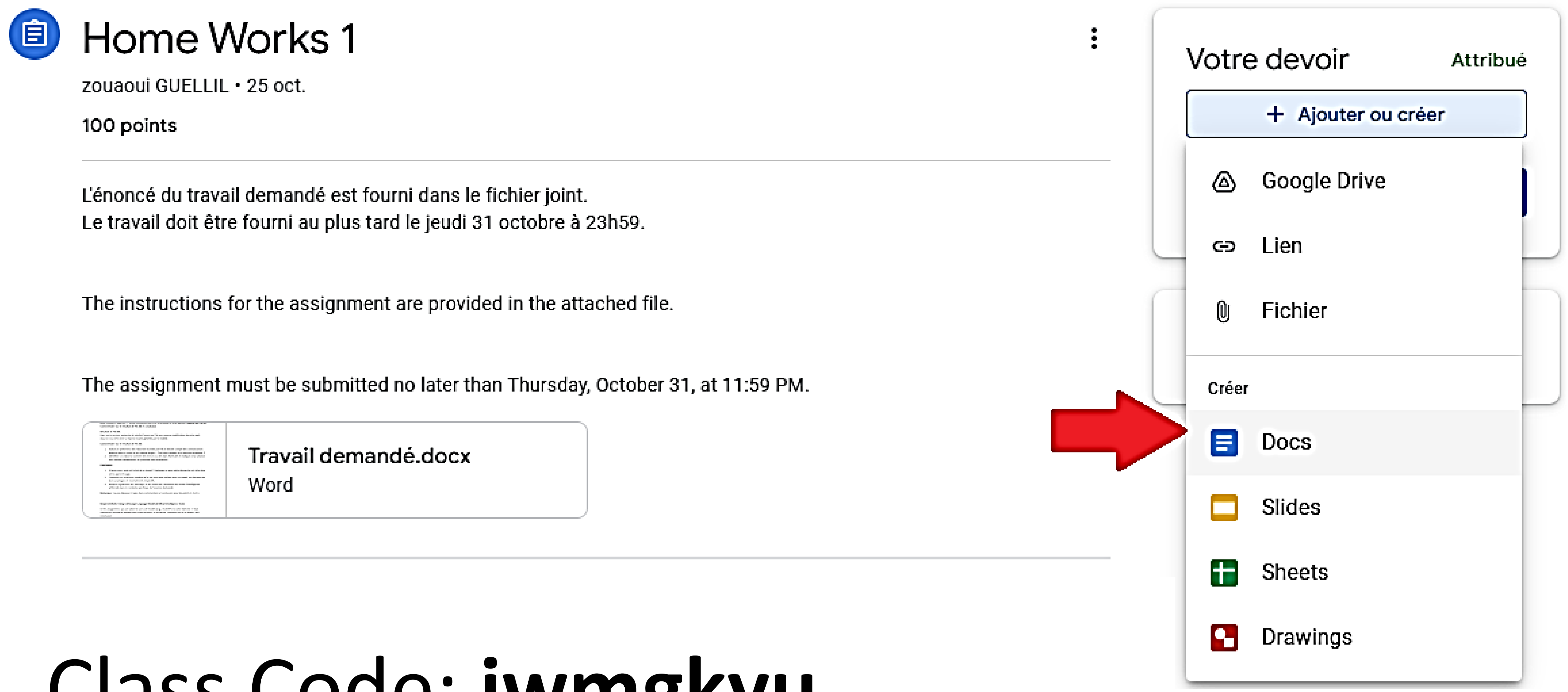
Example

1. Sort arcs in ascending order of weight

F	T
	(b, e) 1
	(d, e) 2
	(a, b) 3
	(c, f) 3
~~(a, d) 4~~	
	(e, g) 5
~~(d, g) 7~~	
	(a, c) 9



Weight = 1 + 2 + 3 + 3 + 5 + 9 = 23



Home Works 1

zouaoui GUELLIL • 25 oct.

100 points

L'énoncé du travail demandé est fourni dans le fichier joint.
Le travail doit être fourni au plus tard le jeudi 31 octobre à 23h59.

The instructions for the assignment are provided in the attached file.

The assignment must be submitted no later than Thursday, October 31, at 11:59 PM.

Travail demandé.docx
Word

Class Code: **iwmgkyu**

- **Résultat de ChatGPT :**
  Dans cette section, présentez le résultat fourni par l'IA sans aucune modification de votre part. Assurez-vous d'inclure la réponse exacte générée par le modèle.
- **Commentaire sur le résultat de ChatGPT :**
  1. Évaluez la pertinence des réponses fournies par l'IA en tenant compte des connaissances acquises dans le cours et les travaux dirigés. (. Êtes-vous satisfait de la solution proposée ?)
  2. Identifiez si la réponse contient des erreurs ou des biais éventuels et indiquez si la solution vous semble satisfaisante, en justifiant votre évaluation.

- **Résultat de copilot :**
- **Commentaire sur le résultat de copilot :**
- **Résultat d'une autre IA:**
- **Commentaire sur le résultat d'une autre IA :**
- ….
- **Conclusion :**

1. D'après vous, quel est le but de ce travail ? Expliquez en quoi cette démarche est utile dans votre apprentissage.
2. Comparez les différents modèles d'IA que vous avez utilisés pour ce travail, en discutant de leurs avantages et inconvénients respectifs.
3. Analysez également les avantages et les limites de l'utilisation des outils d'intelligence artificielle dans le contexte spécifique de l'exercice demandé.

Prim's Algorithm

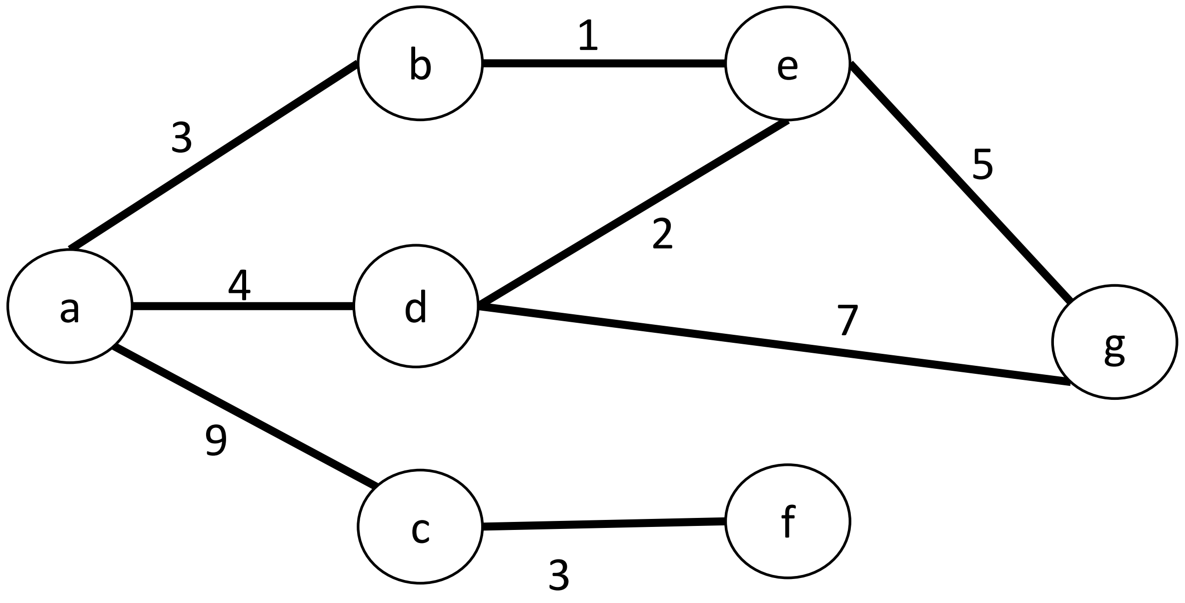
- **Kruskal’s Algorithm** ensures the acyclicity property of the tree, while **Prim’s Algorithm** relies on the connectivity of the tree.
- The minimum spanning tree is built by adding a new branch to the already constructed sub-tree T,
- Selecting one of the minimal-weight edges that connects a vertex of T to a vertex not yet in T.
- The algorithm stops when all the vertices of the graph are part of T.

- **T** is the set of edges in the minimum spanning tree.
- **S** is the set of vertices in **T**.

```
Procedure Prim(G: graph)
    Choose a vertex x from G.
    $S \leftarrow S \cup \{x\}$ // S will contain the vertices of T
    $T \leftarrow \emptyset$
    While ($S \neq X$) do:
        Find an edge (y, s) of minimum weight such that $y \in X - S$ and $s \in S$
        $T \leftarrow T \cup (y, s)$
        $S \leftarrow S \cup y)$
    End while
End procedure
```

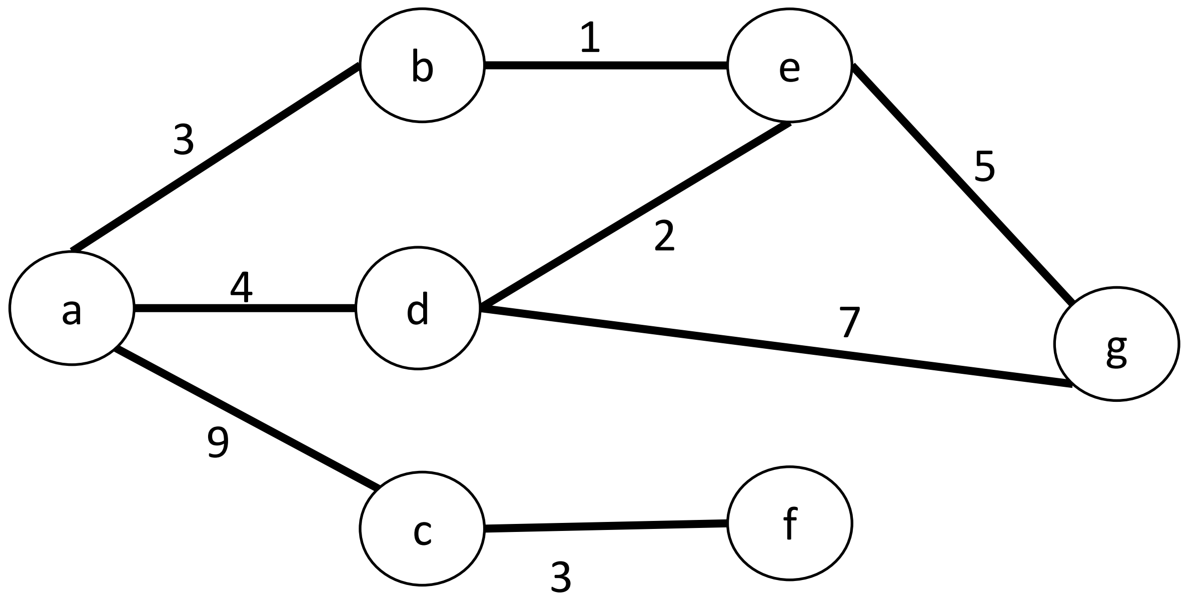
Example

S	T



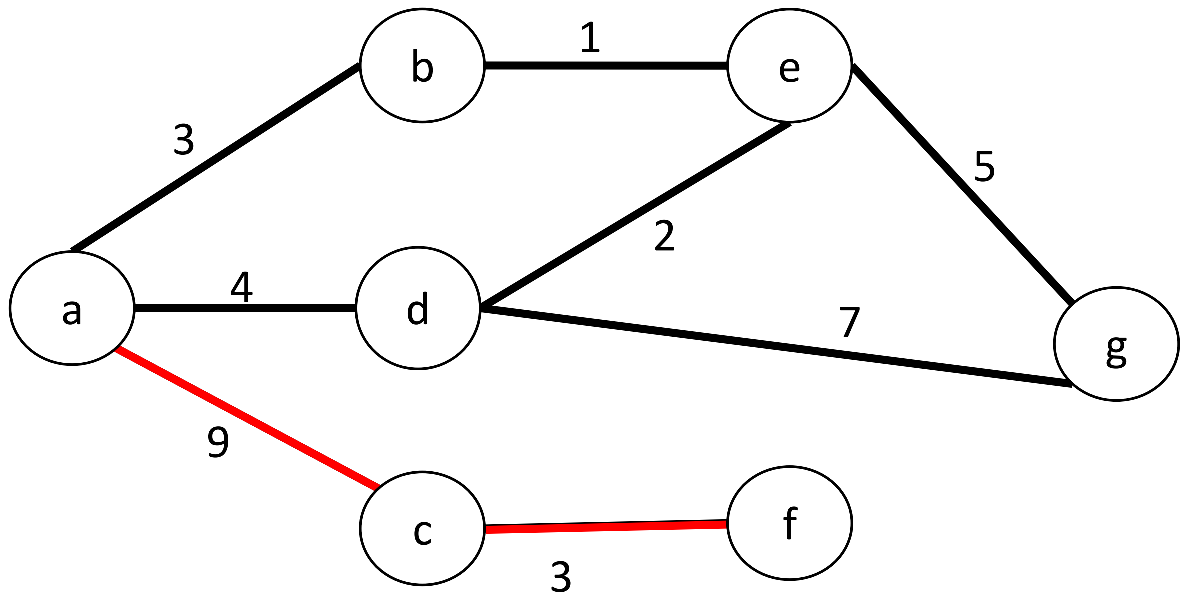
Example

S	**T**
c	



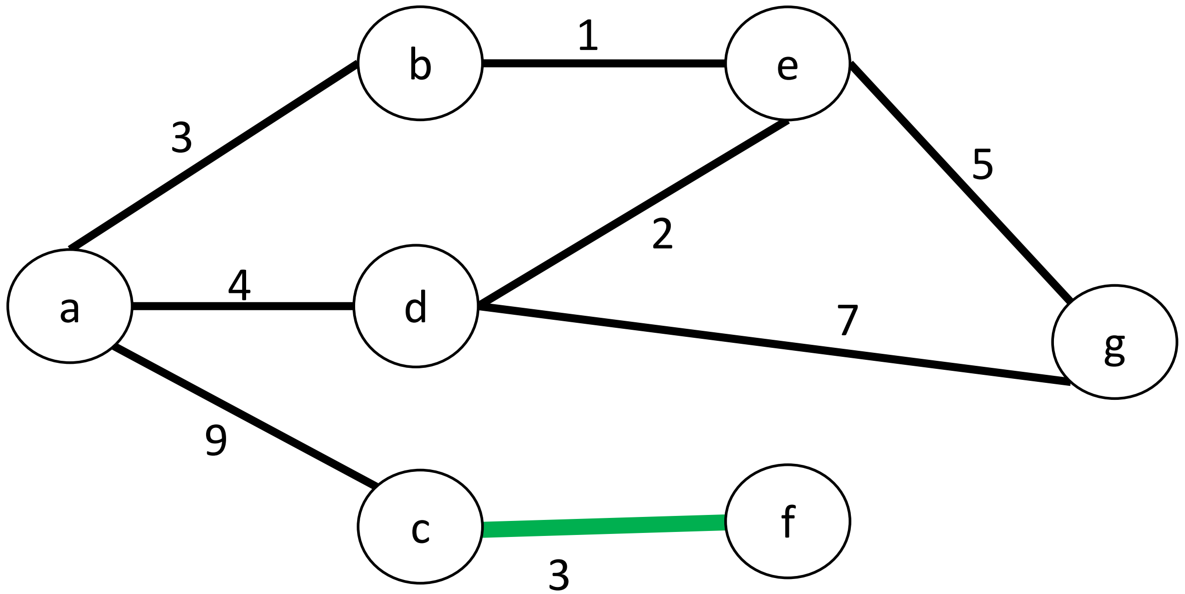
Example

S	T
c	



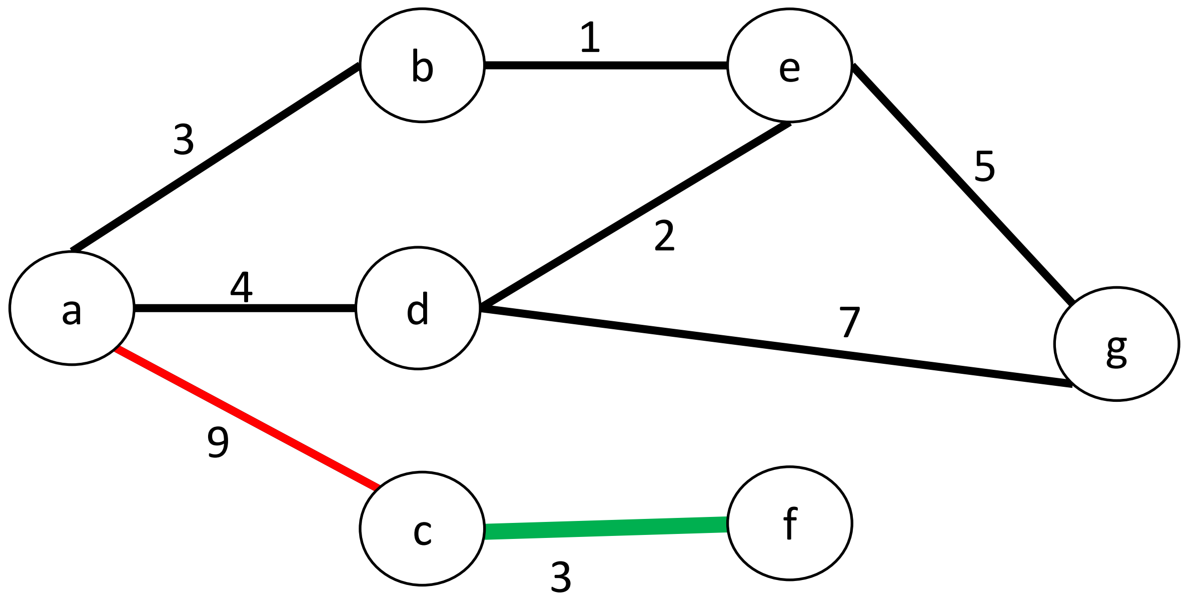
Example

S	T
c	
f	(c, f) 3



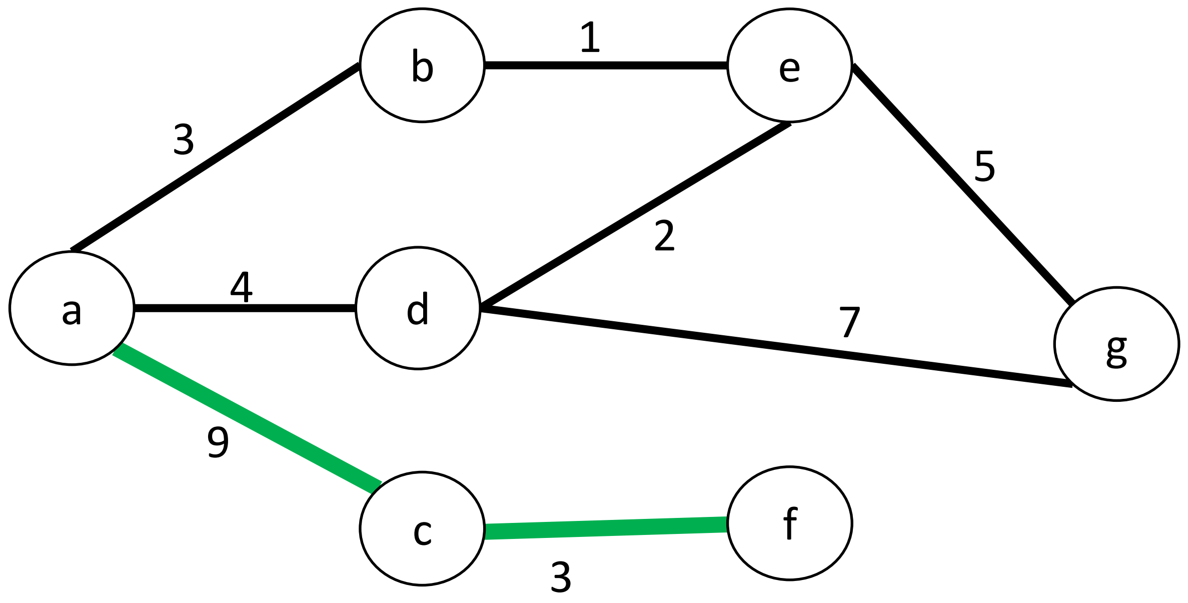
Example

S	T
c	
f	(c, f) 3



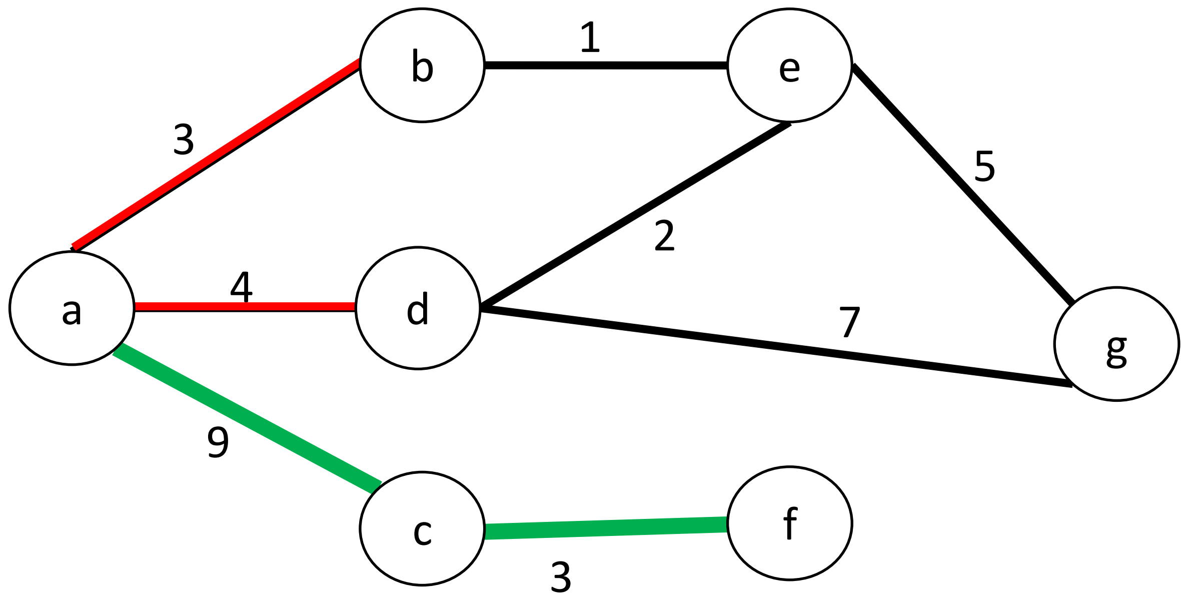
Example

S	**T**
c	
f	(c, f) 3
a	(a, c) 9



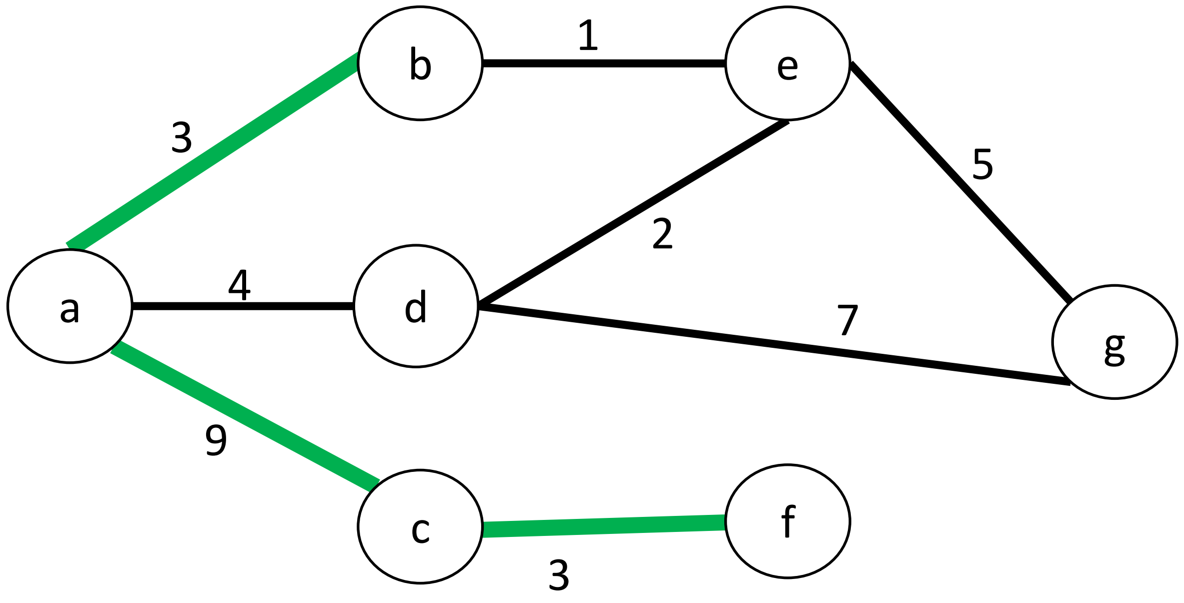
Example

S	T
c	
f	(c, f) 3
a	(a, c) 9



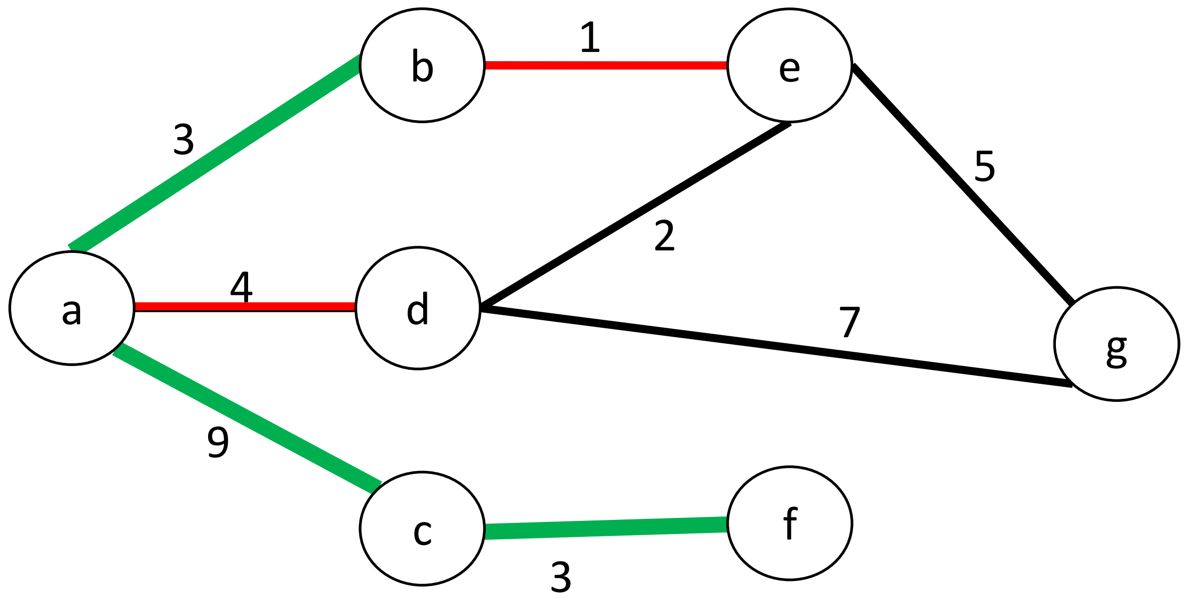
Example

S	T
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3



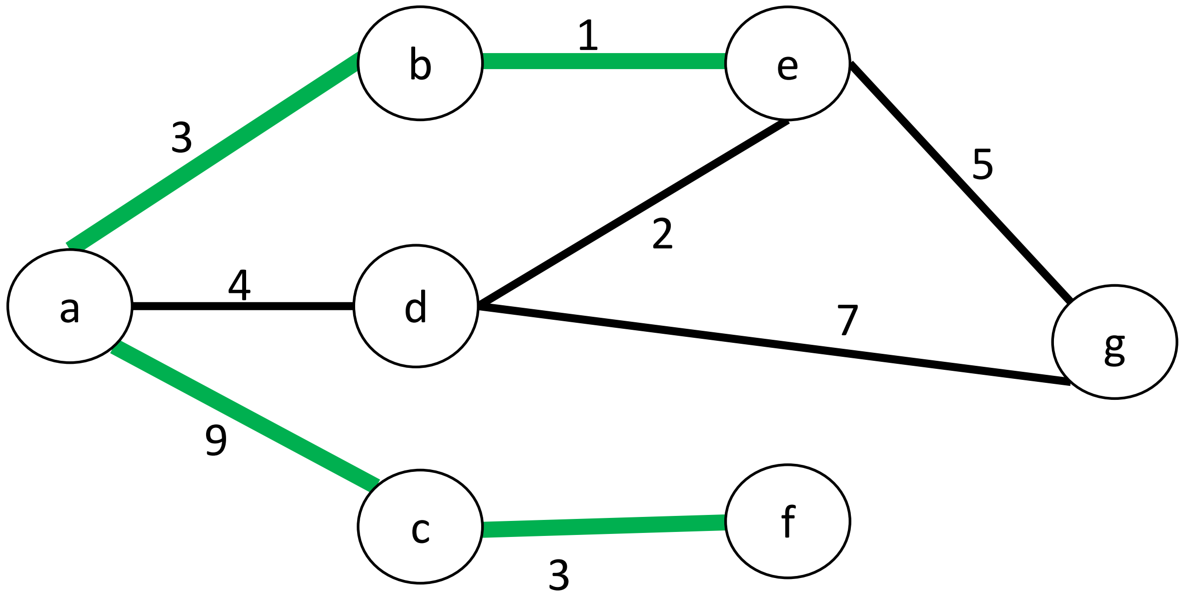
Example

S	T
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3



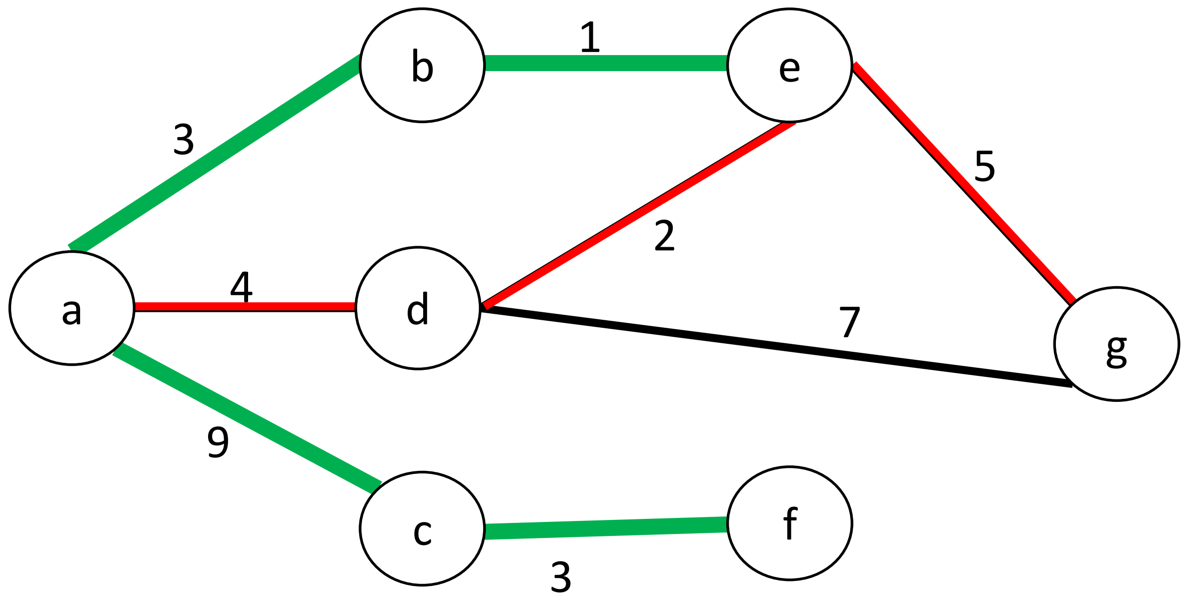
Example

S	T
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1



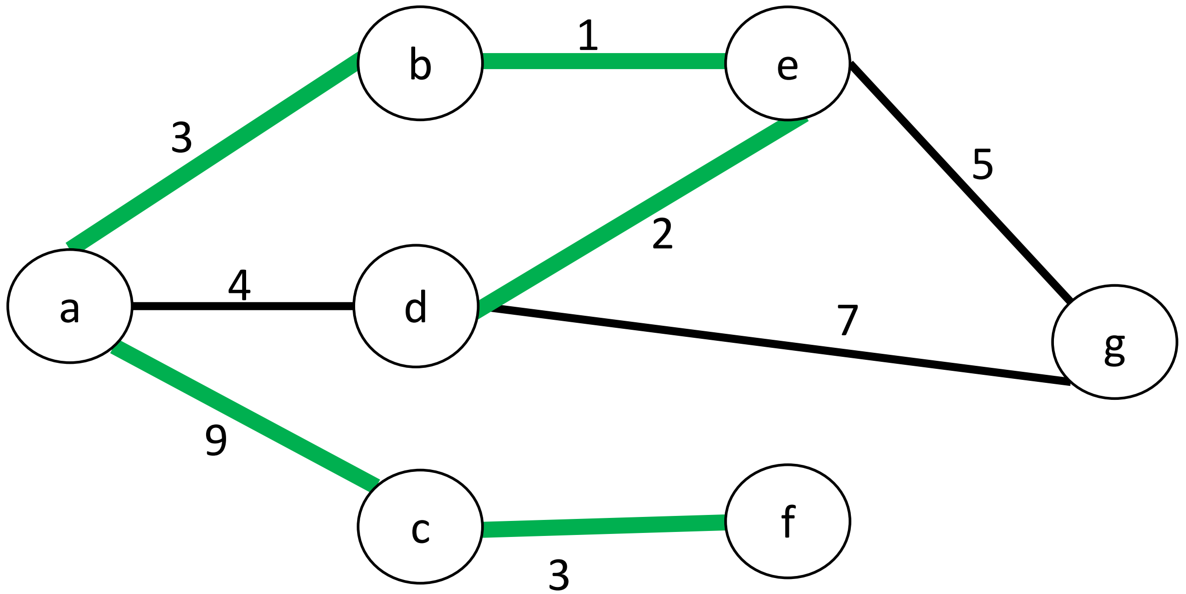
Example

S	**T**
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1



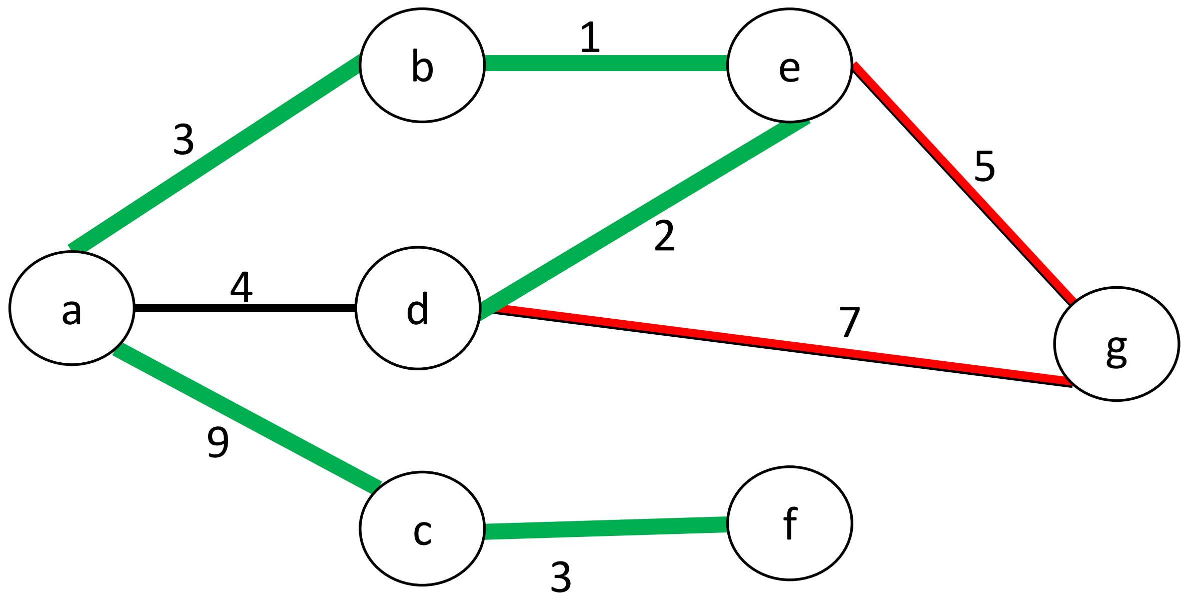
Example

S	**T**
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1
d	(d, e) 2



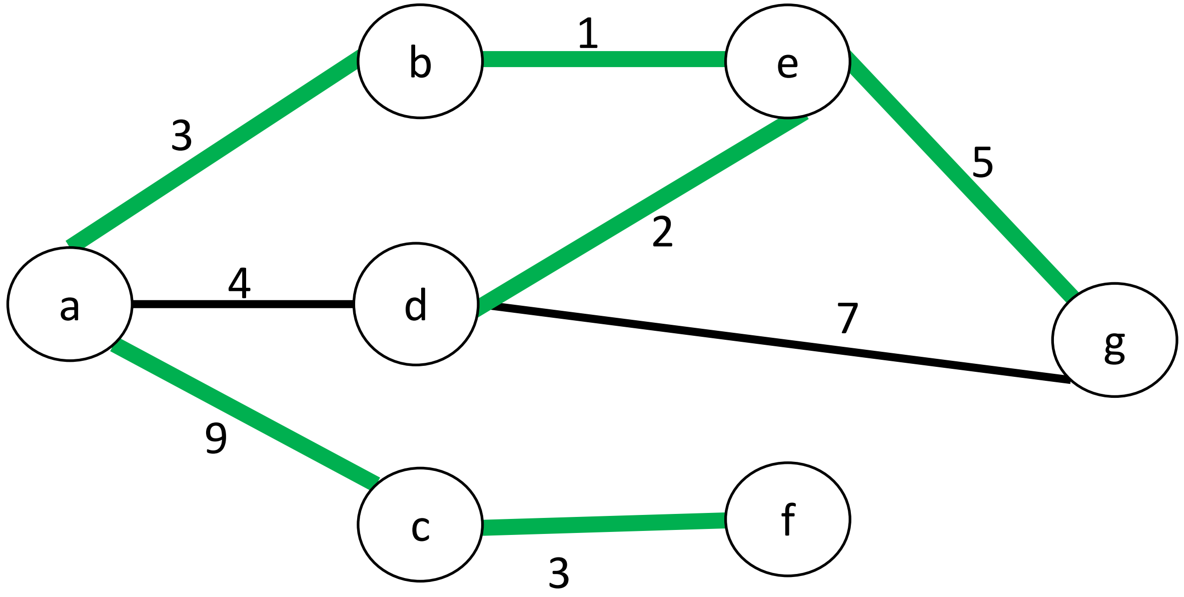
Example

S	**T**
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1
d	(d, e) 2



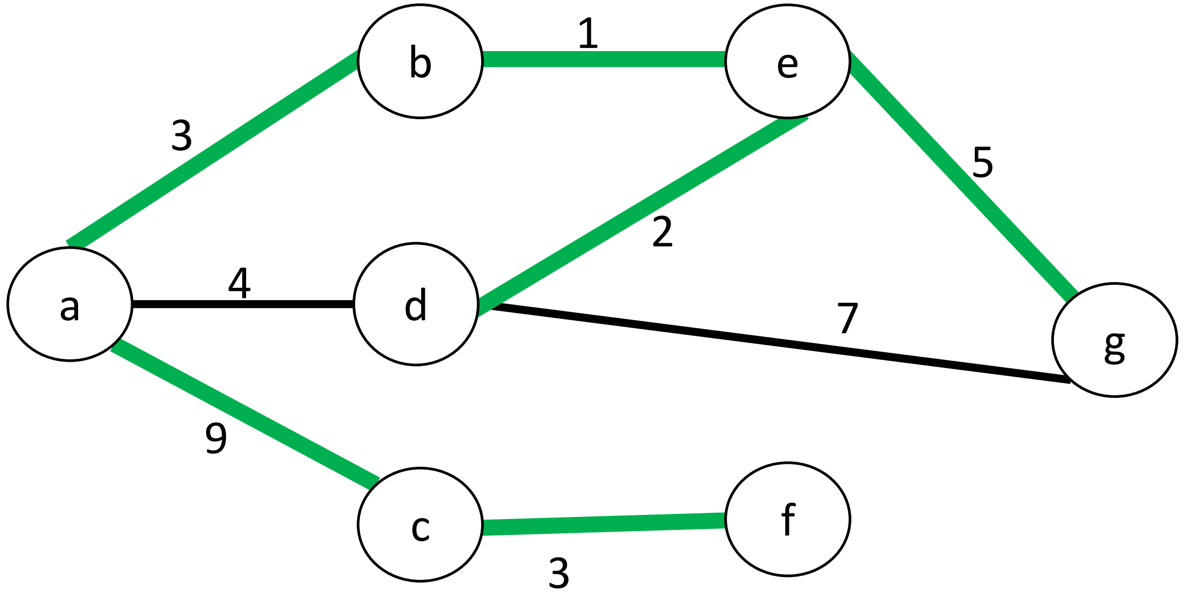
Example

S	T
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1
d	(d, e) 2
g	(e, g) 5

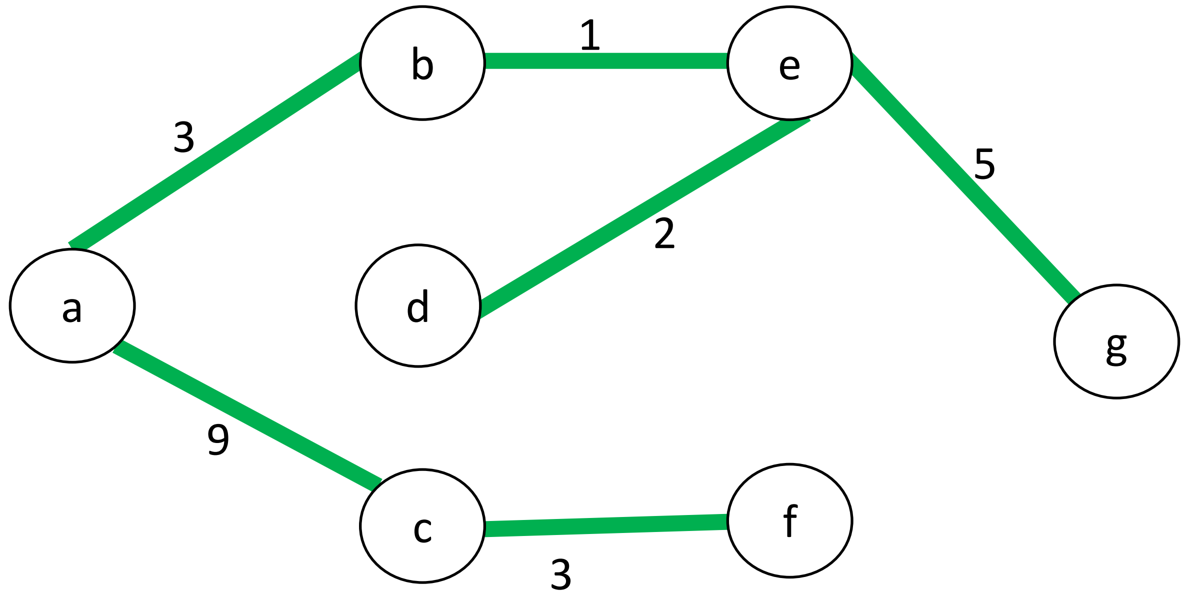


Example

S	T
c	
f	(c, f) 3
a	(a, c) 9
b	(a, b) 3
e	(b, e) 1
d	(d, e) 2
g	(e, g) 5



Weight = 3+9+3+1+2+5= 23



Weight = 3+9+3+1+2+5= 23