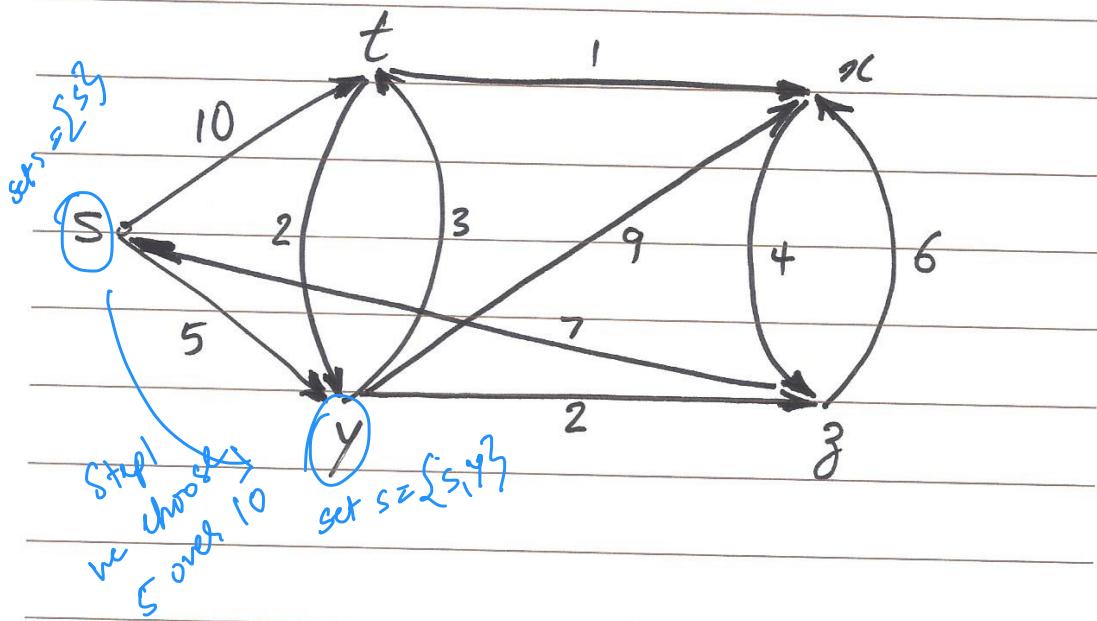


Shortest Path

Problem Statement:

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



Solution

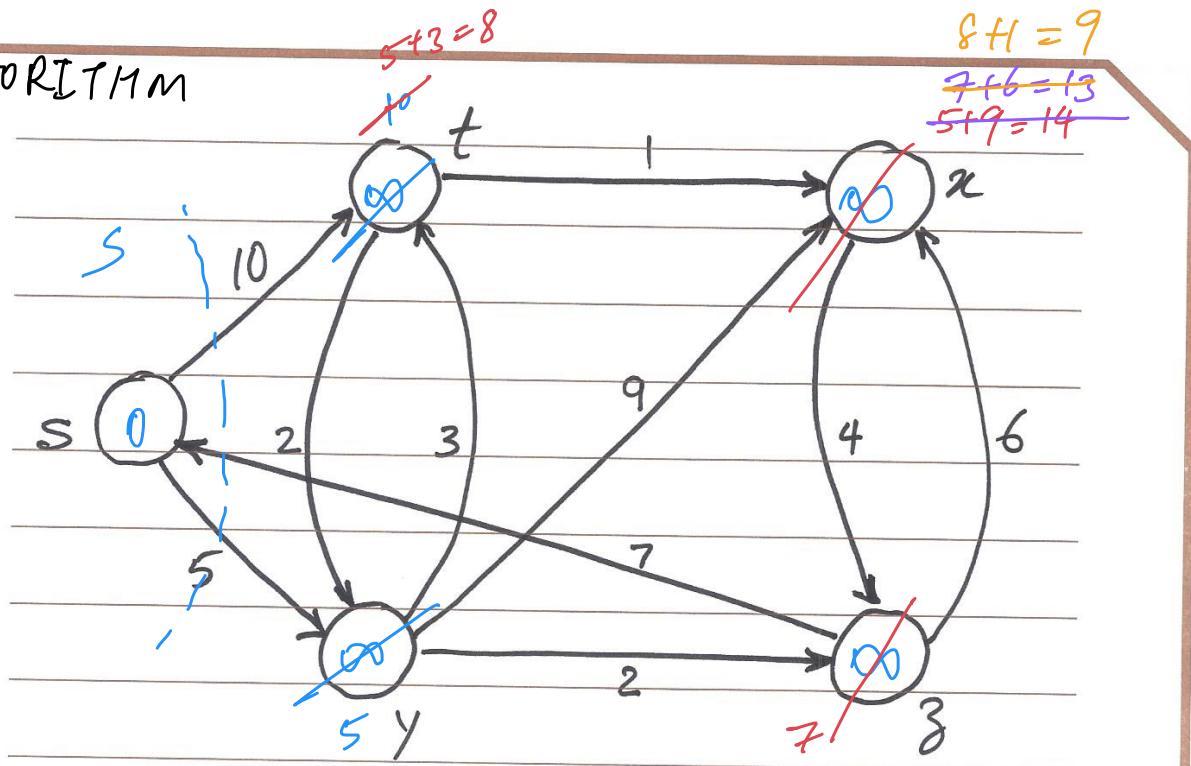
1 Start with a set S of vertices whose final shortest path we already know.

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

3. Add v to S , and repeat.

DJIKSTRA'S

ALGORITHM



Initially all nodes are at ∞ distance from start

$S_1 = \{S\}$ look at neighbours of S S-Y current os
but we can get to Y from S in cost 5

$$S_2 = \{S, Y\}$$

$$S_3 = \{S, Y, Z\}$$

$$S_4 = \{S, Y, Z, T\}$$

$$S_5 = \{S, Y, Z, T, X\}$$

By doing this we get shortest distance from S to all other nodes

To find shortest path we can keep track of parent

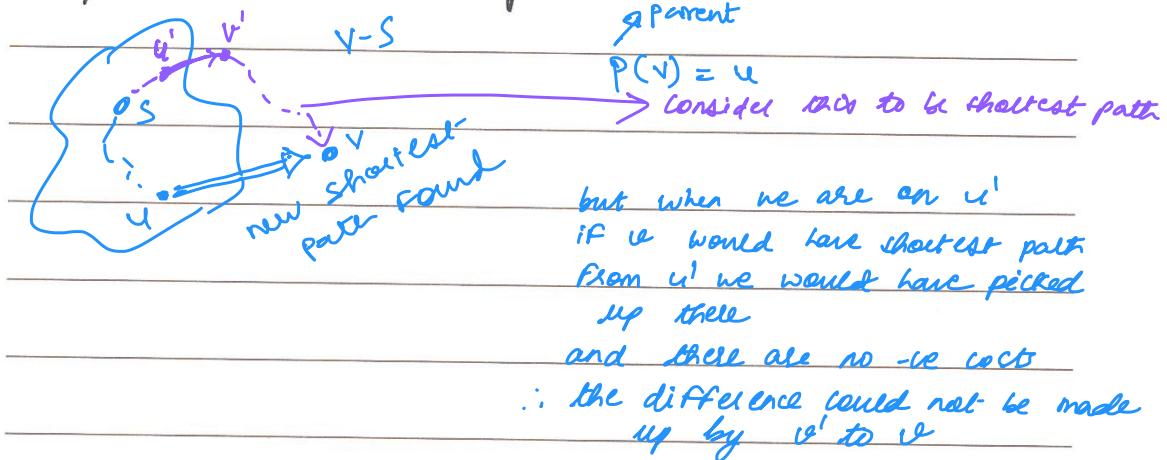
Proof of Correctness

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 $d(v) = \infty$

while $S \neq V$

Select a node $v \notin S$ with at least
one edge from S for which
 $d(v) = \min_{e(u,v): u \in S} (d(u) + le)$

for d
we will use priority queue

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 $S \neq V$

\nwarrow n operations

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

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

\nwarrow $O(n)$ for each vertex $u \in \text{Adj}(v)$

\nwarrow if $d(u) > d(v) + \text{le};$

Decrease-Key ($Q, u, d(v) + \text{le}$)

\nwarrow end for

\nwarrow end while

\nwarrow $O(n^2)$

\searrow But we can only
call every edge once

hence $O(n^2)$ can

be dropped to $O(n^3)$

Complexity Analysis

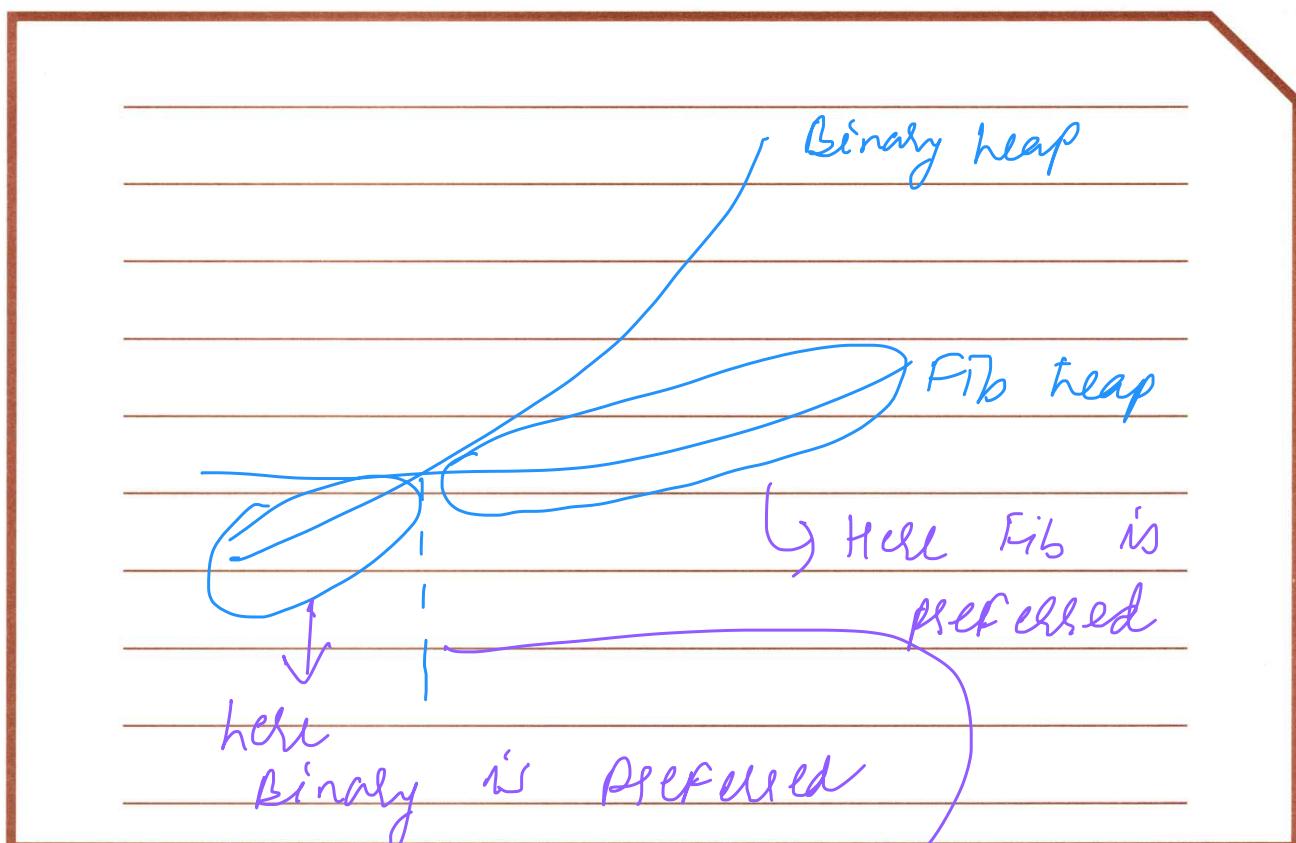
- Initialize Priority Queue $O(n)$

- 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 mins	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
m Decr. keys	$O(m \log n)$	$O(m \log n)$	$O(m)$
assuming there are as many edges as there are nodes			
Total	$O(m \log n)$	$O(m \log n)$	$O(m + n \log n)$
Sparse graph	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$
Dense graph	$O(n^2 \log n)$	$O(n^2 \log n)$	$O(n^2)$

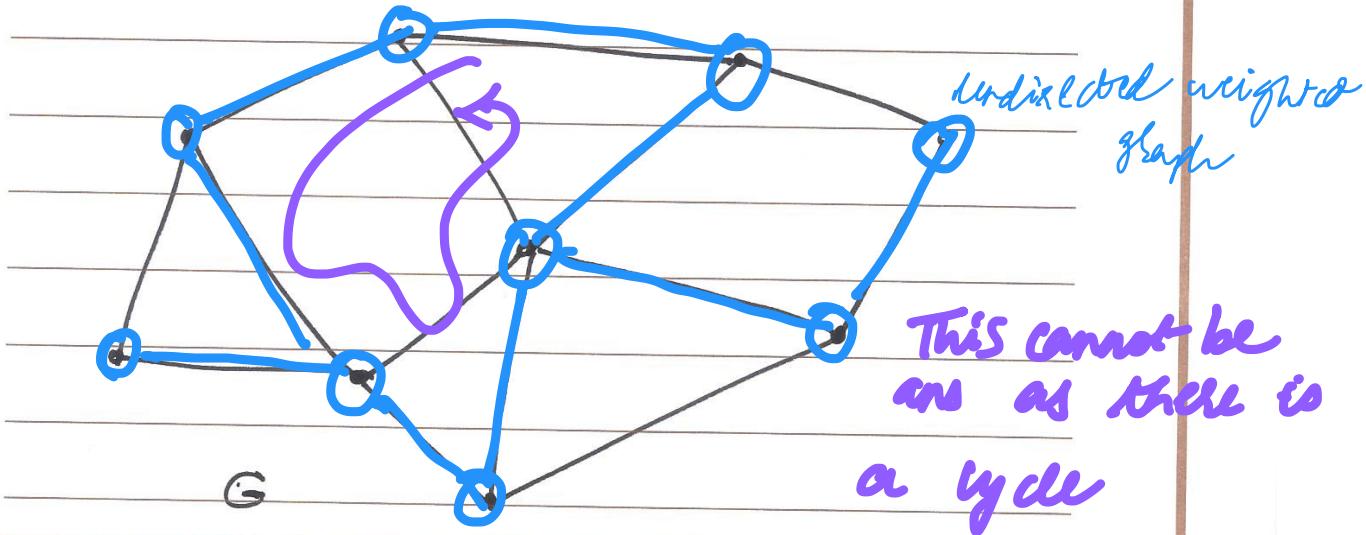
that does
not have that
many edges



But we do not know
where this cut-off
point is

Problem Statement

Find minimum cost network that connects all nodes in G .



→ no cycles

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 algorithm

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.

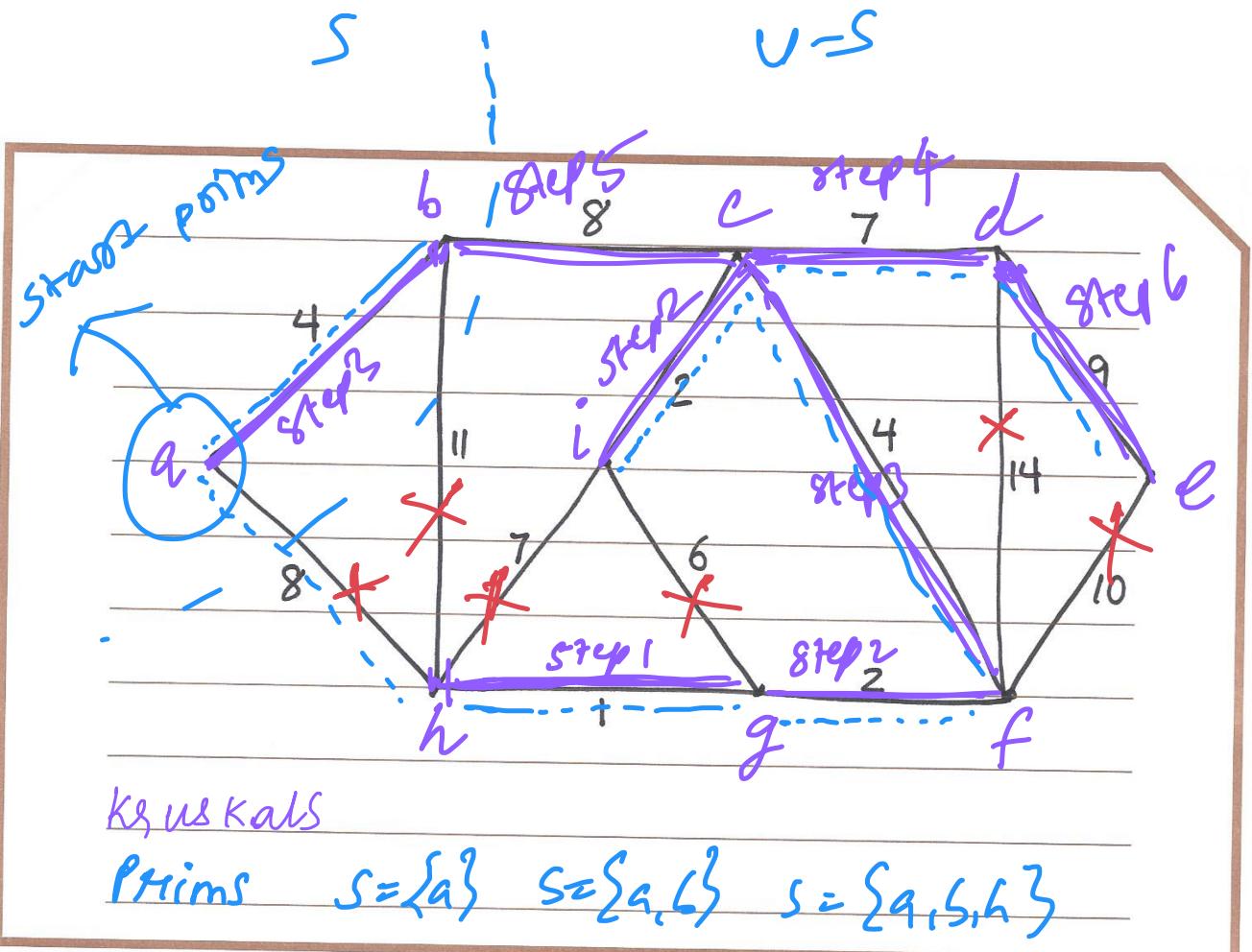
At each step, grow S by one node, adding the node v that minimizes the attachment cost.

Prim's algorithm

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

Backward version of Kruskal

Reverse - Delete



Kruskals

Prims $S = \{a\}$ $S = \{a, b\}$ $S = \{a, b, h\}$

$$S = \{a, b, g, h\} \quad S = \{a, b, g, h, f\}$$

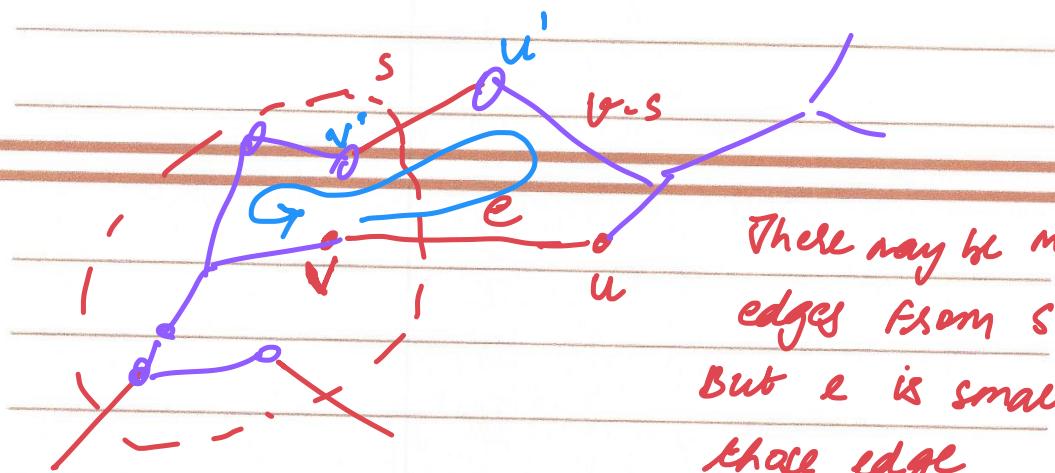
$$S = \{a, b, g, h, f, c\} \quad S = \{a, b, g, h, f, c, i\}$$

$$S = \{a, b, g, h, f, c, i, d\} \quad S = \{a, b, g, h, f, c, i, d, e\}$$

Reverse - Del

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 e .

Considering all edge costs are unique



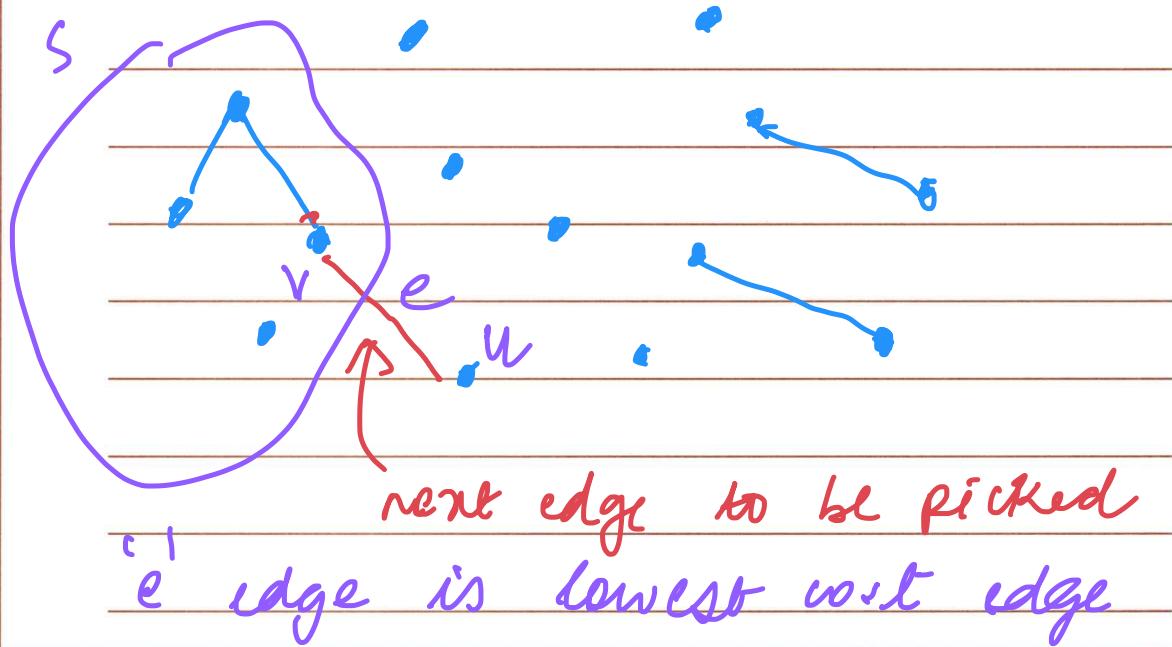
There may be many
 edges from S to $V-S$
 But e is smallest of
 those edge

PROOF OF CONTRADICTION

\therefore To avoid cycle we would remove $v'u'$ because
 its cost is more than vu

\therefore we would take vu instead of $v'u'$

Kruskal's proof of correctness

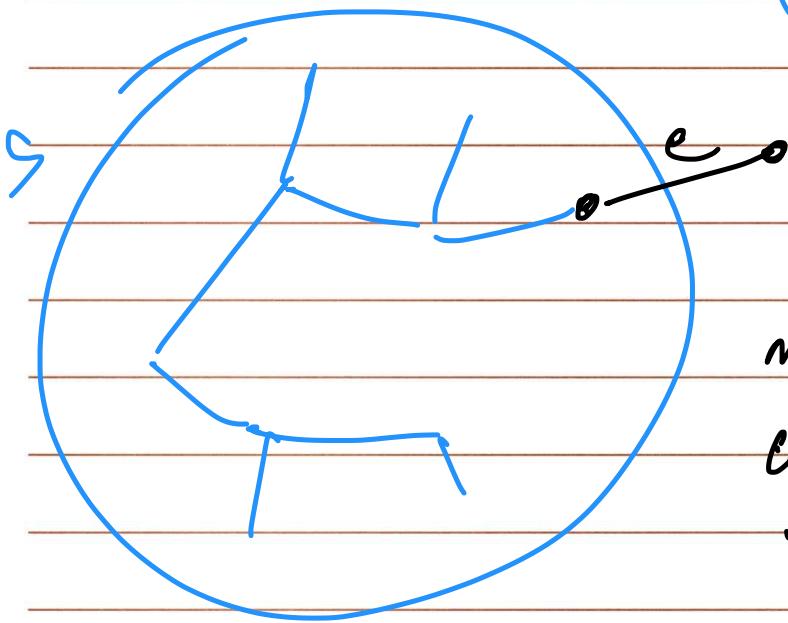


between S to $V-S$

see Page 13 Fact
works everytime

Hence Kruskal's gives min
Spanning
Tree

Pharm



$\sqrt{-\zeta}$

in conclusion

cost 60^n

S and V-S

Page 13 FACT again

Hence Prim's also finds minimum spanning tree

Rev Del

FACT: highest cost edge in cycle
cannot belong to minimum
spanning tree

Rev del 10 summary alg i.e useless and not in minimum spanning Tree

Hence it also finds min spanning Tree

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

$S = \text{NULL}$

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 $u \in \text{Adj}(v)$

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

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

end for

end while

Kruskal's

Create an empty set for each node
 $A = \text{Null}$

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

In initially there are ' n ' independent sets at end we get 1 set

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.

takes null and returns belongs to set

perform union of two sets

pointer implemented for

1 edges have to go b/w 2 diff sets and then merge 2 sets

Implementation of Kruskal's

$A = \text{Null}$

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

end for

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

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

$$O(n + m \log m + m \log n) = O(m \log m)$$

Bims

$O(m \log n)$

Kruskals

$O(m \log n)$

highest m can be n^2

every node

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

$$\subset O(m \log n)$$

connected to

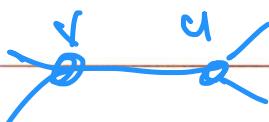
every other node

Reverse Delete

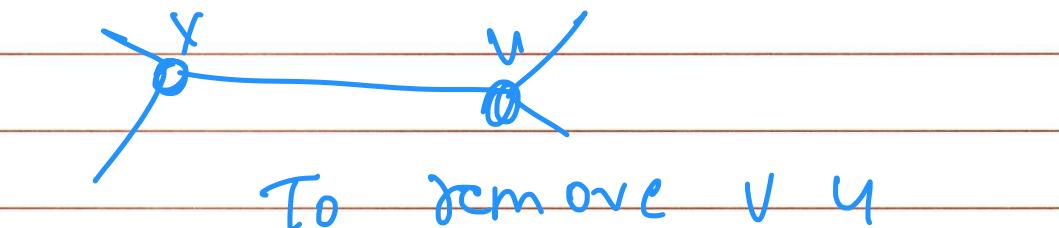
sort edges in dec order

$O(m \log n)$ For x in edges: $\leftarrow O(m)$

$O(m \times r)$ if x does not disconnect graph
remove x



$\cancel{O(m^2)}$



we run bfs starting at v and see if we reach u
 If we can reach u then graph is not disconnected and we remove edge $v \& u$

Clustering

Defn Given a set U of n objects $p_1..p_n$
 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 K clustering of U is a partitions of U into \underline{K} nonempty sets $C_1 .. C_K$

Defn The spacing of a K clustering is the minimum distance between any pair of points lying in different clusters

Problem statement

Given a set of n objects as described above , find a K-Clustering with maximum spacing



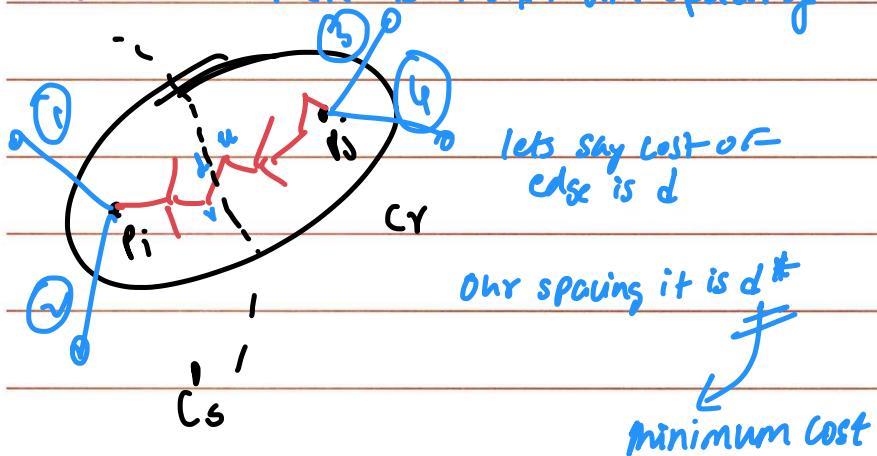
works like Prims

just leave out last $K-1$ edges

dist b/w cluster \uparrow

dist of points in cluster \downarrow

Proof that there is maximum spacing



$$d < d^*$$

Gd

Recap to
Kruskals?

Kruskals drops last k-1 edges those are
the one's marked in blue

Kruskals choose d' over ①②③④
this means

$$d < \textcircled{1} \textcircled{2} \textcircled{3} \textcircled{4}$$

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

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

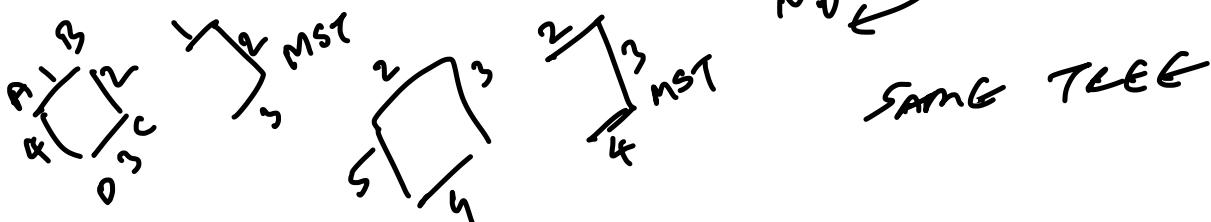
5.

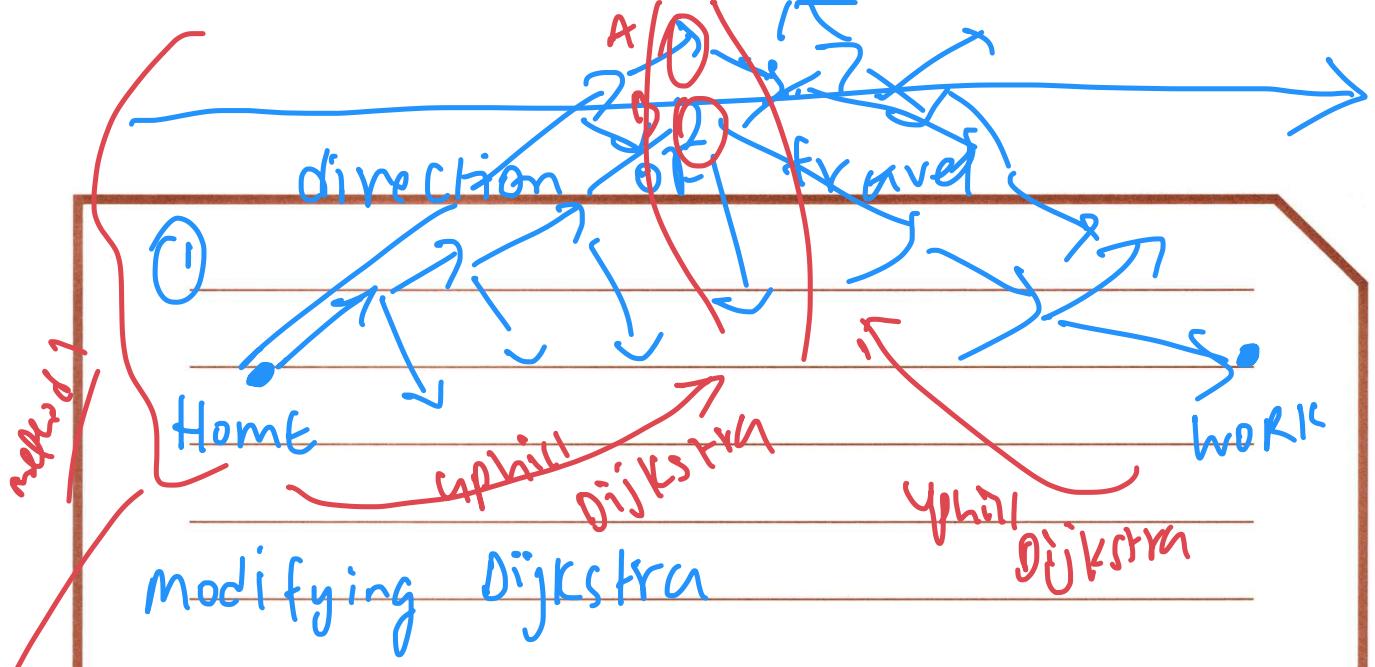
(a): Suppose we are given an instance of the Minimum Spanning Tree problem on a graph G . Assume that all edges costs are distinct. Let T be a minimum spanning tree for this instance. Now suppose that we replace each edge cost c_e by its square, c_e^2 thereby creating a new instance of the problem with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



(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.





while finding uphill node

during relaxation only relax nodes
that are uphill ignore all
nodes downhill

uphill
downhill

return min(A, B)

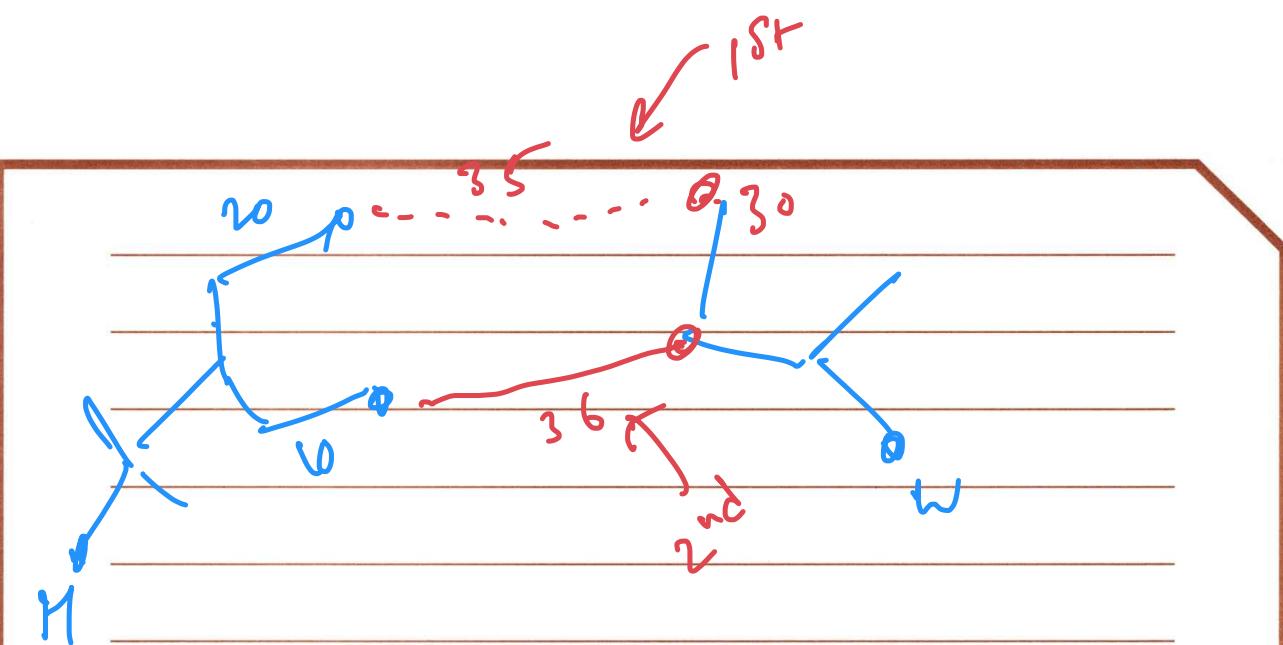
as we know dist of (Home - A) \min
(Home - B)

method 2

convert undirected graph \rightarrow directed
graph

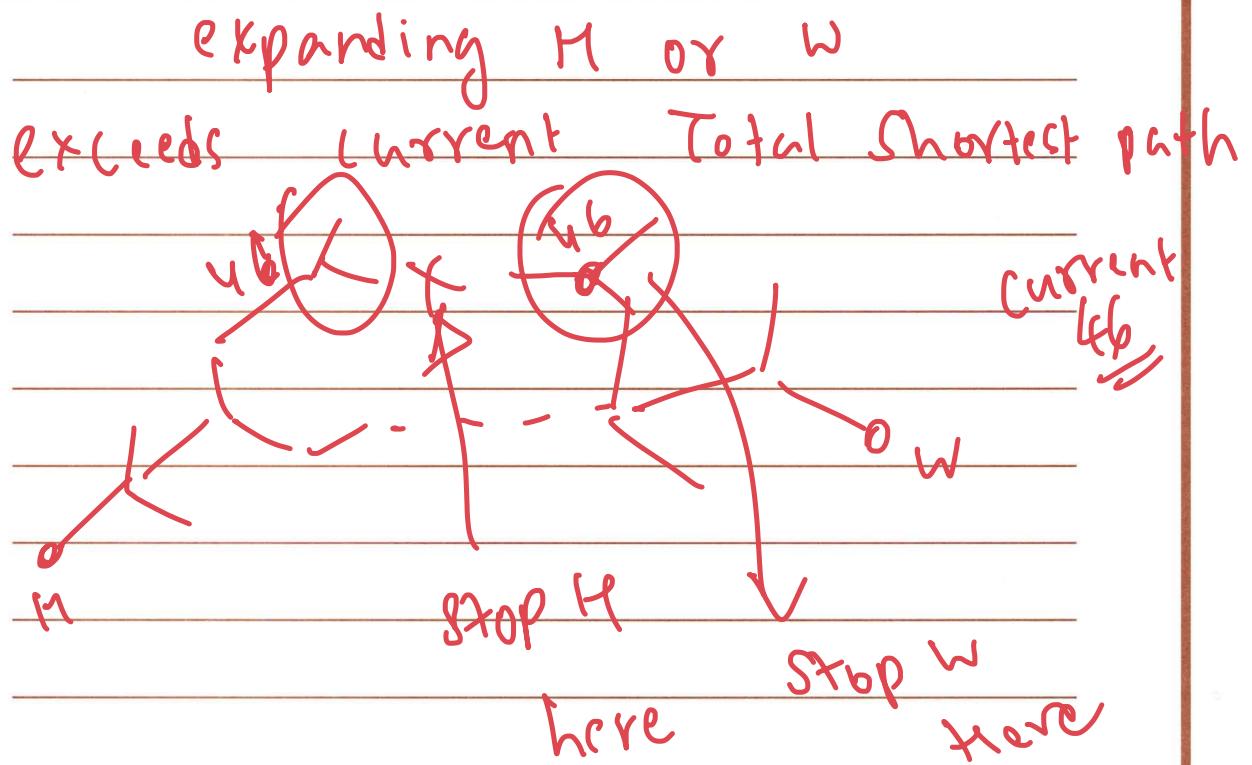
Then on moving \nearrow will guarantee we are going
upward

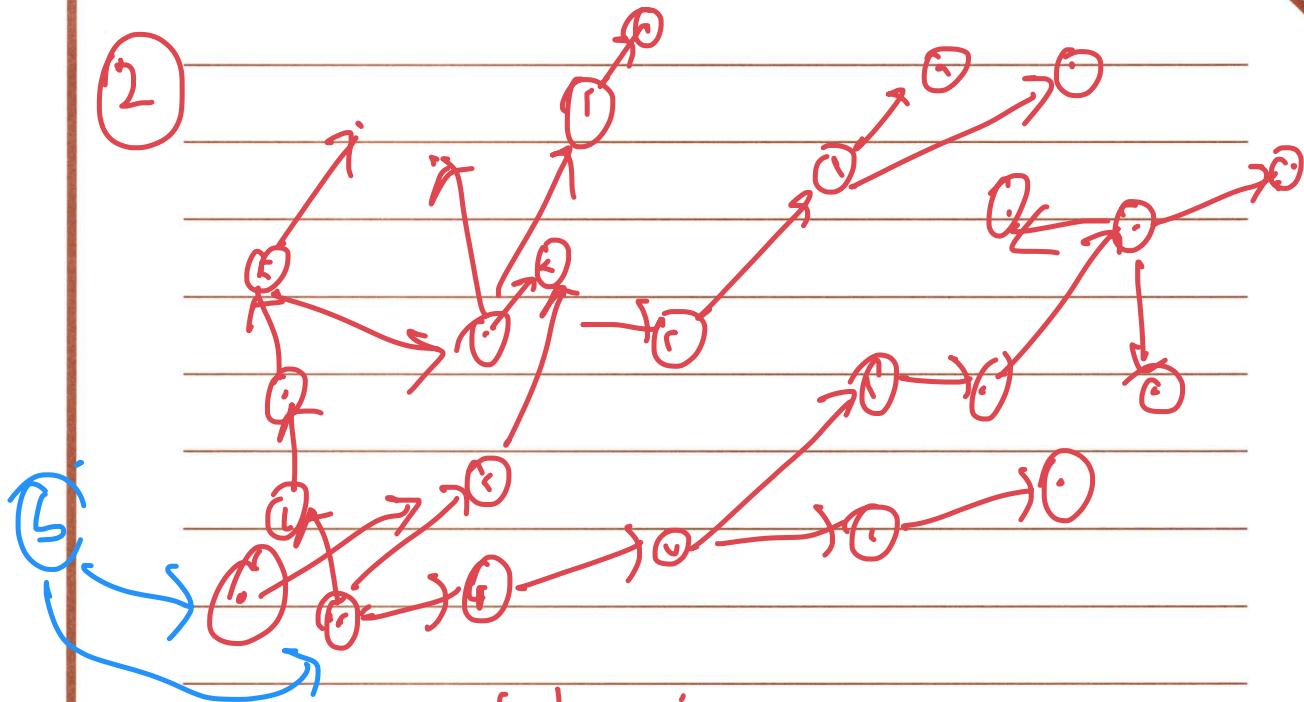
initially this graph is undirected with elevations of each
points



1st point might not always bc best

stop parallel execution when





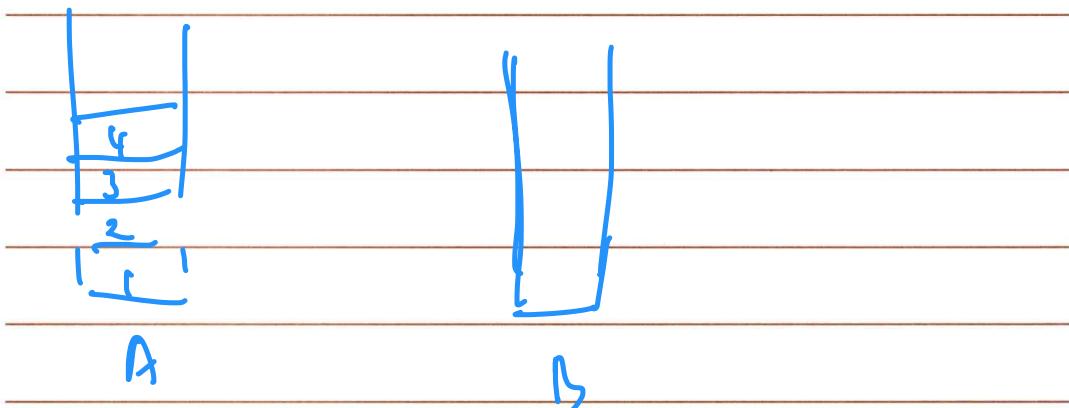
Soln1 run ' n ' dijkstra from r starting nodes

and ret min of all dists

Soln2 add new node 5 and connect it to all starting nodes

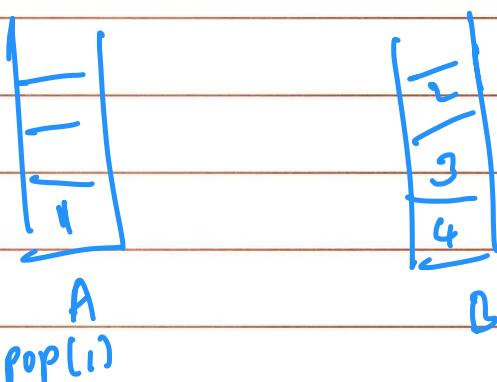
and run dijkstra's only once then,

⑤ Implement Queue using two stacks (FIFO)



To dequeue we transfer
 A: pop(4) push(4) in B
 pop(3) push(3) in B

pop(2) push(2) in B



now dequeue will be pop() from B

n enqueue \rightarrow n pushes \rightarrow n

1 dequeue \rightarrow $(n \text{ pop})$
 $(n \text{ push})$
 $(n \text{ pop})$

Total cost of n operation = $3n + 1$

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

one hand
4 @

④ 3, 5, 2, 8, 1, 5, 2, 7

