March 8

Hint for arbitrage

(1) $a > b$ $\ln a > \ln b$

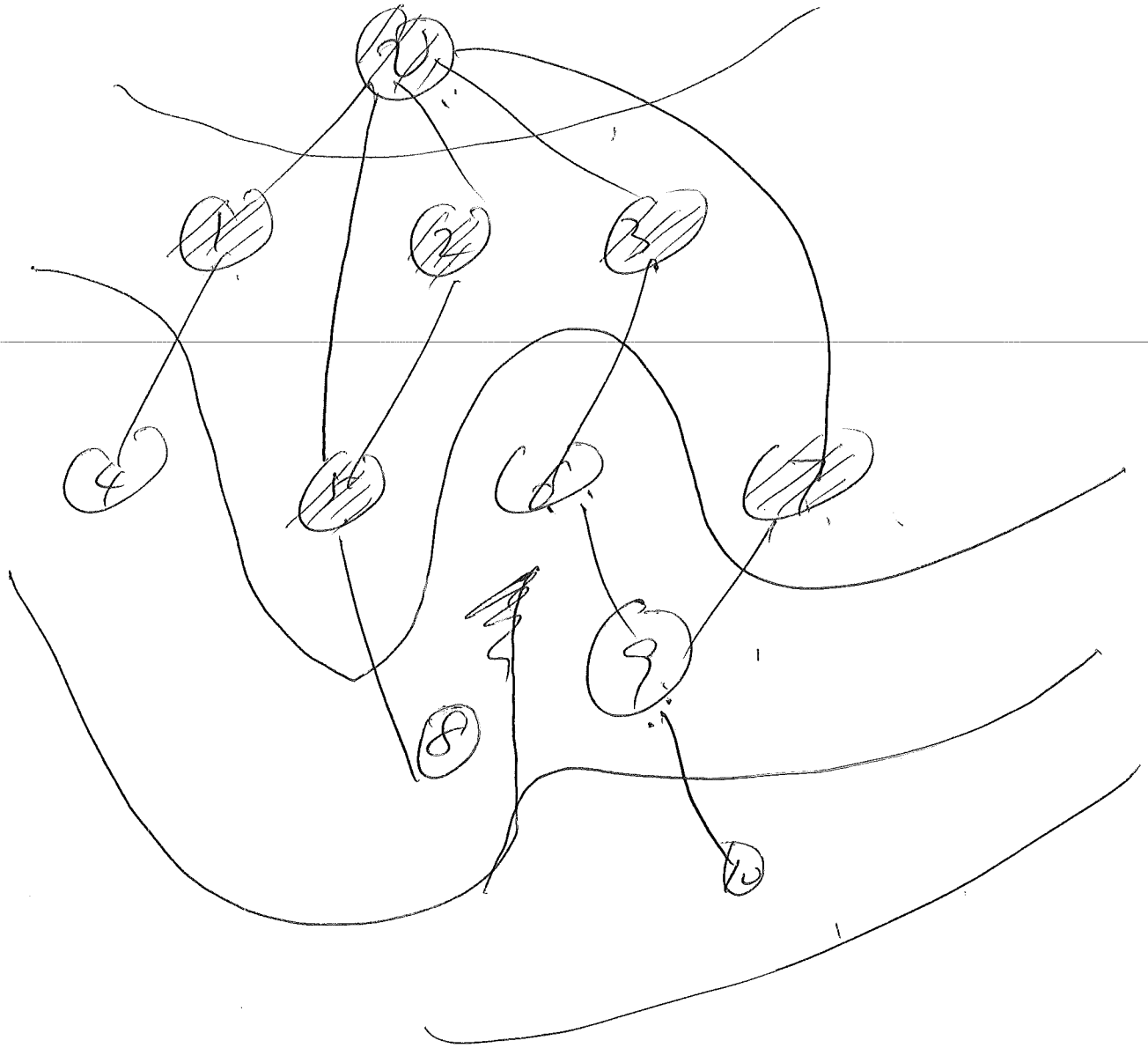
(2) max $ab \Leftrightarrow$ max $\ln(ab) \Leftrightarrow$ max $\ln a + \ln b$

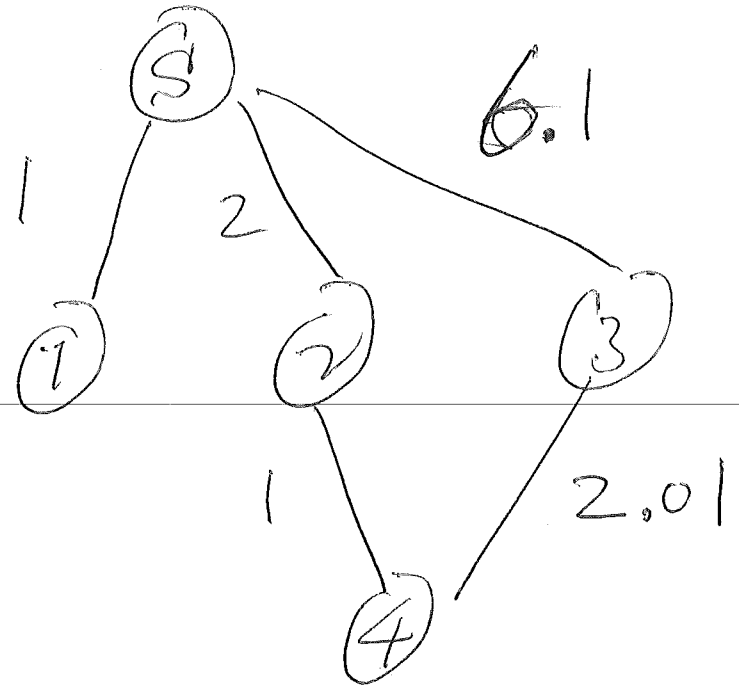
Breadth First Search

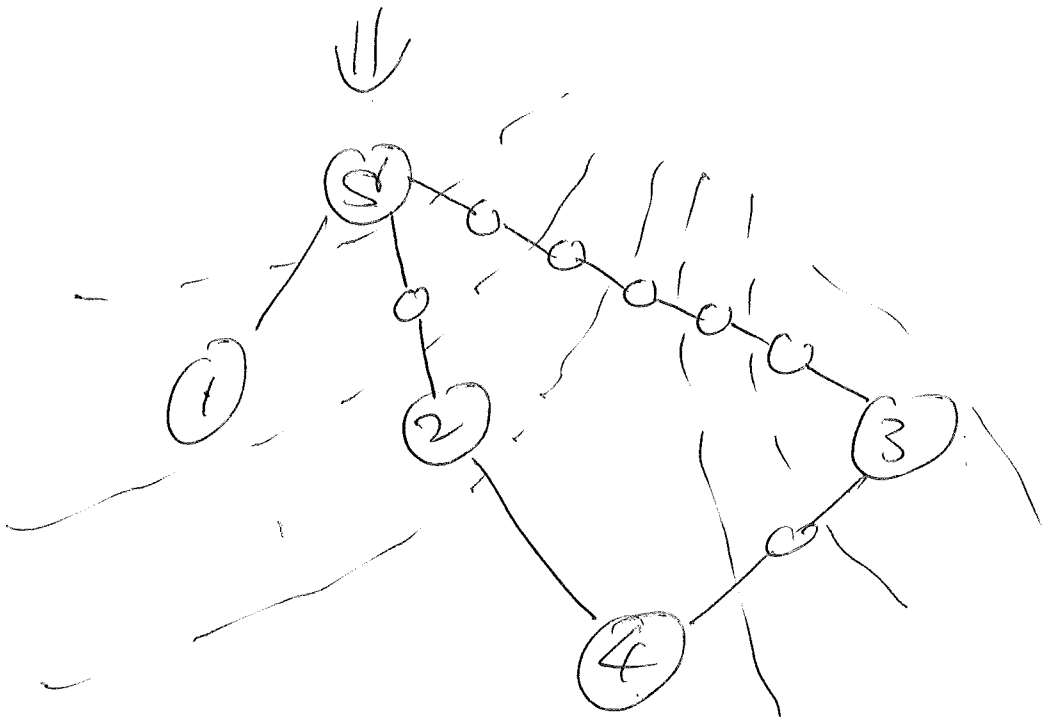
use a queue

BFS (G, v)

mark all vertices as unvisited
mark v
enqueue (v)
~~for all edges~~
while Q is not empty,
u = dequeue (Q)
For all ~~edges~~ arcs (u, w)
if w is unvisited
mark w
enqueue (w)

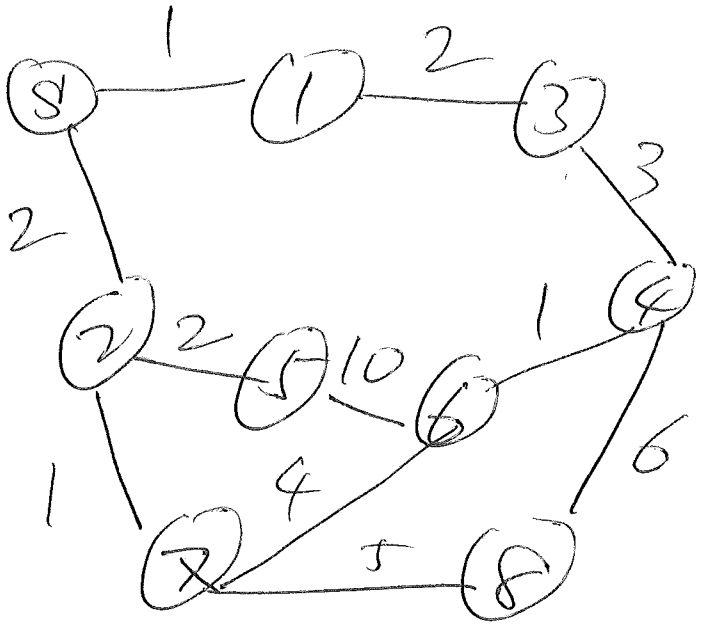

A
B
C
D
E
F
G
H
I
J


S
6.1
1
2
1
2
3
1
2.01
4


S
1
2
3
4

Dijkstra's algorithm

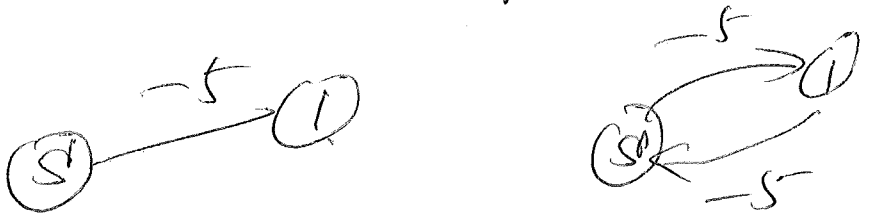
Strategy: find the shortest path of the vertices in the order of how far they are from the source s, i.e., the closer vertices their shortest paths will be discovered first



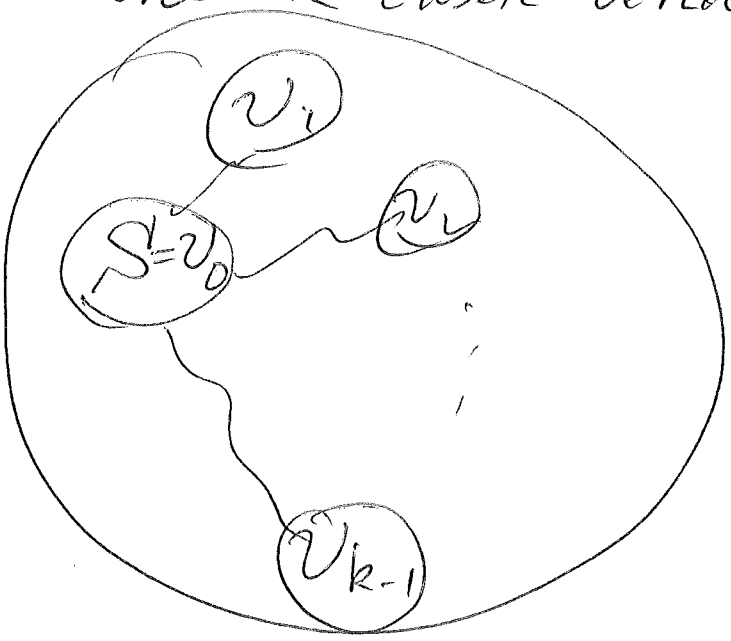
s, 1, 2, 3, 7, 5, 4, ...

Basis: we need to find the vertex closest to the source.

if G has non-negative edges costs, then the closest vertex to the source is the source s itself.

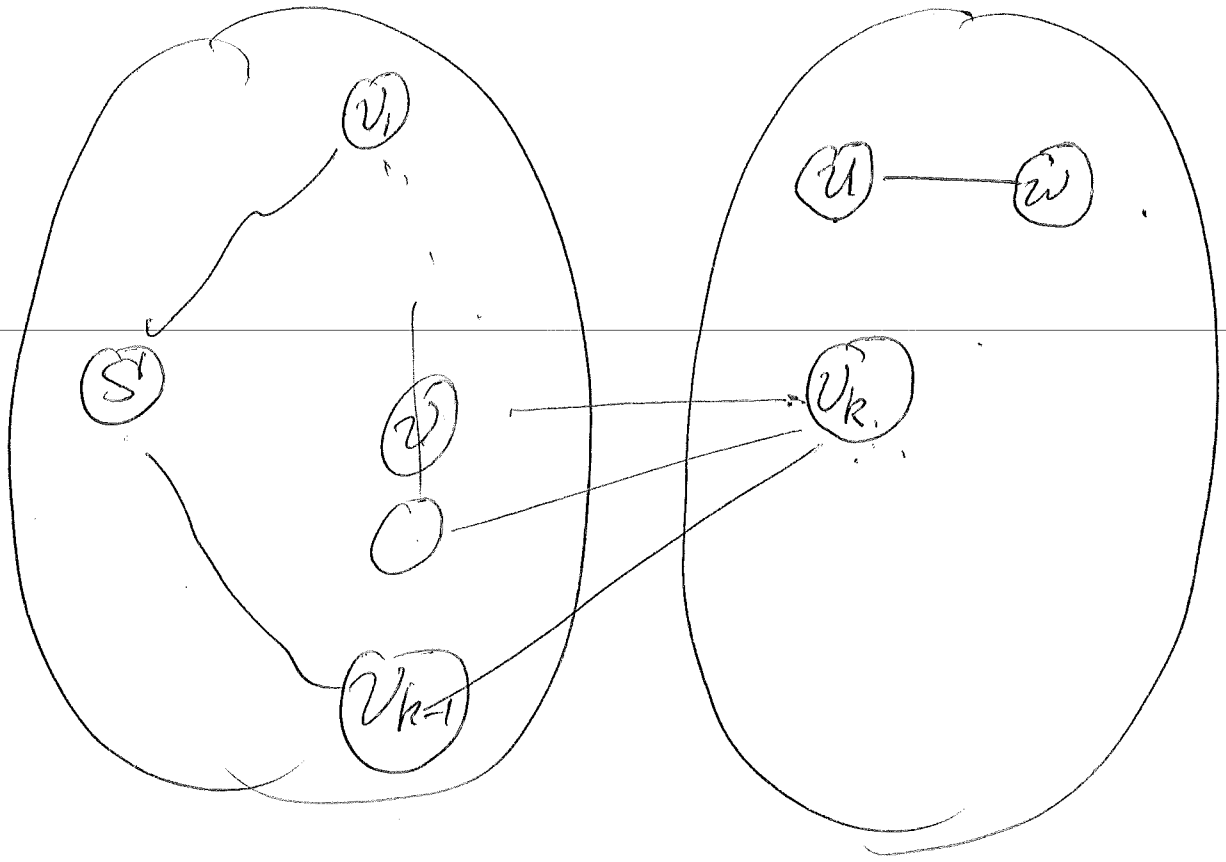


I.H. Assume we have the shortest paths of the k closest vertices to the source s



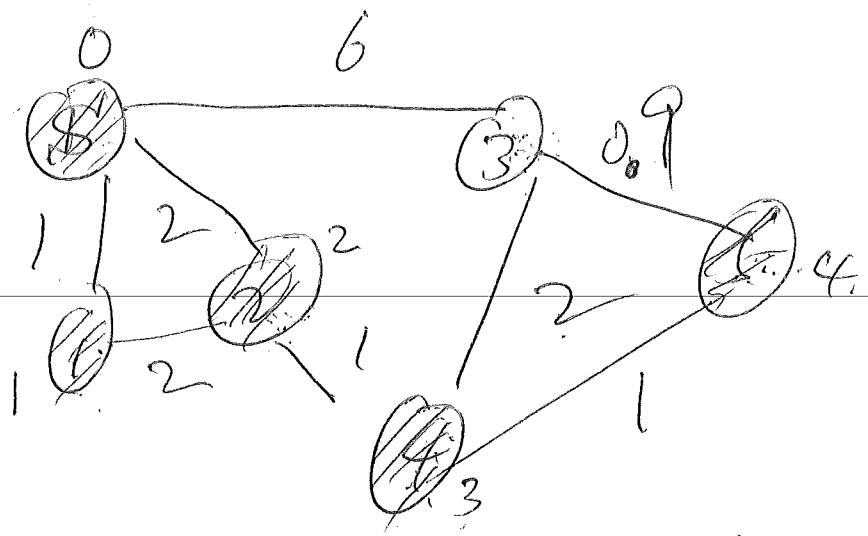
I.S. How can we find the $(k+1)^{th}$ closest?

$S' = \{s, v_1, \cdots, v_{k-1}\}$ $\quad V - S'$



① v_k must be adjacent to a vertex in S'

$$\min_{v_k} \{ sp(s,v) + c(v,v_k) \mid v \in S' \}$$



$O(|V|^2 + |E|)$.

~~$O(|V||E|)$~~

$sp(s,s) = 0$ $\quad S = \{s\}$

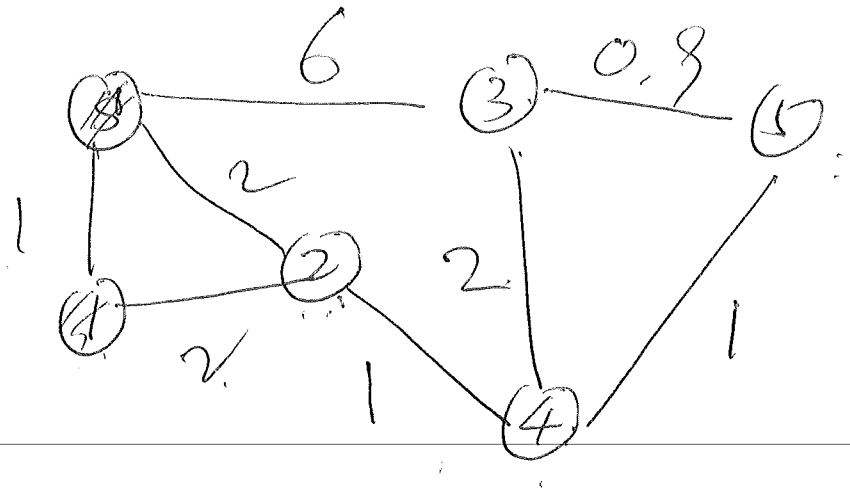
$sp(s,1) = 1$ $\quad S = \{s, 1\}$

$sp(s,2) = \phi 2$ $\quad S = \{s, 1, 2\}$

$sp(s,4) = 3$ $\quad S = \{s, 1, 2, 4\}$

$sp(s,5) = 4$ $\quad S = \{s, 1, 2, 4, 5\}$

$sp(s,3) = 4.9$ $\quad S = \{s, 1, 2, 4, 5, 3\}$



Use a heap to store the ~~bare~~ shortest path to get to a vertex so far.

min H $\boxed{SP(S,S)=0}$

$SP(S,S)=0$

$\boxed{\begin{array}{l} SP(S,1)=1 \\ SP(S,2)=2 \\ SP(S,3)=6 \end{array}}$

$SP(S,1)=1$

$\boxed{\begin{array}{l} SP(S,2)=2 \\ SP(S,3)=6 \end{array}}$

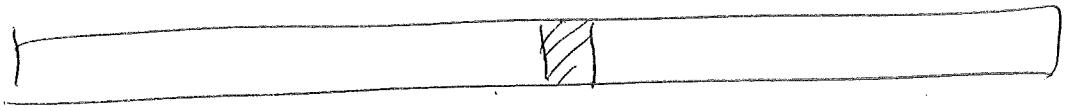
$SP(S,2)=2$

$\boxed{\begin{array}{l} SP(S,4)=3 \\ SP(S,3)=6 \end{array}}$

$sp(s,4) = 3$

$sp(s,5) = 4$

$sp(s,3) = 5$



$O(|V|+|E|)\lg|V|$

Minimum Spanning Tree

Given a graph G of non-negative edgeweights, find a connected subgraph T of G including all vertices that minimizes the total edges costs of T

Two algorithms

(1) grow the tree by adding one vertex to the tree at a time

(2) maintain a spanning forest by adding an edges at a time

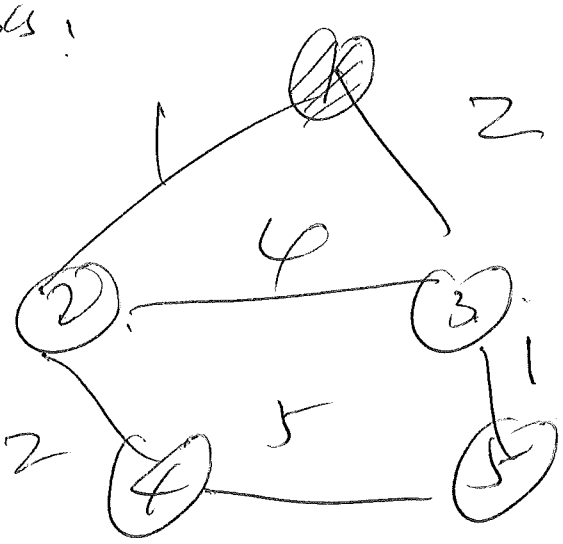
Strategy 1:

We will maintain a subtree of the MST.

In each iteration, we increase the number of nodes in this subtree by 1

The algorithm terminates when the tree has $|V|$ vertices

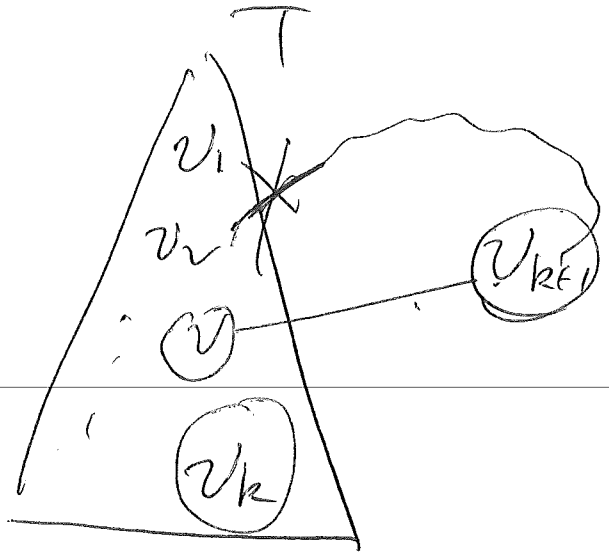
Basis:



Any singleton node is a subtree of the final MST

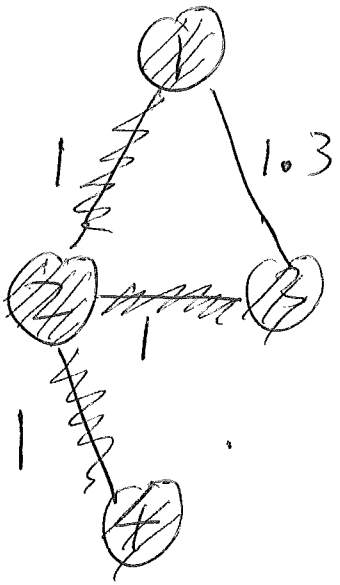
I.H. Assume we have a subtree of the MST with k nodes,

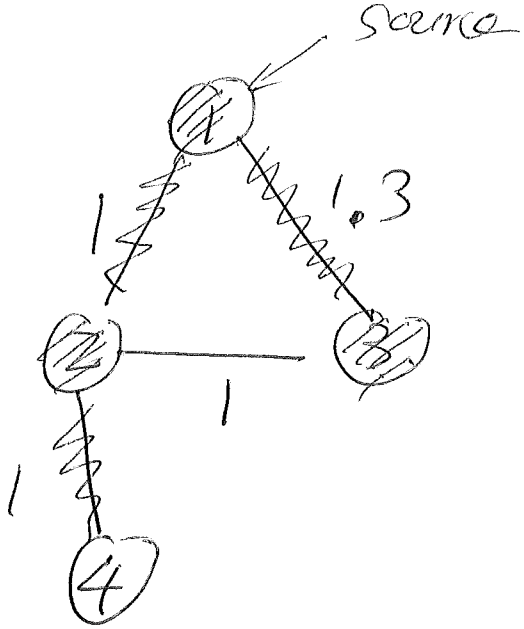
I.S. How to grow this subtree to have $k+1$ nodes



Observation: v_{k+1} has to be adjacent to one of the vertices in the current tree T.

MST vs. Dijkstra

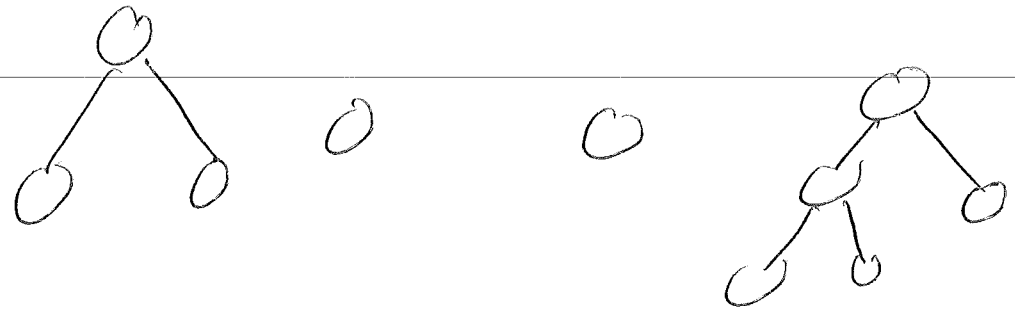




Shortest path tree

Strategy 2: maintaining a spanning forest

A forest is a collection of trees.



a forest of 4 trees

Q: for a k-tree forest of n vertices ($k \leq n$), how many edges are there? $n-k$

$k=1$	$n-1$
$k=2$	$n-2$
⋮	
k	$n-k$
$k=n$	$0=n-n$

Minimum Spanning Forest of k-trees

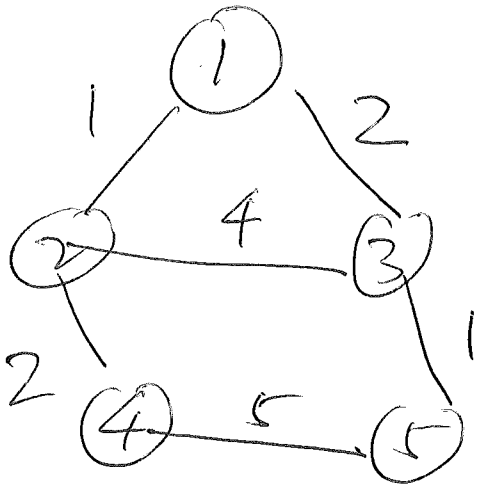
MST = MSF of 1 tree

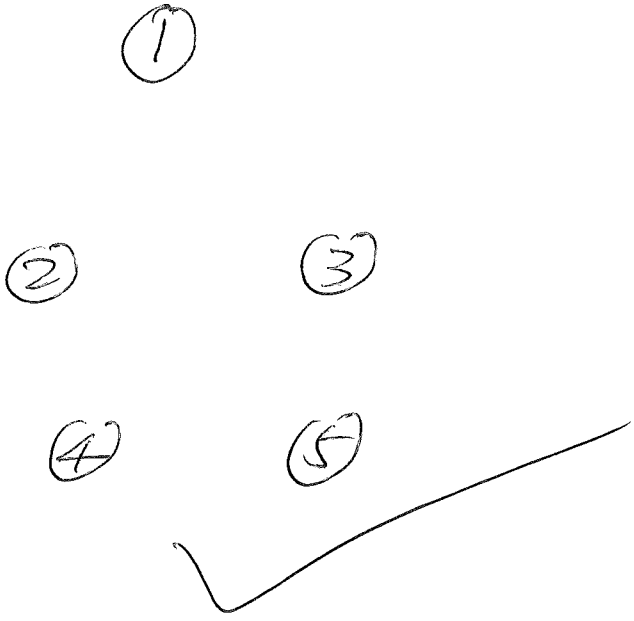
~~In each iteration~~, we will maintain a MSF

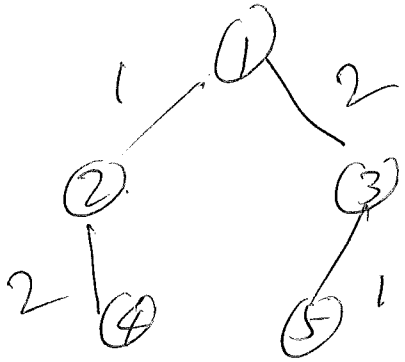
in each iteration, we will reduce the number of trees in the MSF by 1

The algorithm terminates when the MSF ⇒ MST

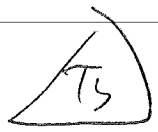
Basis:

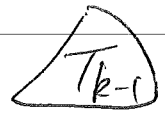






inter tree edge.
with minimum edge
costs





the catch:

① how to maintain a forest?

② how to merges trees in a forest?

Union-Find.

Given n elements $x_1, x_2, \cdots, x_n$

optimize:

1 ~~makeset(x): create a singleton set $\{x\}$~~

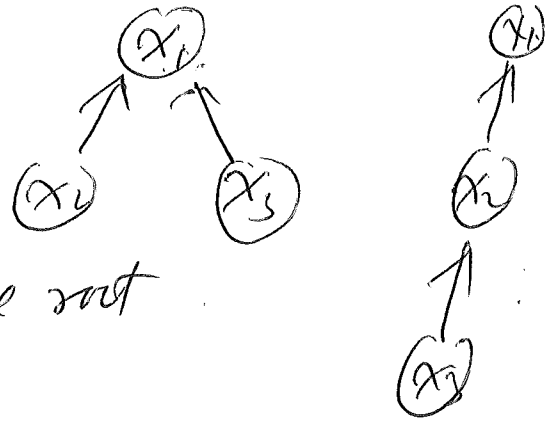
2 find(x): return the name of the set containing x

3 union(x, y): merge the ~~sets~~ two sets containing x and y

---

Solution:

a) each set is a tree

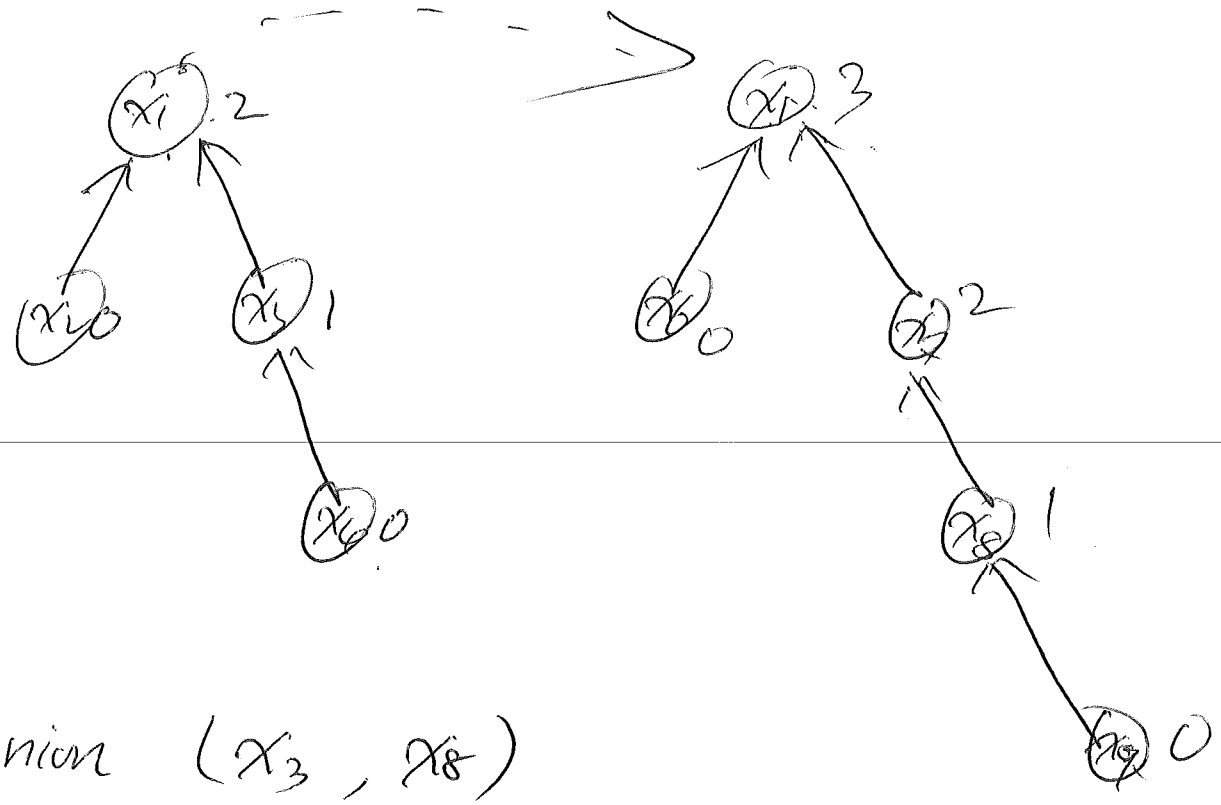


the name of the set is the root

b) union by rank

For each node, ~~we~~ its rank is the height of its sub-tree

in union, always connect the shorter tree to the taller tree



union (x_3, x_8)

$O(\log n)$ for find and union

③ path compression