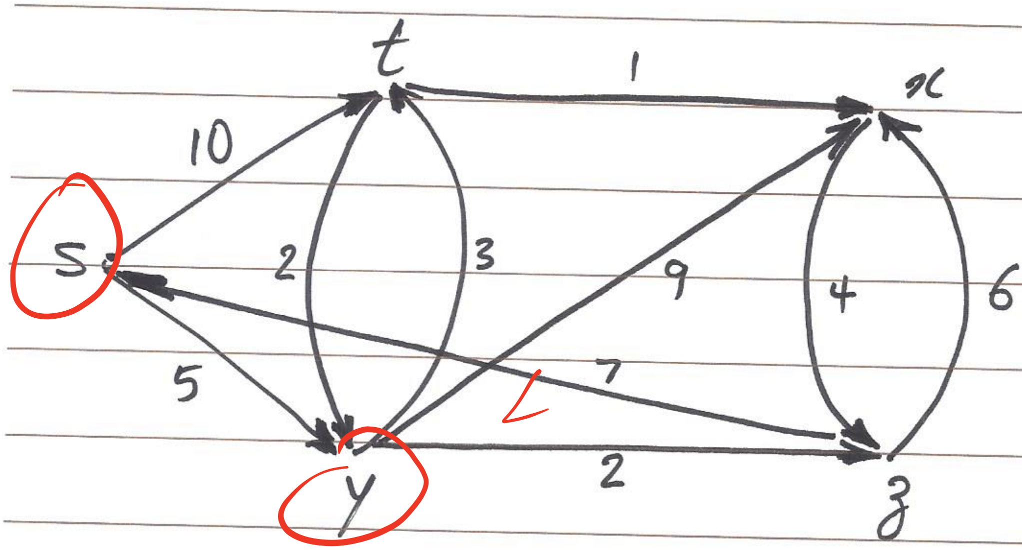
Shortest Path

Problem Statement:

Given $G = (V, E)$ with $\boxed{w(U,V) \geq 0}$ for each edge $(U,V) \in E$, find the shortest path from $s \in V$ to $V - \cancel{S}\, s$.



Solution

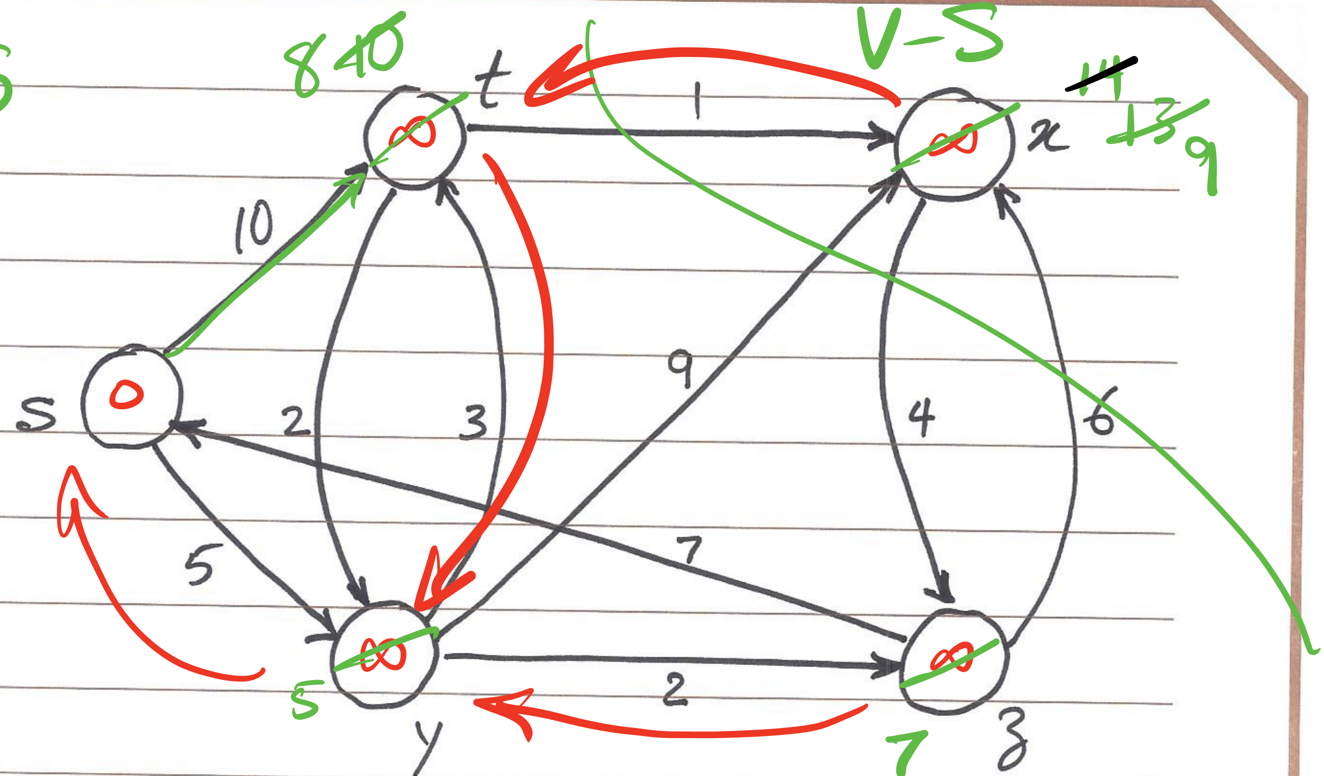
1. Start with a set S of vertices whose final shortest path we already know.

2. At each step, find a vertex $u \in V - S$ with shortest distance from s.

3. Add u to S, and repeat.

S

V-S



$S_1 = \{s\}$, $S_2 = \{s, y\}$, $S_3 = \{s, y, z\}$

$S_4 = \{s, y, z, t\}$, $S_5 = \{s, y, z, t, x\}$

Dijkstra's Alg.

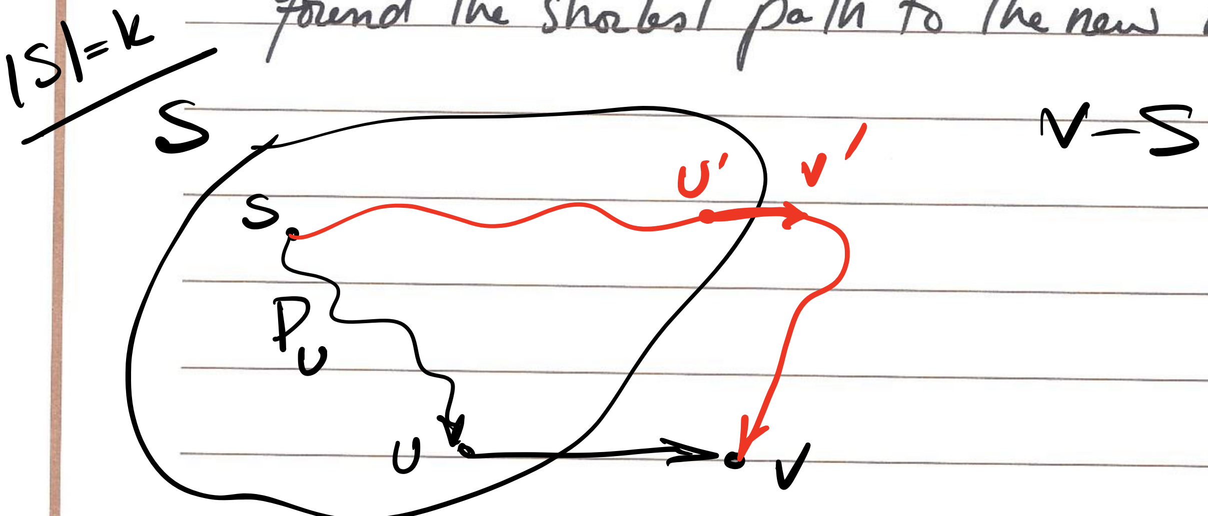
Proof of Correctness

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

Proof by mathematical induction:

Base Case: $|S| = 1$, $S = \{s\}$ and $d(s) = 0$

Inductive Step: Suppose the claim holds when $|S| = k$ for some $k \geq 1$. We now grow S to size $k+1$ and prove that we have found the shortest path to the new node.



Implementation of Dijkstra's

Initially $S = \{s\}$ and $d(s) = 0$
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

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

Add v to S

endwhile

More Detailed Implementation of Dijkstra's

$S = \text{Null}$

Initialize priority Queue Q with all nodes V where $d(v)$ is the key value.

(All $d(v)$'s are $= \infty$, except for s where $d(s) = 0$)

While $S \neq V$

$v = \text{Extract-Min}(Q)$

$S = S \cup \{v\}$

for each vertex $u \in Adj(v)$

if $d(u) > d(v) + l_e$;

cost of edge from v to u

Decrease-key$(Q, u, d(v) + l_e)$

end for

endwhile

$O(n)$

$O(n)$

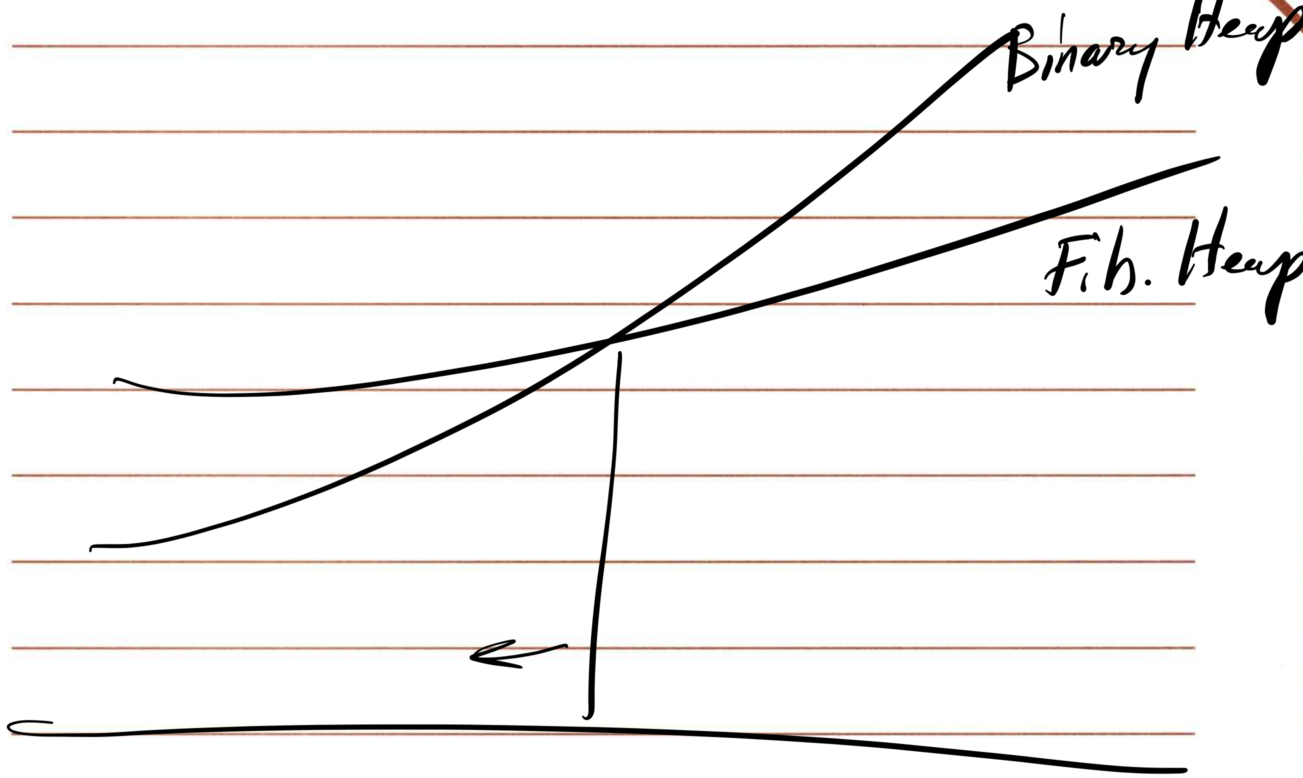
$O(n^2)$

$O(m)$

Complexity Analysis

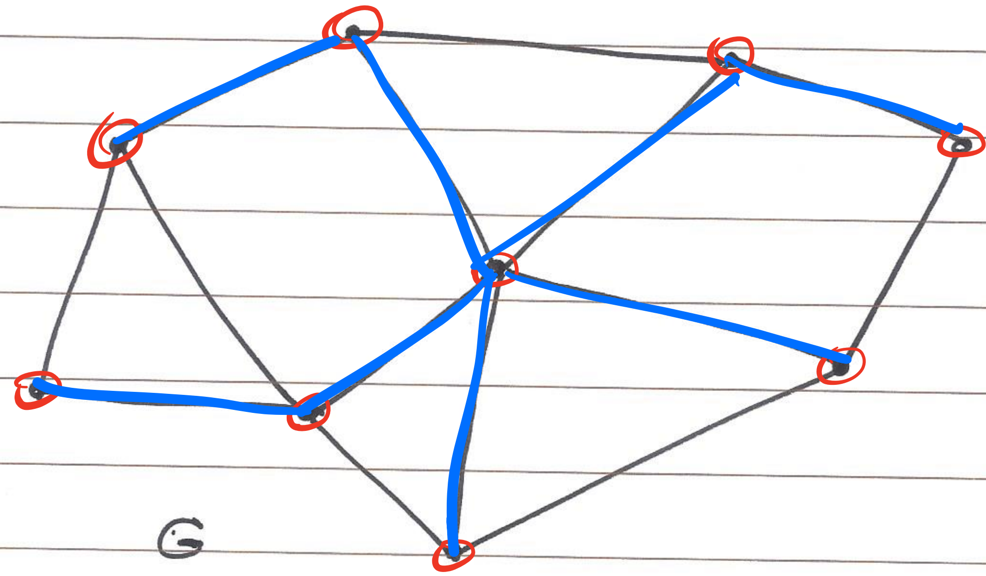
- Initialize Priority Queue $O(n)$
- Max. no. of Extract-Min op's : $O(n)$
- Max. no. of Decrease-key op's : $O(m)$

	Binary Heap	Binomial Heap	Fibonacci Heap
n*Extract-Min's	$O(n \lg n)$	$O(n \lg n)$	$O(n \lg n)$
m*Decrease_key's	$O(m \lg n)$	$O(m \lg n)$	$O(m)$
Total	$O(m \lg n)$	$O(m \lg n)$	$O(m + n \lg n)$
Sparse Graphs	$O(n \lg n)$	$O(n \lg n)$	$O(n \lg n)$
Dense Graphs	$O(n^2 \lg n)$	$O(n^2 \lg n)$	$O(n^2)$


Binary Heap
Fib. Heap

Problem Statement

Find minimum cost network that connects all nodes in G.



G

Def. Any tree that covers all nodes of a graph 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

Sol. 1: Sort all edges in increasing order of cost. Add edges to T in this order as long as it does not create a cycle. If it does, discard the edge.

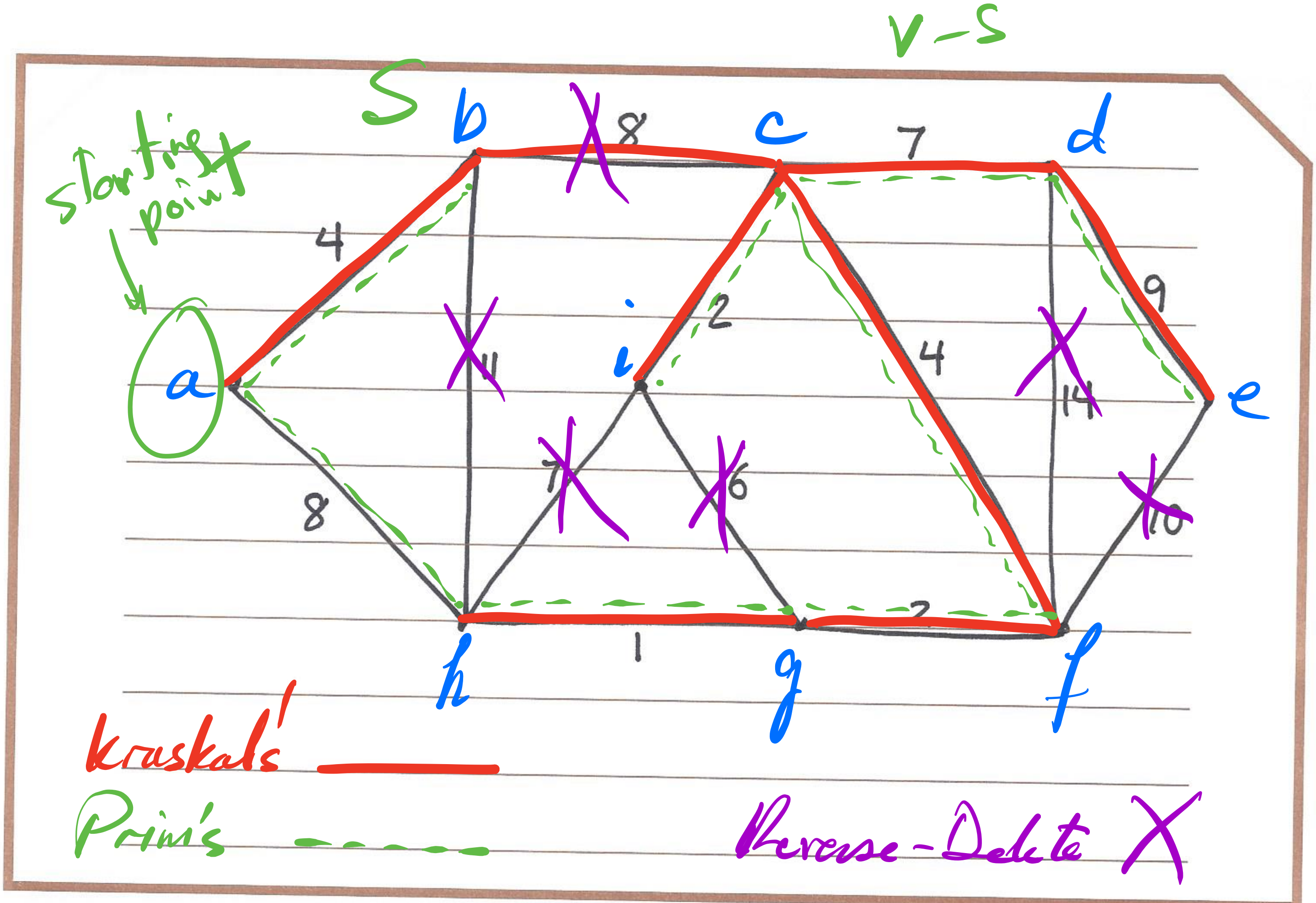
Kruskal's Alg.

Sol. 2: Similar to Dijkstra's algorithm, start with a node set S (initially the root node) on which a minimum spanning tree has been constructed so far. At each step, grow S by one node, adding the node v that minimizes the attachment cost.

Prim's Alg.

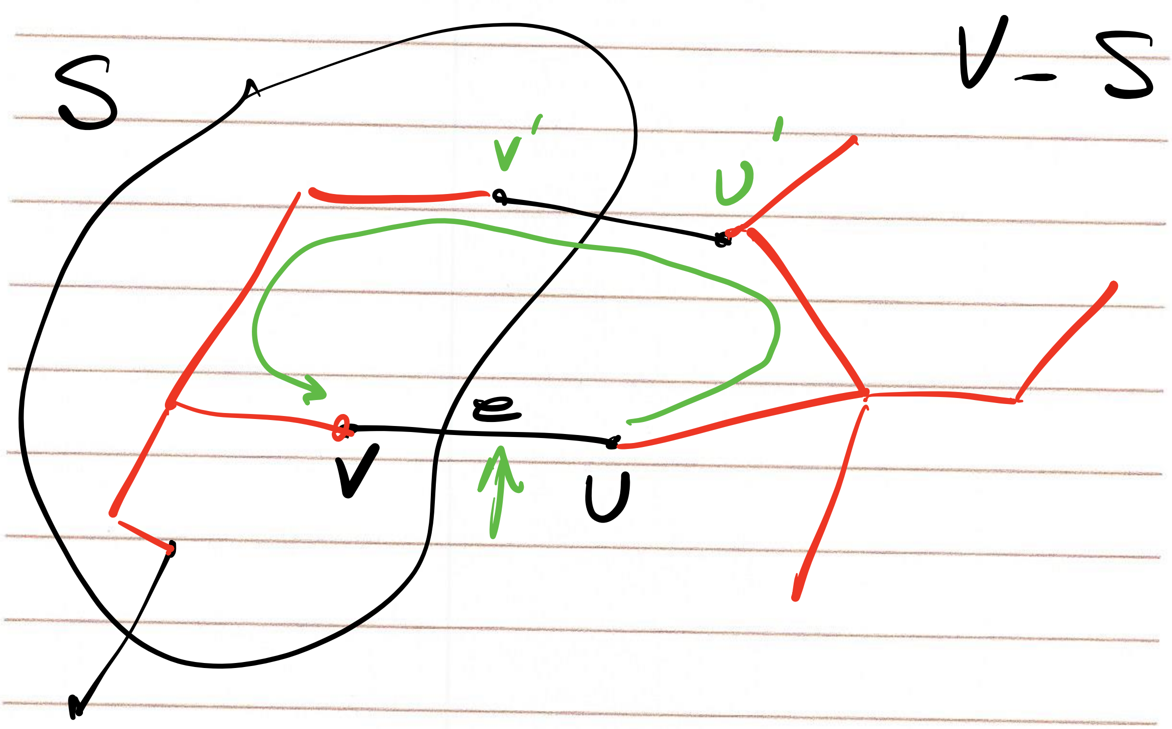
Sol. 3: Backward version of Kruskal's.
Start with a full graph (V, E).
Begin deleting edges in order
of decreasing cost as long as
it does not disconnect the graph

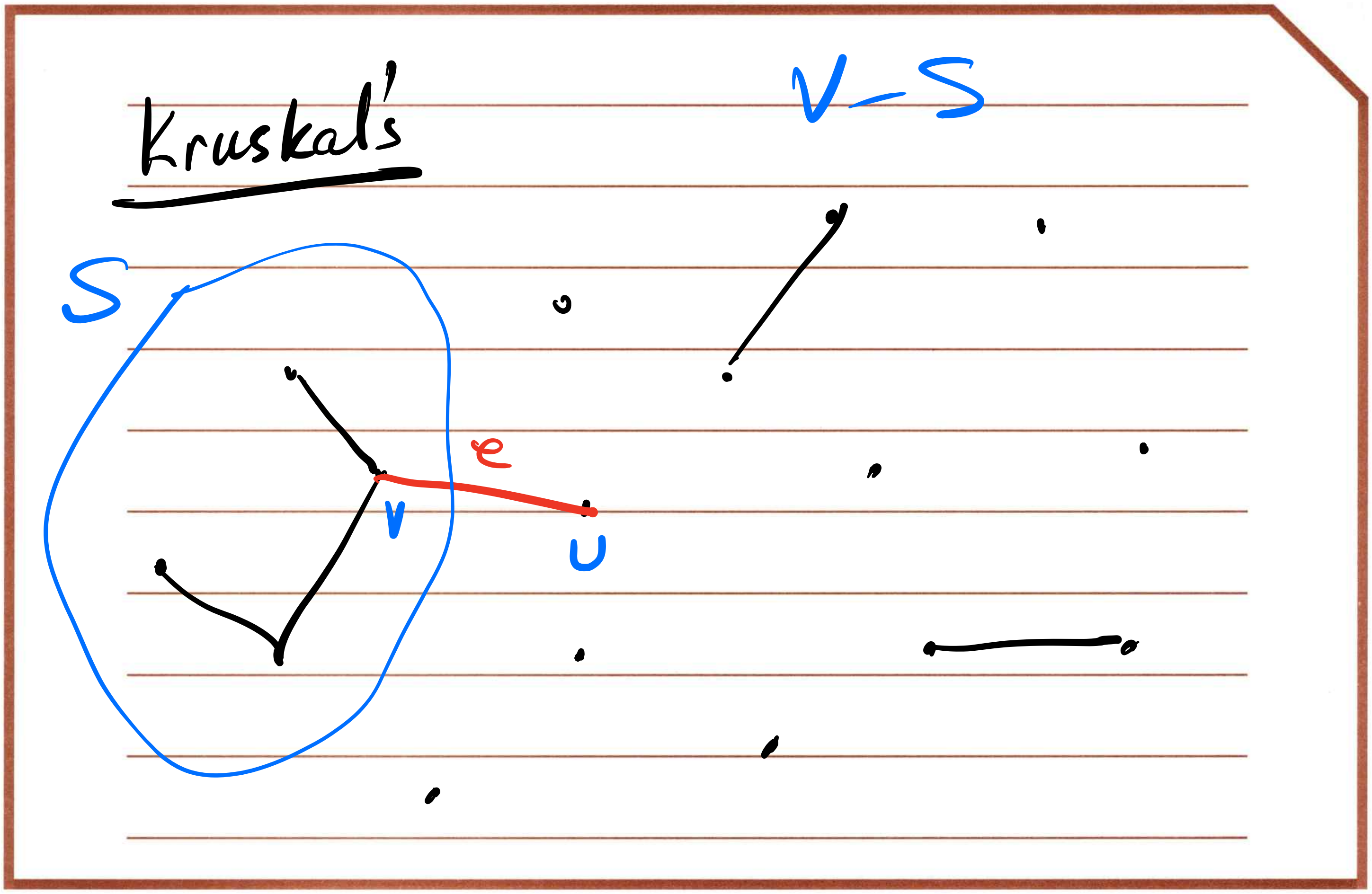
Reverse_Delete

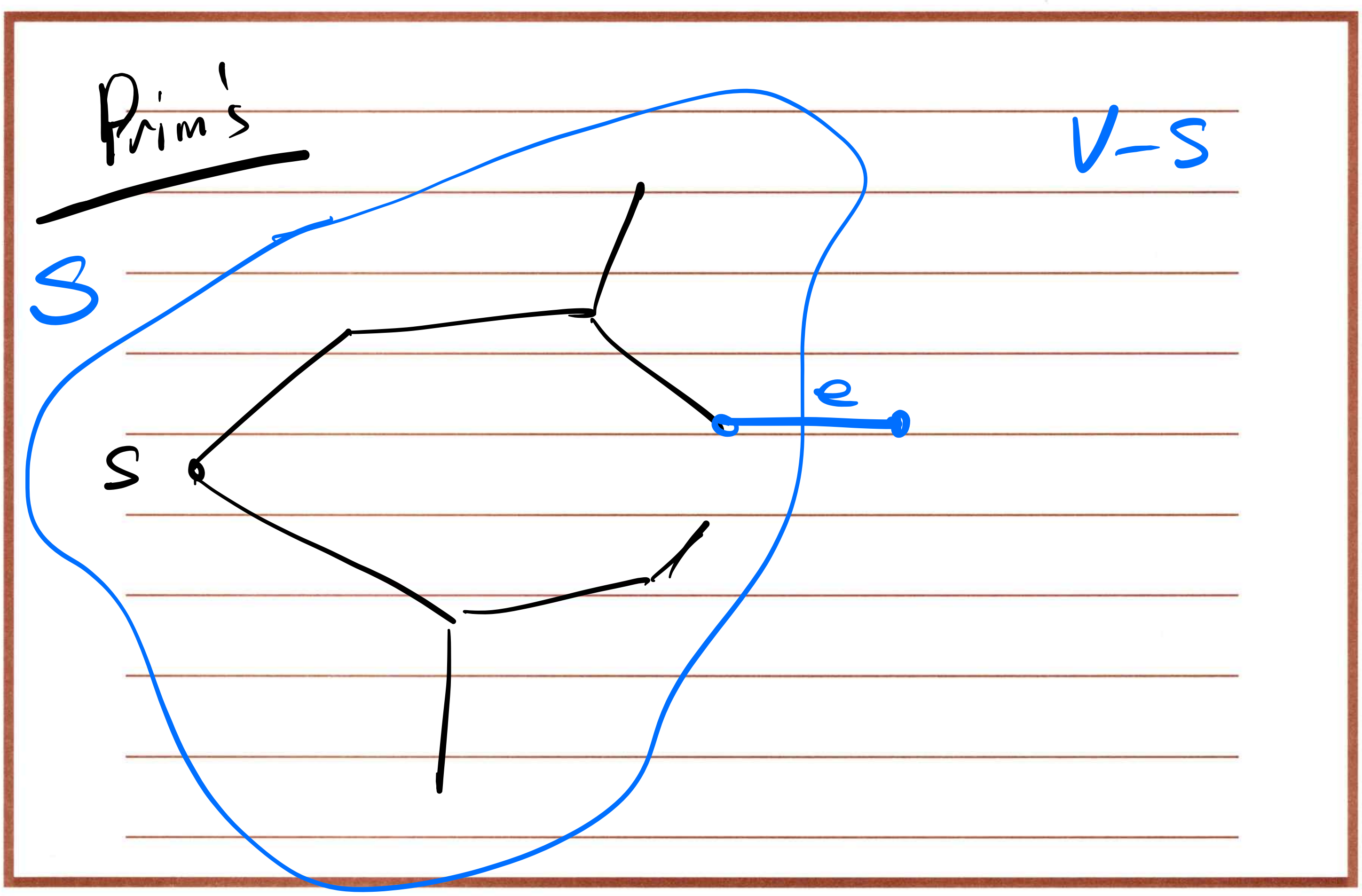


V-S
S
starting point
a
b
c
d
e
f
g
h
i
4
8
11
8
7
2
6
1
2
4
14
7
9
10
Kruskal's
Prim's
Reverse-Delete X

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 min cost edge with one end in S and the other end in $V - S$.
Then every MST contains the edge $\underline{e}$.




Kruskal's
V-S
S
e
v
u


Prim's
V-S
S
s
e

More Detailed Implementation of ~~Dijkstra's~~ Prim's

$S = \text{Null}$

Initialize priority Queue Q with
all nodes V where $d(v)$ is the key value.
(All $d(v)$'s are $= \infty$, except for s where $d(s) = 0$)

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

Kruskal's

Create an empty 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 & V are NOT in the same set then
$A = A \cup \{(U,V)\}$
merge the two sets
endif
endfor

Union-Find data structure

- Make set — $O(1)$ for set size = 1
- Find set — $O(1)$ or $O(\lg n)$
- Union — $O(\lg n)$ or $O(1)$

array impl. — ptr impl.

Implementation of Kruskal's

A = Null

$O(n)$ for each vertex $v \in V$
    Make-set (v)
endfor

$O(m \lg m)$ Sort the edges of E into non-decreasing order of cost

$O(m)$ for each edge $(u,v) \in E$ in this order.
    if Find-set (u) $\neq$ Find-set(v) then
        $A = A \cup \{(u,v)\}$
        Union (u,v)
    endif
endfor

overall complexity $= O(n) + O(m \lg m) + O(m \lg n)$

$= O(m \lg m)$

Prim's

$O(m \lg n)$

Kruskal's

$O(m \lg m)$

$$O(m \lg m) = O(m \lg n^2)$$
$$= O(m \lg n)$$

Your name!

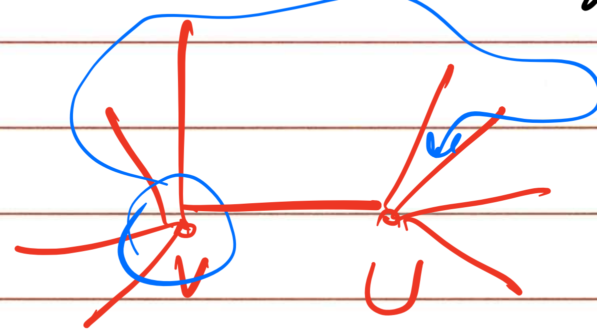
Reverse-Delete

$O(m \lg m)$ Sort edges ~ ~ ~

loop over edges

$O(m)$ $O(m+n)$ for edge e check to see if removing it will disconnect the graph

⋮



overall cost = $O(m^2)$

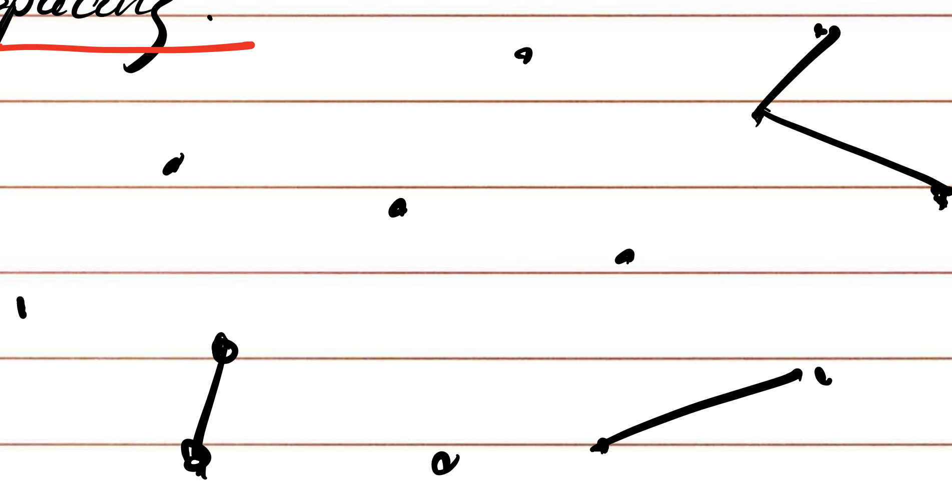
Clustering

Def. Given a set U of $\underline{n}$ objects $p_1 \ldots p_n$ where $d(p_i, p_i) = 0$, $d(p_i, p_j) > 0$ for $p_i \neq p_j$, and $d(p_i, p_j) = d(p_j, p_i)$, a <u>k-clustering</u> of U is a partition of U into $\underline{k}$ nonempty sets $C_1 \ldots C_k$

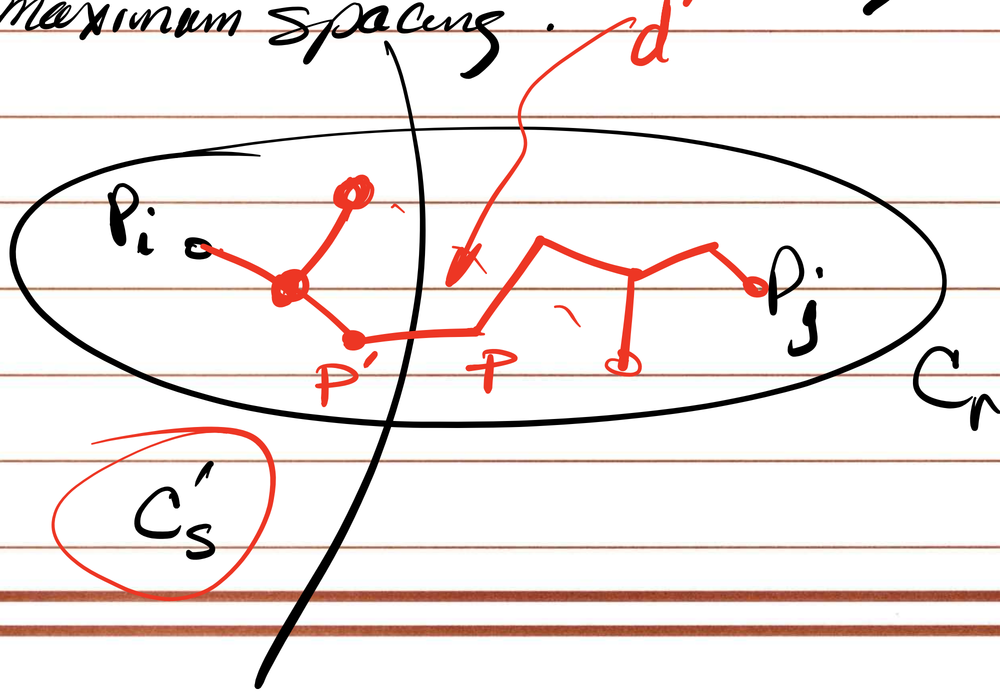
Def. The <u>spacing</u> of a k-clustering is the minimum distance between any pair of points lying in different clusters.

Problem Statement:

Given a set of $\underline{n}$ objects as described above, find a k-clustering with <u>maximum spacing</u>.



Claim: clusters $C_1 \ldots C_k$ created by deleting the $k-1$ most expensive edges of the MST T give us a k-clustering of maximum spacing.



Say our spacing is $\underline{d}$

Cost of $PP' \leq \underline{d}$

d'

Discussion 4

1. Hardy decides to start running to work in San Francisco city to get in shape. He prefers a route that goes entirely uphill and then entirely downhill so that he could work up a sweat uphill and get a nice, cool breeze at the end of his run as he runs faster downhill. He starts running from his home and ends at his workplace. To guide his run, he prints out a detailed map of the roads between home and work with k intersections and m road segments (any existing road between two intersections). The length of each road segment and the elevations at every intersection are given. Assuming that every road segment is either fully uphill or fully downhill, give an efficient algorithm to find the shortest path (route) that meets Hardy's specifications. If no such path meets Hardy's specifications, your algorithm should determine this. Justify your answer.

2. You are given a graph representing the several career paths available in industry. Each node represents a position and there is an edge from node *v* to node *u* if and only if *v* is a prerequisite for *u*. Top positions are the ones which are not prerequisites for any positions. The cost of an edge *(v, u)* is the effort required to go from one position *v* to position *u*. Ivan wants to start a career and achieve a top position with minimum effort. Using the given graph can you provide an algorithm with the same run time complexity as Dijkstra's? You may assume the graph is a DAG.

3. You have a stack data type, and you need to implement a FIFO queue. The stack has the usual POP and PUSH operations, and the cost of each operation is 1. The FIFO has two operations: ENQUEUE and DEQUEUE. We can implement a FIFO queue using two stacks. What is the amortized cost of ENQUEUE and DEQUEUE operations.

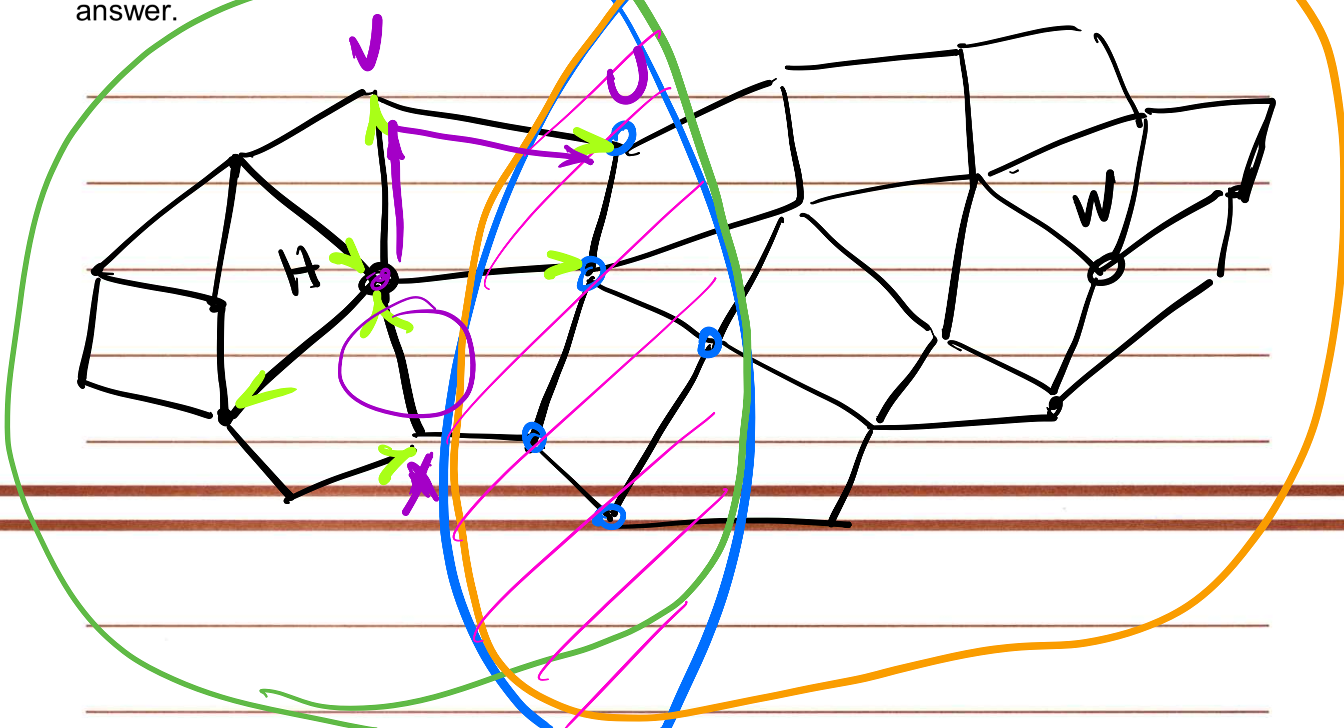
4. Given a sequence of numbers: 3, 5, 2, 8, 1, 5, 2, 7

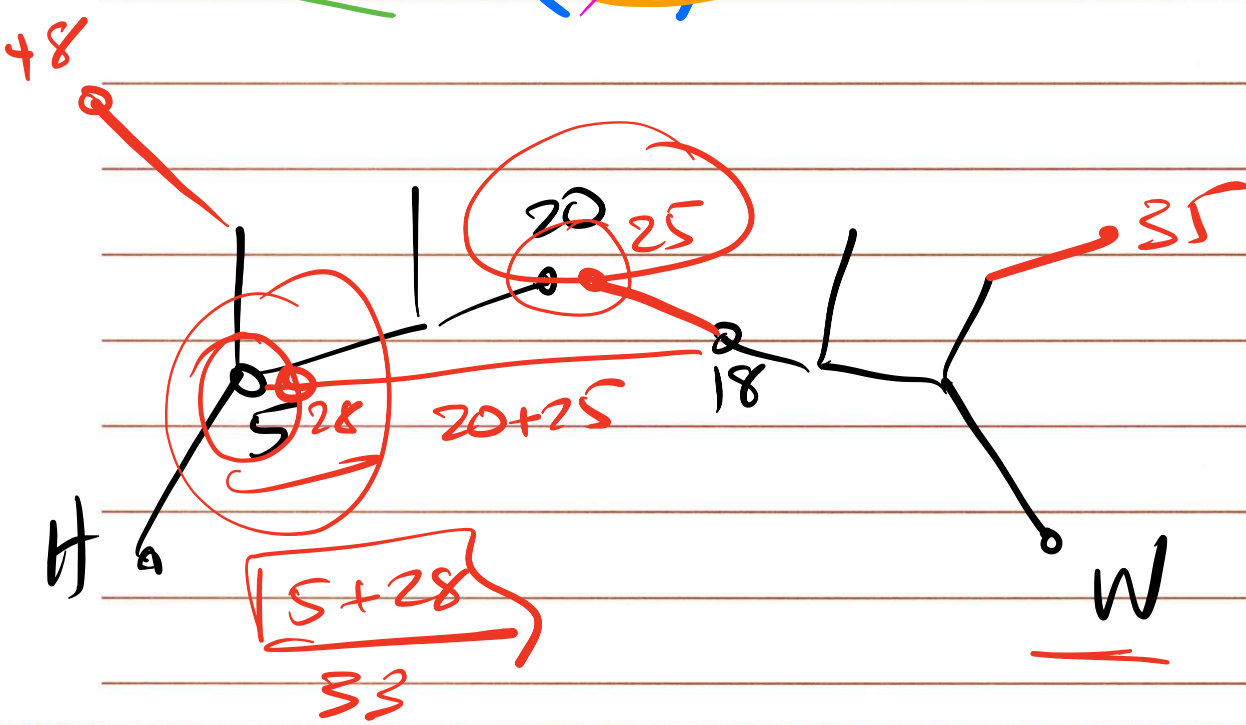
a. Draw a binomial heap by inserting the above numbers reading them from left to right
b. Show a heap that would be the result after the call to deleteMin() on this heap

5. (a): Suppose we are given an instance of the Minimum Spanning Tree problem on a graph G. Assume that all edges costs are distinct. Let T be a minimum spanning tree for this instance. Now suppose that we replace each edge cost c_e by its square, c_e^2 thereby creating a new instance of the problem with the same graph but different costs. Prove or disprove: T is still a MST for this new instance.

(b): Consider an undirected graph G = (V, E) with distinct nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G. Now suppose each edge weight is increased by 1: the new weights are $c'_e = c_e + 1$. Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

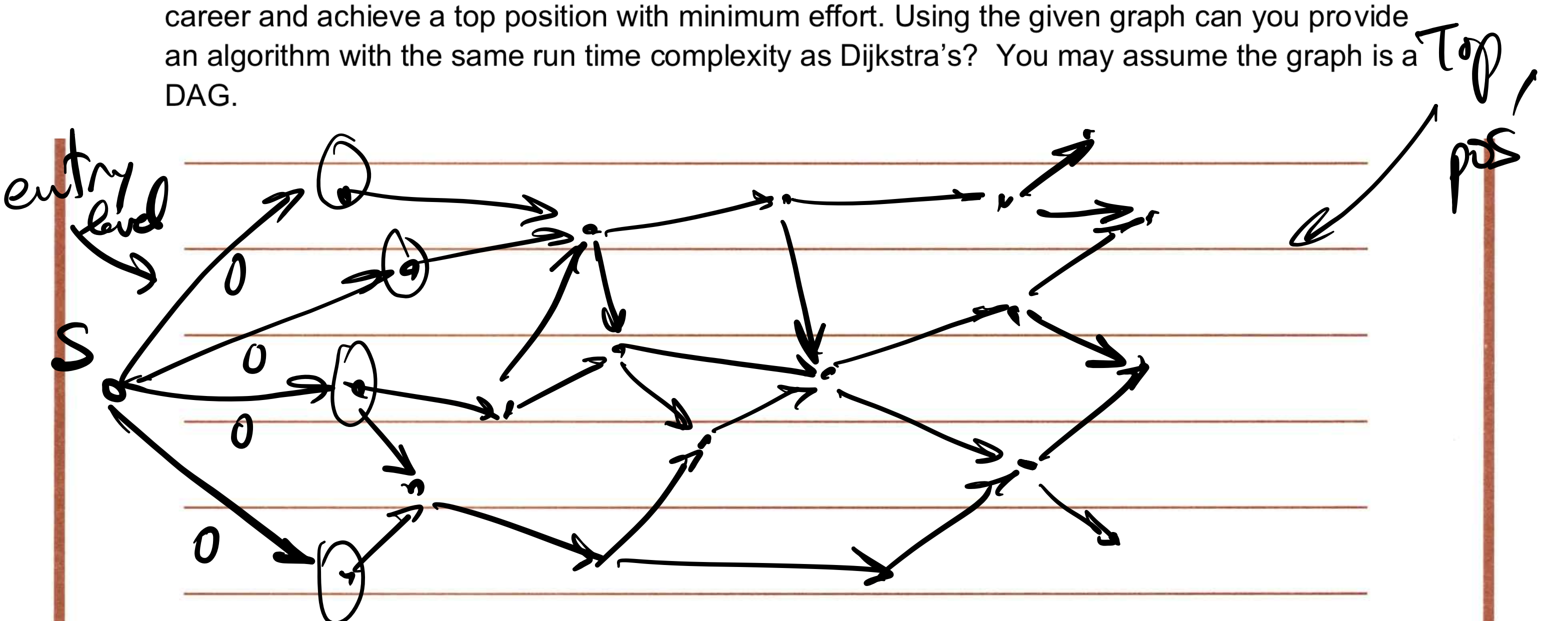
1. Hardy decides to start running to work in San Francisco city to get in shape. He prefers a route that goes entirely uphill and then entirely downhill so that he could work up a sweat uphill and get a nice, cool breeze at the end of his run as he runs faster downhill. He starts running from his home and ends at his workplace. To guide his run, he prints out a detailed map of the roads between home and work with k intersections and m road segments (any existing road between two intersections). The length of each road segment and the elevations at every intersection are given. Assuming that every road segment is either fully uphill or fully downhill, give an efficient algorithm to find the shortest path (route) that meets Hardy's specifications. If no such path meets Hardy's specifications, your algorithm should determine this. Justify your answer.





answer.

2. You are given a graph representing the several career paths available in industry. Each node represents a position and there is an edge from node *v* to node *u* if and only if *v* is a prerequisite for *u*. Top positions are the ones which are not prerequisites for any positions. The cost of an edge *(v, u)* is the effort required to go from one position *v* to position *u*. Ivan wants to start a career and achieve a top position with minimum effort. Using the given graph can you provide an algorithm with the same run time complexity as Dijkstra's? You may assume the graph is a DAG.



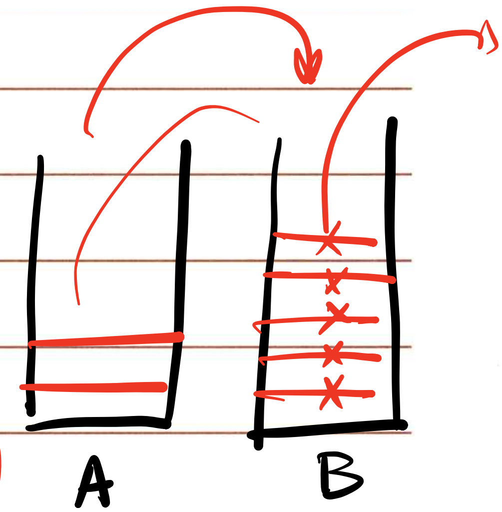
3. You have a stack data type, and you need to implement a FIFO queue. The stack has the usual POP and PUSH operations, and the cost of each operation is 1. The FIFO has two operations: ENQUEUE and DEQUEUE. We can implement a FIFO queue using two stacks. What is the amortized cost of ENQUEUE and DEQUEUE operations.

n enqueues $\rightarrow$ n pushes $= n$

1 dequeue $\rightarrow$ $n+n+1$

total Cost $= 3n+1$

Amortized Cost $= \frac{3n+1}{n} = O(1)$

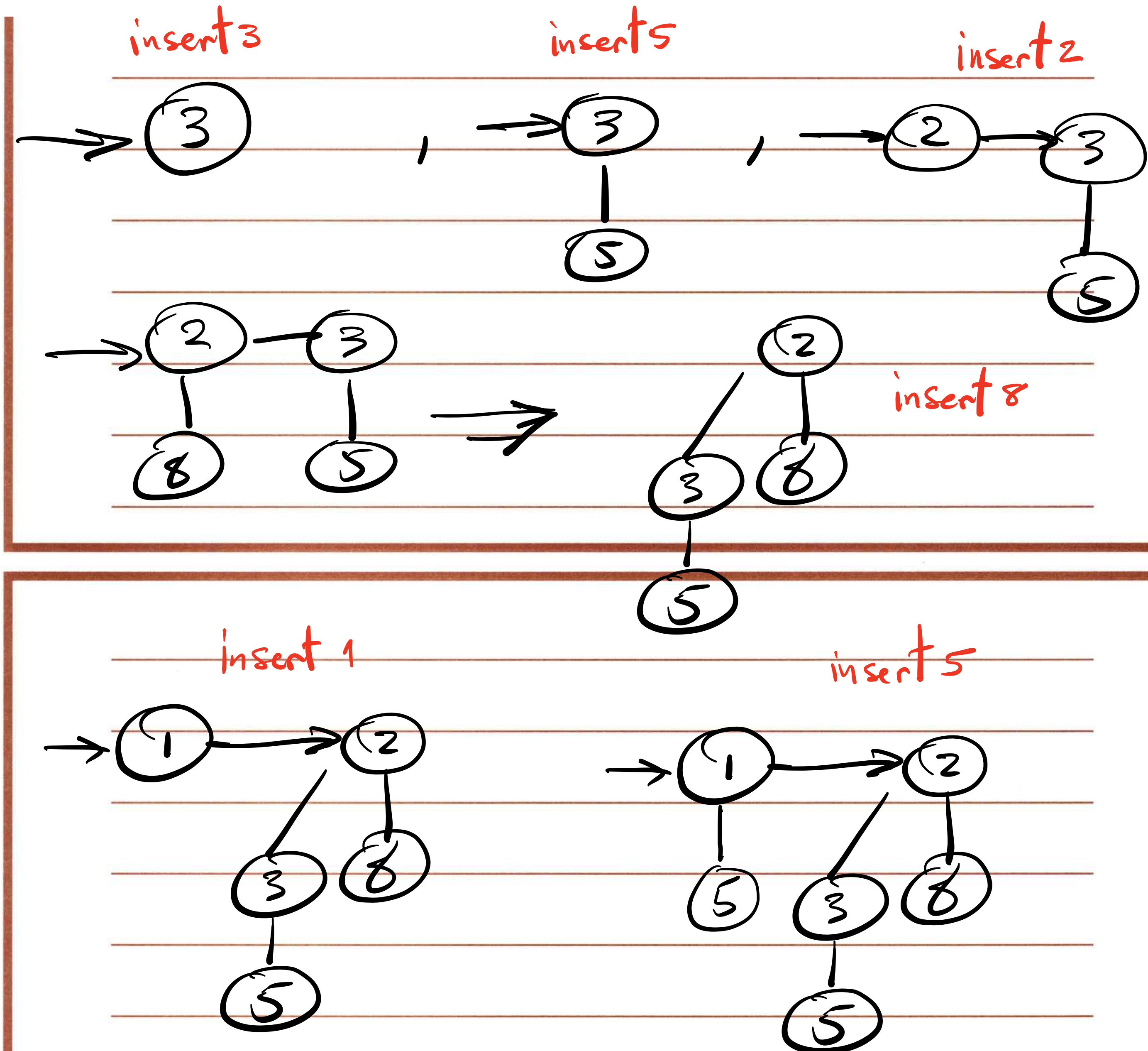


5. (a): Suppose we are given an instance of the Minimum Spanning Tree problem on a graph G. Assume that all edges costs are distinct. Let T be a minimum spanning tree for this instance. Now suppose that we replace each edge cost c_e by its square, c_e^2 thereby creating a new instance of the problem with the same graph but different costs. Prove or disprove: T is still a MST for this new instance.

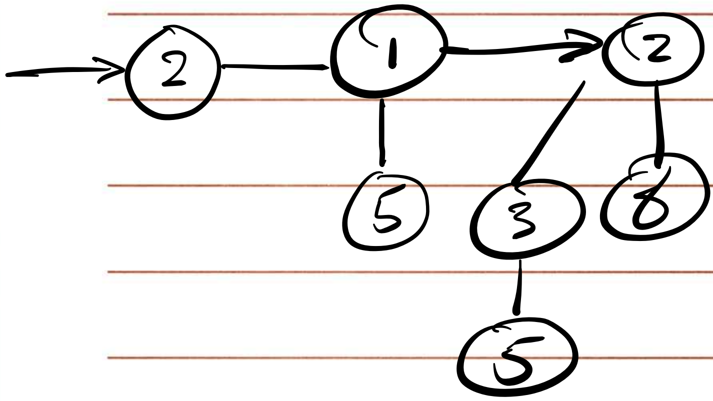
(b): Consider an undirected graph G = (V, E) with distinct nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G. Now suppose each edge weight is increased by 1: the new weights are $c'_e = c_e + 1$. Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

4. Given a sequence of numbers: 3, 5, 2, 8, 1, 5, 2, 7

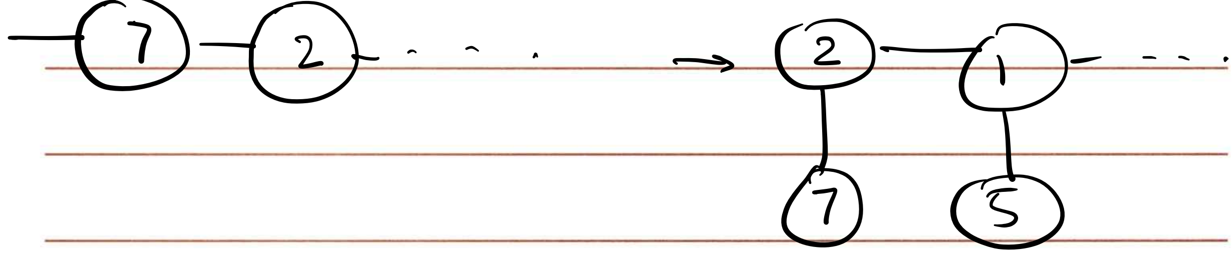
a. Draw a binomial heap by inserting the above numbers reading them from left to right
b. Show a heap that would be the result after the call to deleteMin() on this heap

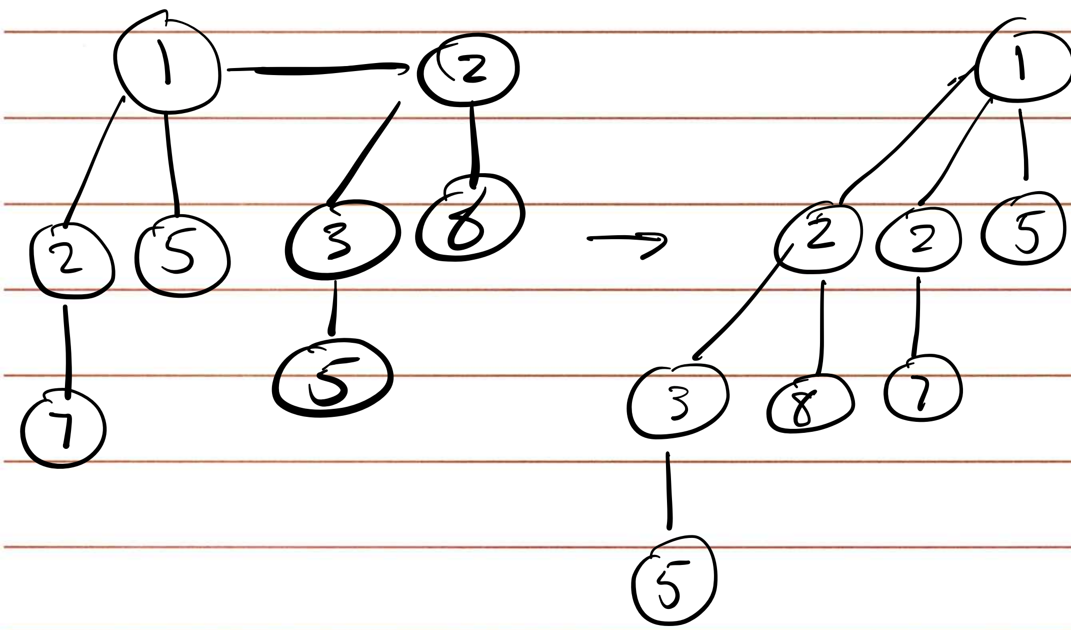


insert 2



insert 7





Delete.Min