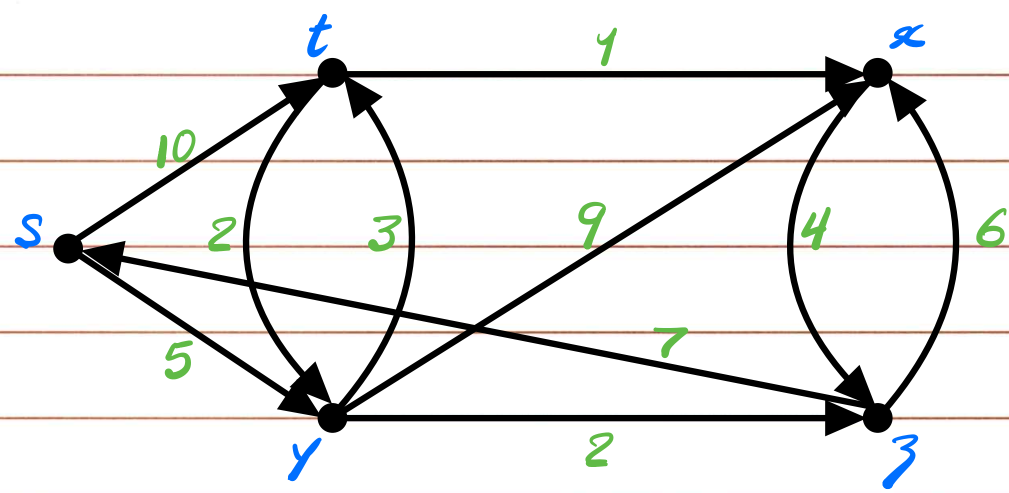
Shortest Path Problem

Problem statement:

Given $G=(V,E)$ with $\omega(u,v)\geq 0$ for each edge $(u,v)\in E$, find the shortest path from $s\in V$ to $V-s$

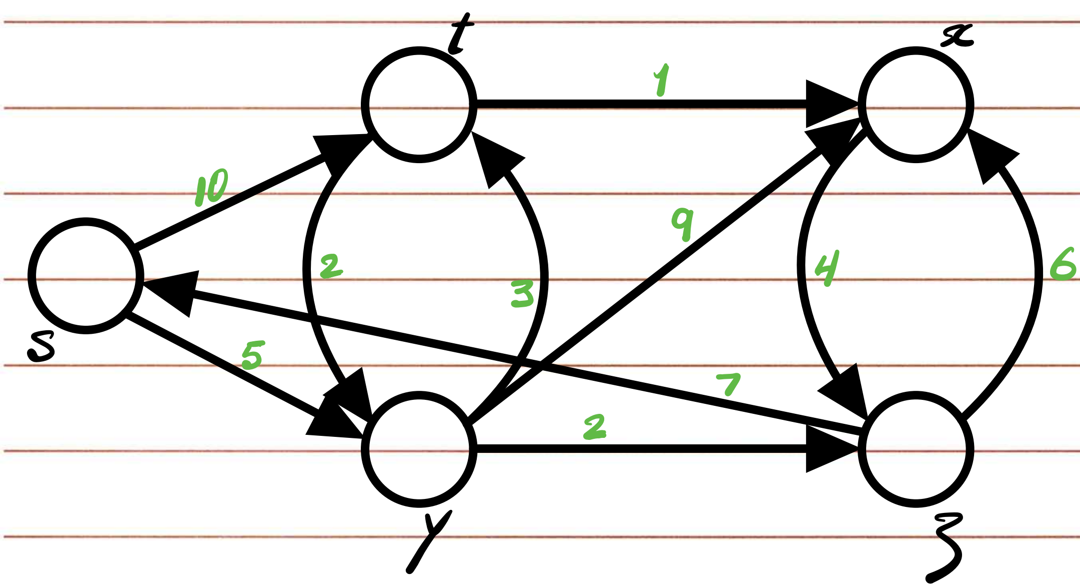


High Level Solution

1_ Start with a set S of vertices whose final shortest path we already know.

2_ At each step, find a vertex $v \in V - S$

3_ Add v to S, and repeat.



Proof of Correctness

We will prove that at each step, Dijkstra's algorithm finds the shortest path to a new node in the graph.

Implementation of Dijkstra's

Initially $S = \text{Null}$, $d(s) = 0$, and
for all other nodes $d(u) = \infty$

While $S \neq V$

Select a node $v \notin S$ with at least one edge from S for which

$$d(v) = \min_{e(u,v):\, u \in S} (d(u) + l_e)$$

Add v to S

endwhile

What is the ideal data structure to store the set V?

More Detailed Implementation of Dijkstra's Alg.

S = Null
Initialize priority queue Q with all nodes V where d(v) is the key value. (All d(v)'s are set to ∞, except for s where d(s) = 0)

While $S \neq V$
    v = Extract_Min(Q)
    $S = S \cup \{v\}$
    for each vertex $u \in Adj(v)$
        if $d(u) > d(v) + l_e$
            Decrease-key(Q, u, $d(v) + l_e$)
    endfor
endwhile

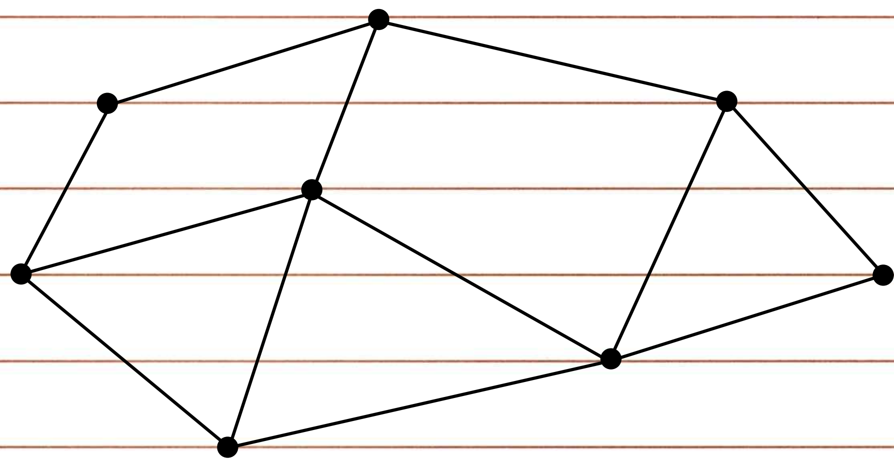
Complexity Analysis

- Initialize / construct priority queue
- Max. no. of Extract-Min operations
- Max. no. of Decrease-key operations

	Binary Heap	Binomial Heap	Fibonacci Heap
n Extract_Min's m Decrease-key's			
Total			
Sparse graphs			
Dense graphs			

Problem Statment

Find a minimum cost network that connects all nodes in the weighted undirected gragh G.

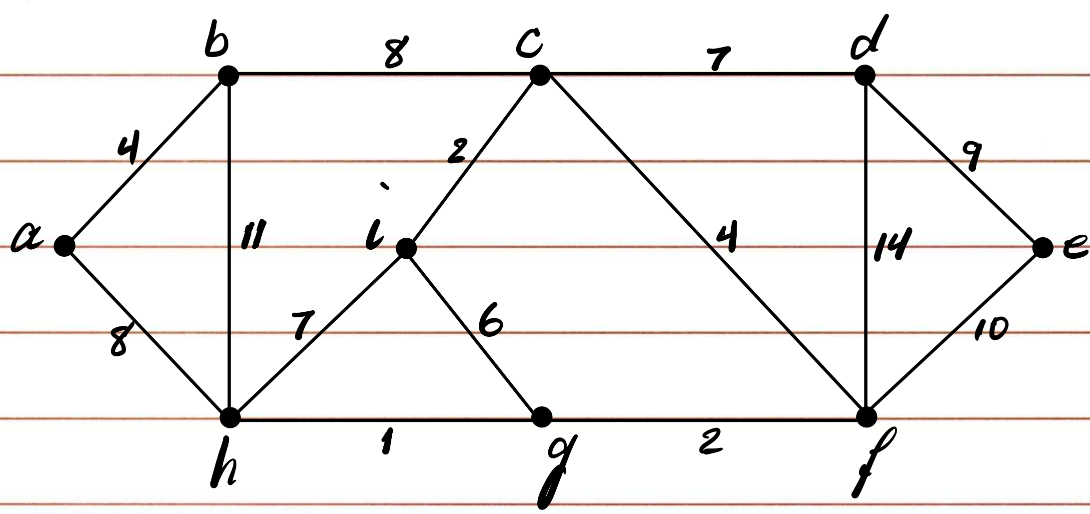


Def. Any tree that covers all nodes of a gragh is called a spanning tree.

Def. A spanning tree with minimum total edge cost is a minimum spanning tree. (MST)

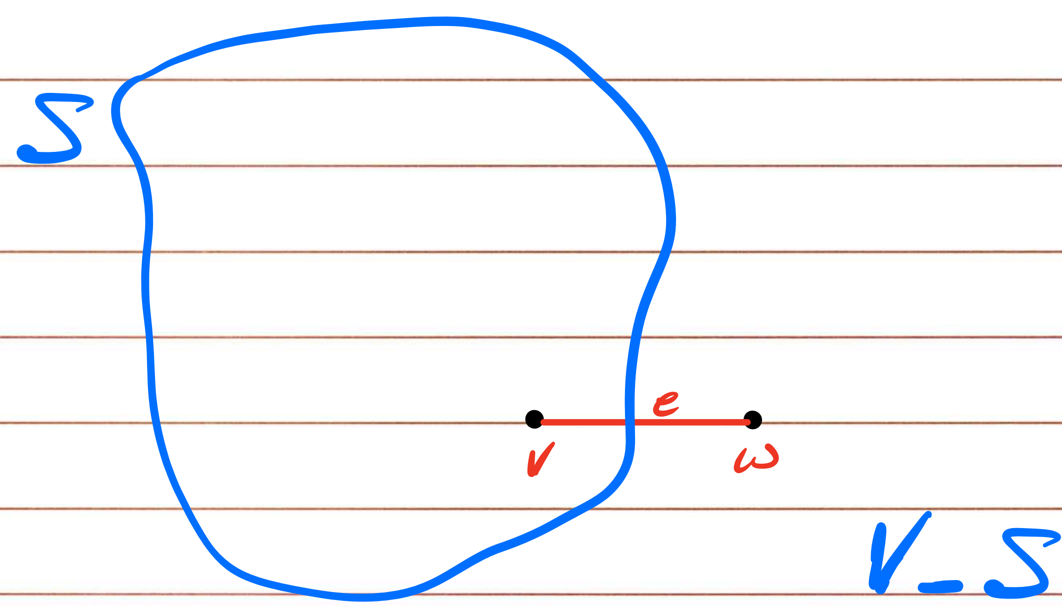
Problem Statement

Find a MST in an undirected graph.



b
8
c
7
d
4
2
9
a
11
i
4
14
e
8
7
6
10
h
1
g
2
f

Fact: Let S be any subset of nodes that is neither empty nor equal to all of V, and let edge $e = (v, w)$ be the minimum cost edge with one end in S and the other end in $V - S$. Then every MST contains the edge $\underline{e}$.



Proof of correctness for Kruskal's Alg.

Proof of correctness for Prim's Alg.

Proof of correctness for Reverse-Delete.

More Detailed Implementation of Dijkstra's Alg.

$S = \text{Null}$

Initialize priority queue Q with all nodes V where $d(v)$ is the key value.
(All $d(v)$'s are set to ∞, except for $\underline{s}$ where $d(s) = 0$)

While $S \neq V$
    $v = \text{Extract-Min}(Q)$
    $S = S \quad \{v\}$
    for each vertex $u \in Adj(v)$
        if $d(u) > d(v) + l_e$
            Decrease-key$(Q, u, d(v) + l_e)$
    endfor
endwhile

Kruskal's Alg. – High level Implementation

Create an independent set for each node

A = Null

Sort edges in non-decreasing order of weight

For each edge $(u,v) \in E$ taken in this order

    if u and v are NOT in the same set, then

        $A = A \cup \{(u,v)\}$

        Merge the two sets

    endif

Endfor

More Detailed Implementation of Kruskal's Alg.

Prim's	Kruskal's
$O(m \lg n)$	$O(m \lg m)$

Reverse-Delete - High level Implementation:

Sort the edges of E into a non-increasing order of cost

For each edge $(u,v) \in E$ in this order
    if removing (u,v) does not disconnect the graph, then
        Remove (u,v)
    endif
Endfor