Analysis of Algorithms

V. Adamchik

Lecture 5

CSCI 570

University of Southern California

Greedy Algorithms

Reading: chapter 4

Homework - 2

We do not have enough time to cover the master theorem this week. Therefore, you do not need to solve the last homework problem #5, it will be a part of the next Hw-3 assignment.

The Minimum Spanning Tree

Find a spanning tree of the minimum total weight.

MST is fundamental problem with diverse applications.

Kruskal's Algorithm

algorithm builds a tree one EDGE at a time.

- Sort all edges by their weights.
- Loop:
  - Choose the minimum weight edge and join correspondent vertices (subject to cycles).
  - Go to the next edge.
  - Continue to grow the forest until all vertices are connected.

Union-Find

O(1)

A.C.

Sorting edges – O(E log E)

Cycle detection – O(V) for each edge

~~binary heap~~

Total: O(V*E + E*log E)

Prim's Algorithm

algorithm builds a tree one VERTEX at a time.

- Start with an arbitrary vertex as a sub-tree C.
- Expand C by adding a vertex having the minimum weight edge of the graph having exactly one end point in C.
- Update distances from C to adjacent vertices.
- Continue to grow the tree until C gets all vertices.

Fib. heap

deleteMin – O(V), for each vertex

update(decreaseKey) – O(log V), for each edge

O(1)

A.C.

Total: O(V log V + E log V)

Discussion Problem

1. Assume that an unsorted array is used instead of a binary heap. What would the running time of the Prim algorithm?

array : $O(V \cdot V + E \cdot 1)$

binary heap : $O(V \cdot \log V + E \cdot \log V)$

$E = O(V^2)$

2. Assume that we need to find an MST in a dense graph using Prim's algorithm. Which implementation (heap or array) shall we use?

$\boxed{\text{array} : O(V^2)}$

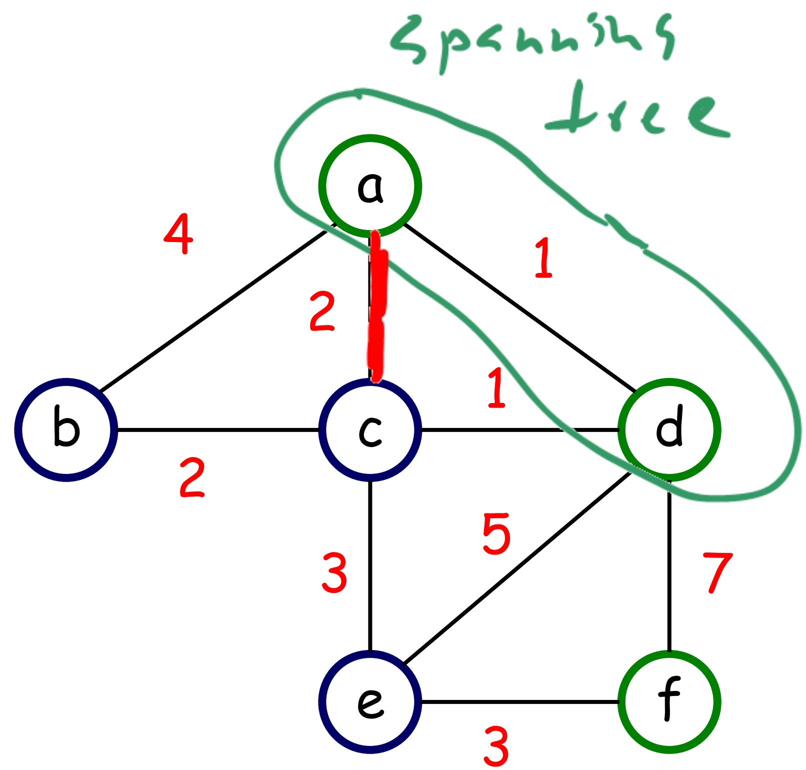
heap: $O(V^2 \log V)$

MST: Proof of the correctness

A cut of a graph is a partition of its vertices into two disjoint sets (blue and green vertices below.)

(a, c)

A crossing edge is an edge that connects a vertex in one set with a vertex in the other.



The smallest crossing edge must be in the MST.

MST: Proof of the correctness

Lemma: Given any cut in a weighted graph , the crossing edge of minimum weight is in the MST of the graph.

By contradiction.

$e \notin MST, f \in MST$

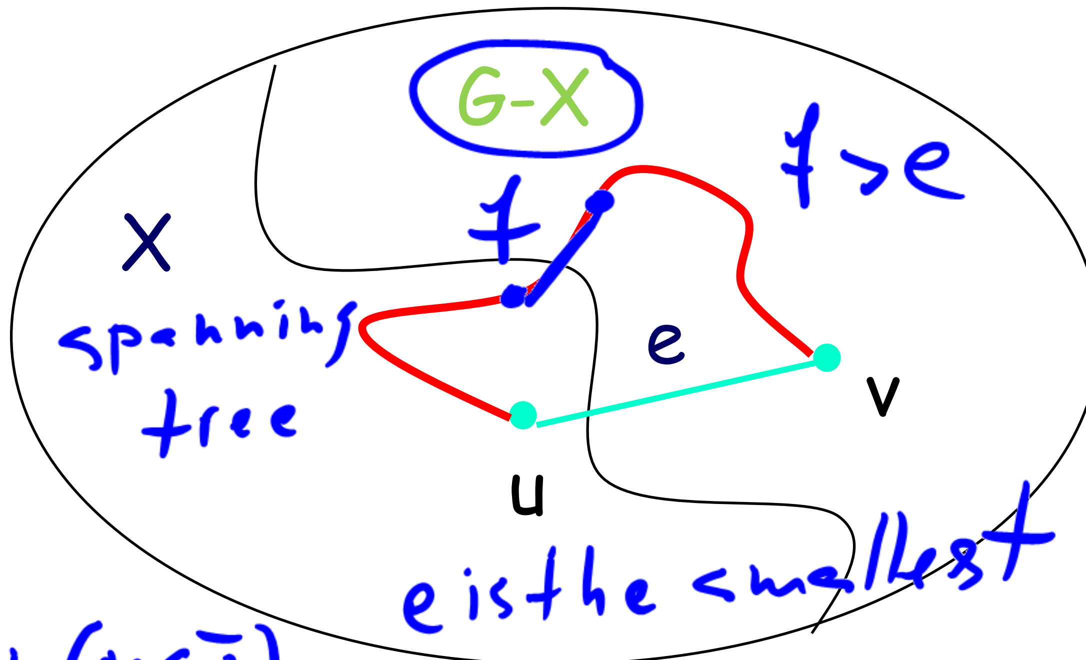
Add e to MST

MST will get a cycle

$MST_1 = MST - f + e$

$cost(MST_1) < cost(MST)$

contradiction



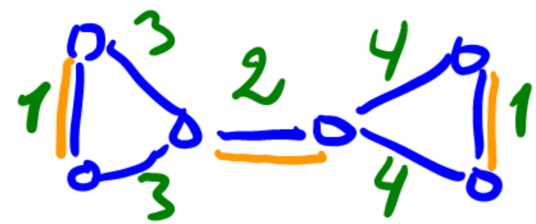
Review Questions

find an example

(T/F) The first edge added by Kruskal's algorithm can be the last edge added by Prim's algorithm.

find an example

(T/F) Suppose we have a graph where each edge weight value appears at most twice. Then, there are at most two minimum spanning trees in this graph.

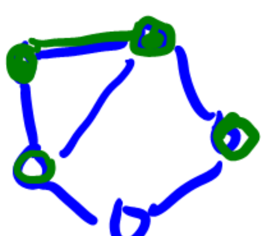


$V \to \infty$

$E = V + 1$

(T/F) If a connected undirected graph $G = (V, E)$ has $V + 100$ edges, we can find the minimum spanning tree of G in $O(V)$ runtime.

$V = 5$
$E = 6$



Prim's

$O\left((E+V)\underbrace{\log V}_{\text{size of the heap}}\right)$

heap size = 2

$|\log V| < \text{const}$

A*

The Shortest Path Problem

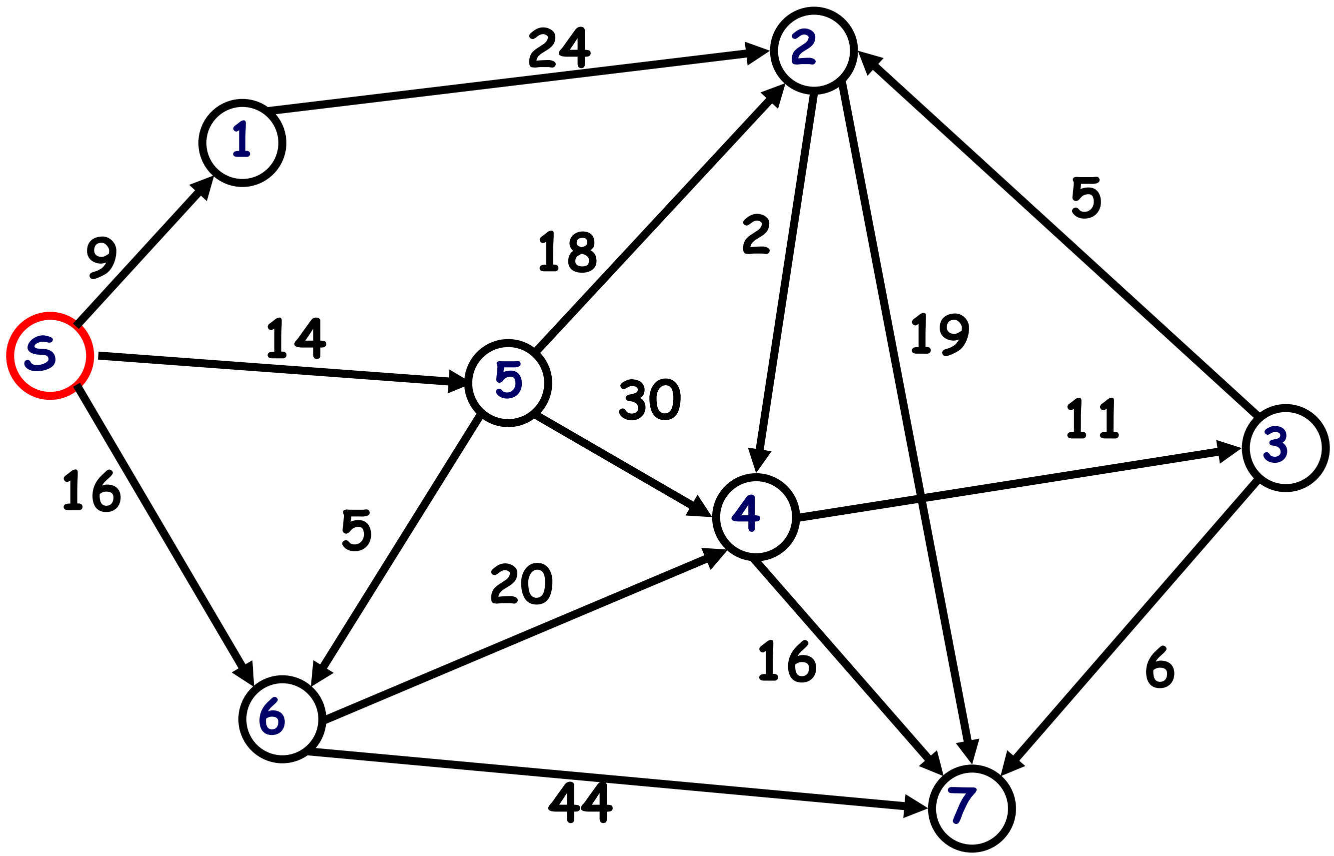
Edsger Dijkstra
(1930-2002)

SSSP

The Shortest Path Problem

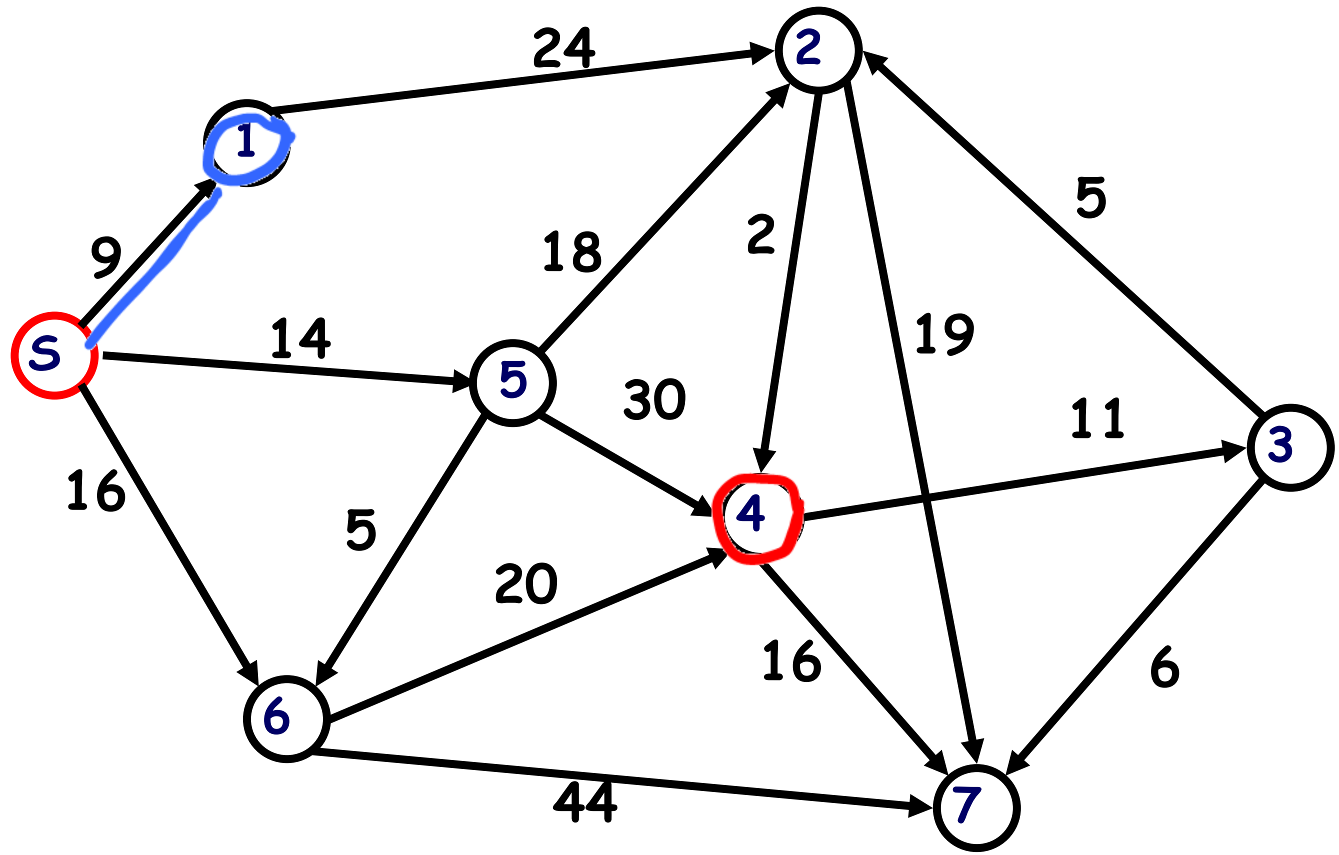
Given a positively weighted graph G with a source vertex s, find the shortest path from s to all other vertices in the graph.

$e \in \mathbb{R}^+$



The Shortest Path Problem

What is the shortest distance from s to 4?



Prim's

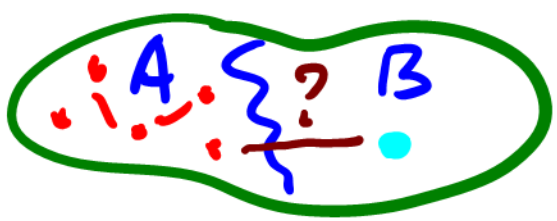
Greedy Approach

When algorithm proceeds all vertices are divided into two groups

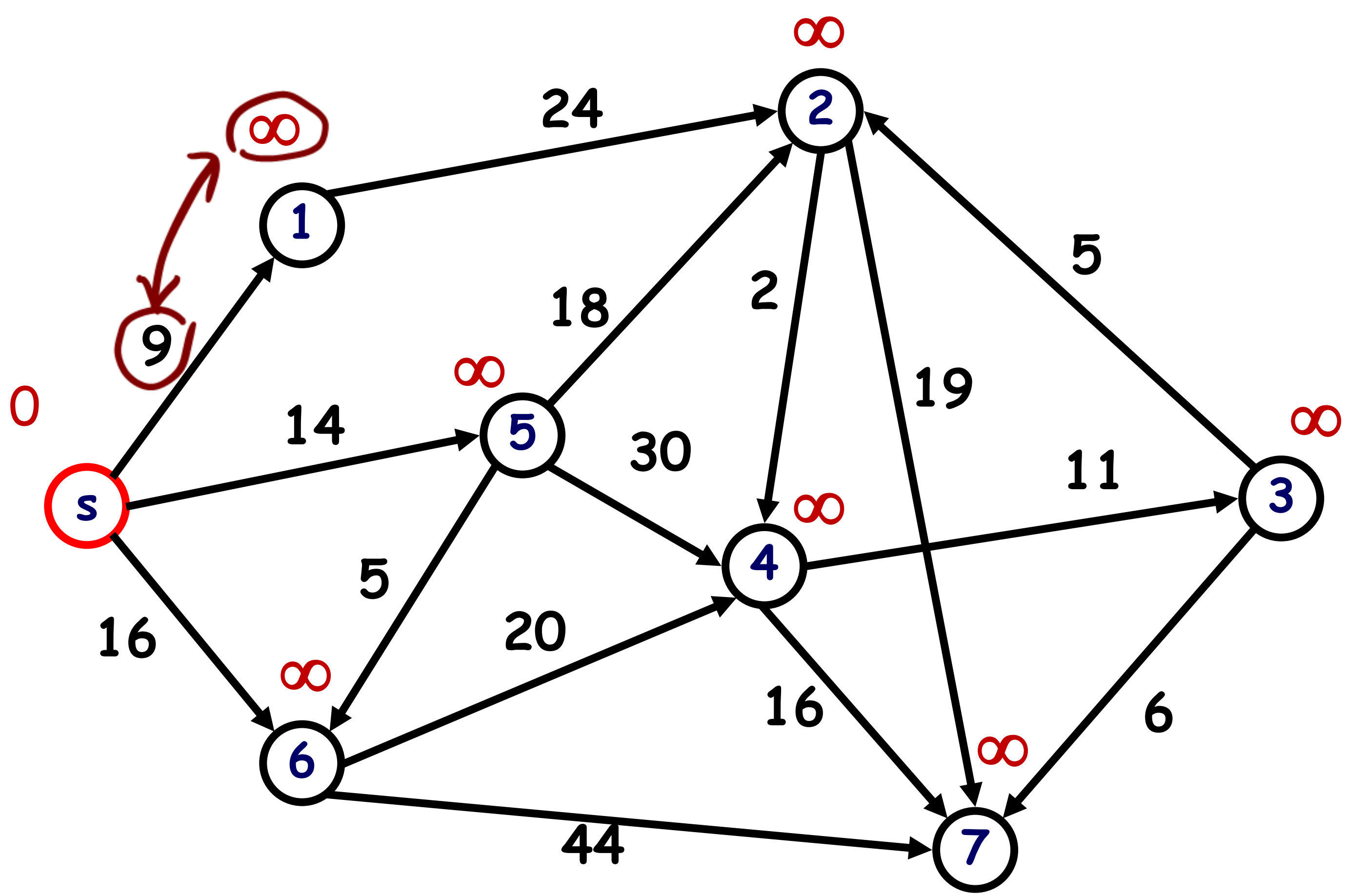
- vertices whose shortest path from the source is known
- vertices whose shortest path from the source is NOT discovered yet

Move vertices one at a time from the undiscovered set of vertices to the known set of the shortest distances, based on the shortest distance from the source.

G = A + B

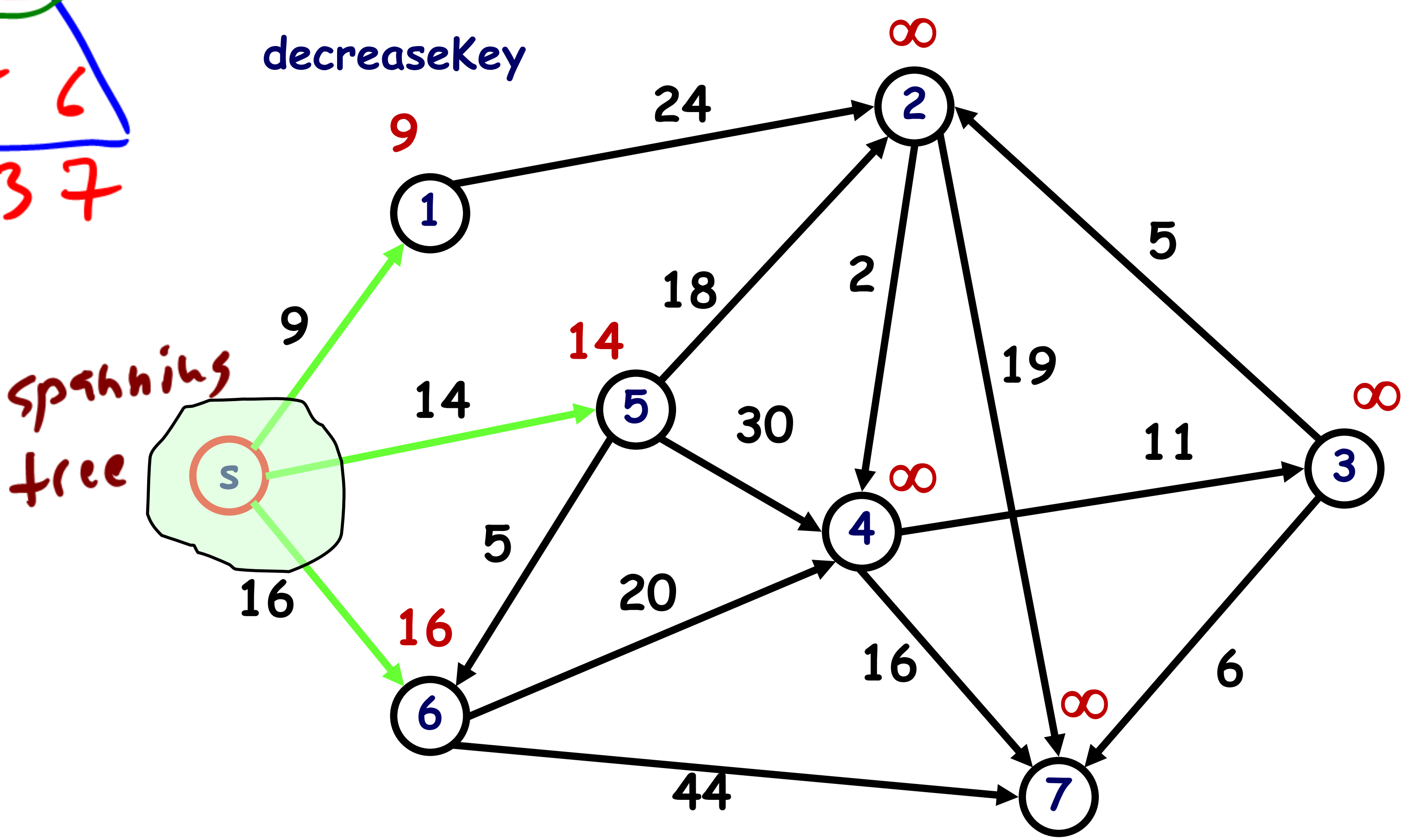


solution tree = s
heap = {1, 2, 3, 4, 5, 6, 7}



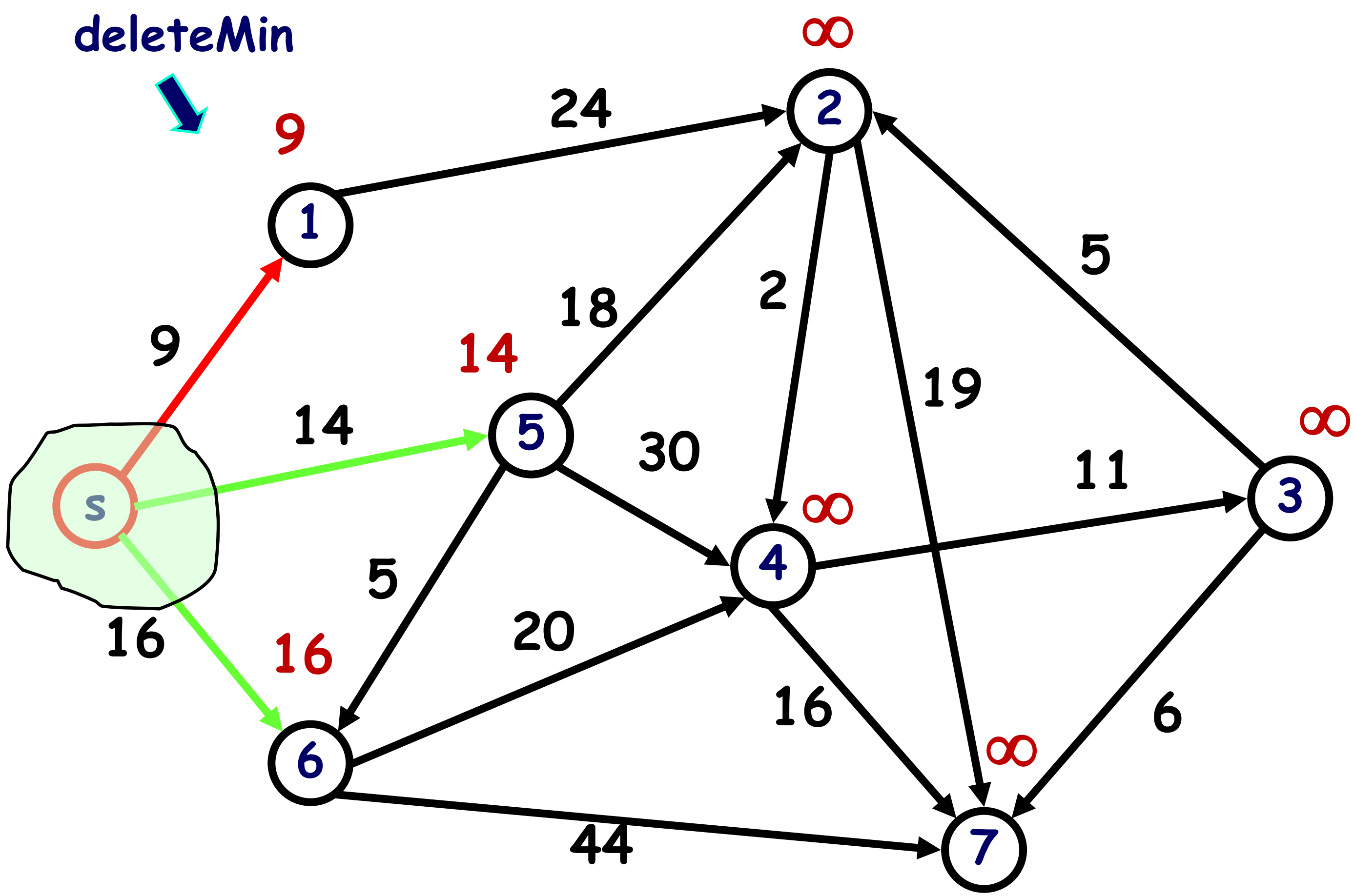
solution tree = { s }

heap = {1, 2, 3, 4, 5, 6, 7}

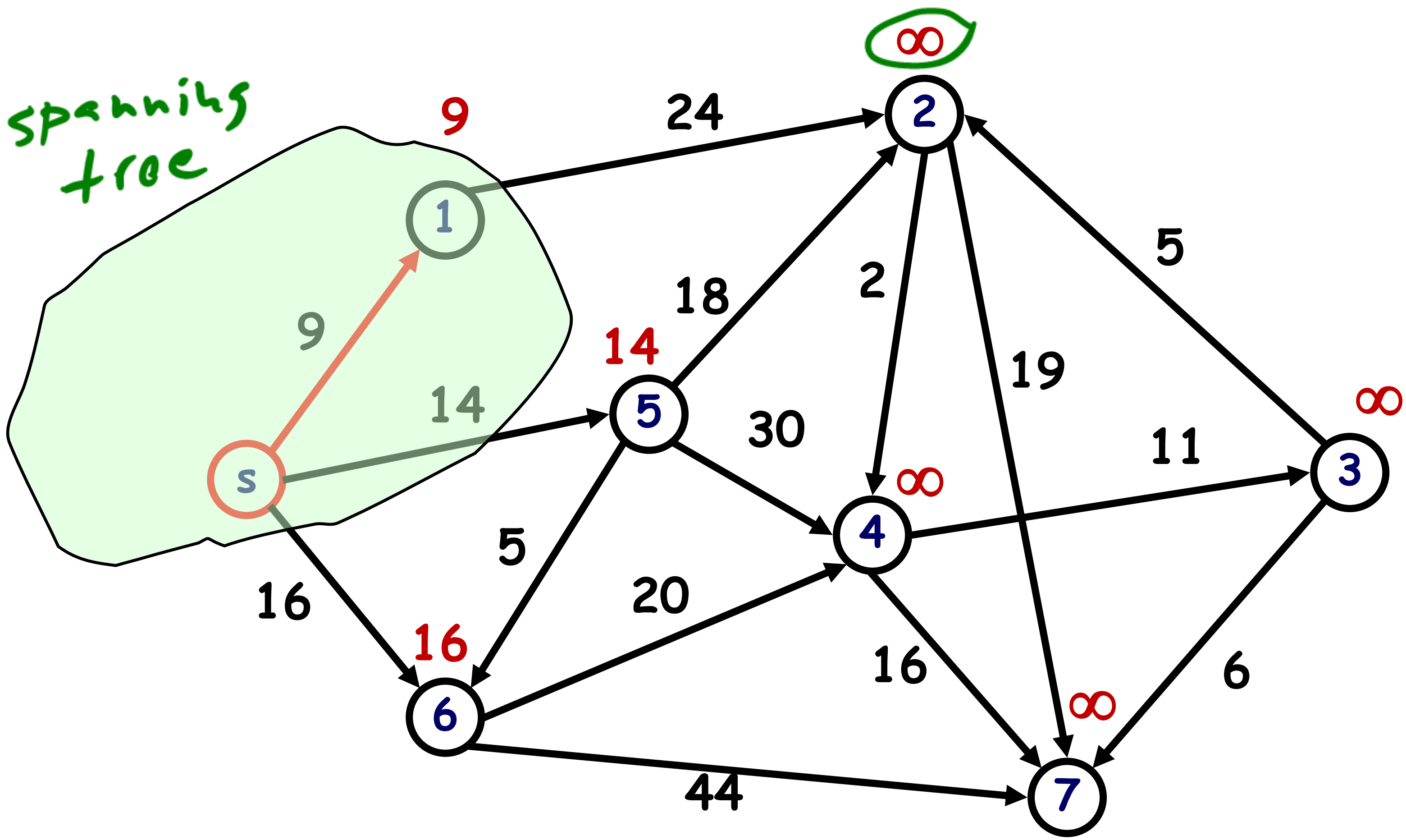


solution tree = { s }

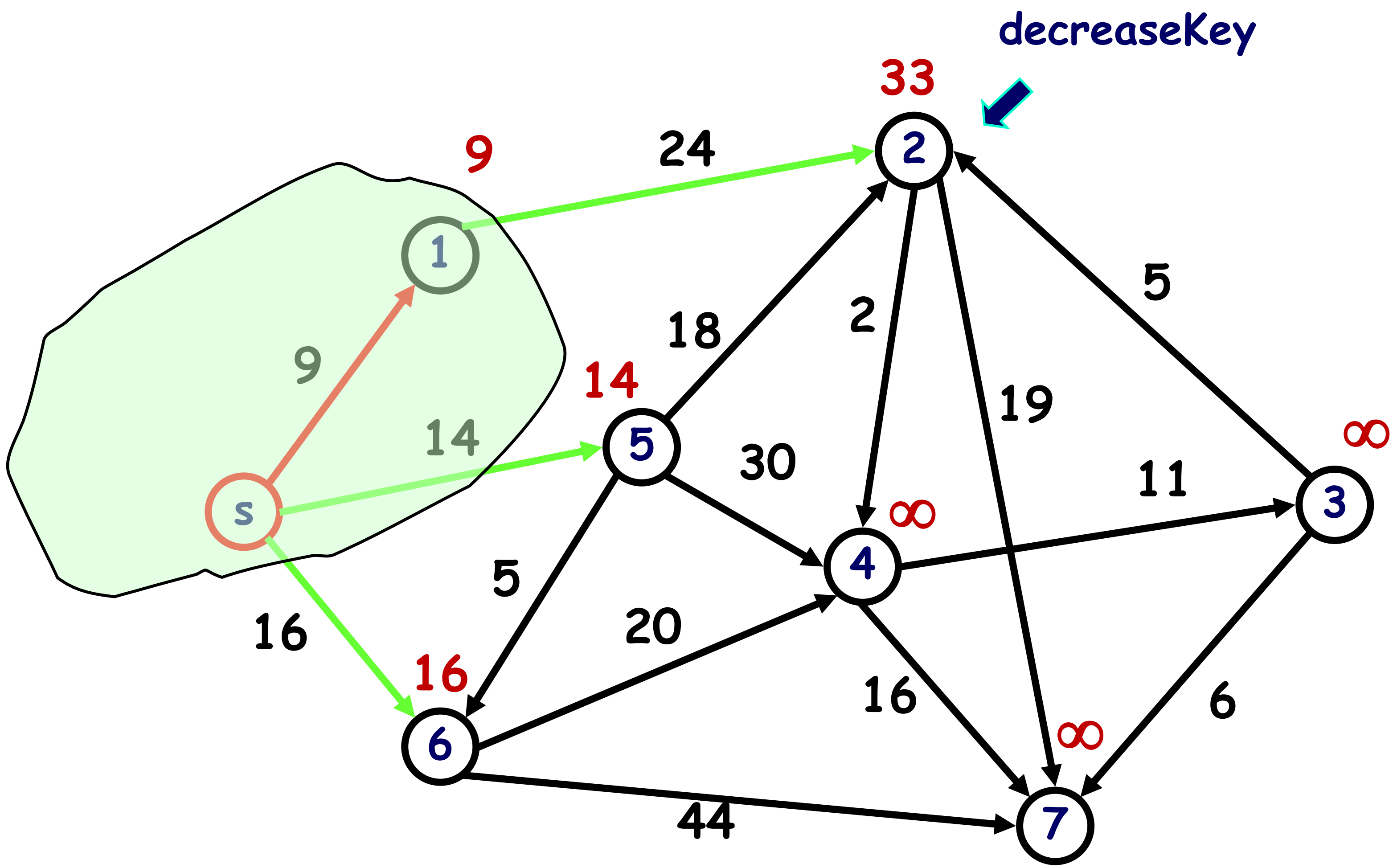
heap = {1, 2, 3, 4, 5, 6, 7}



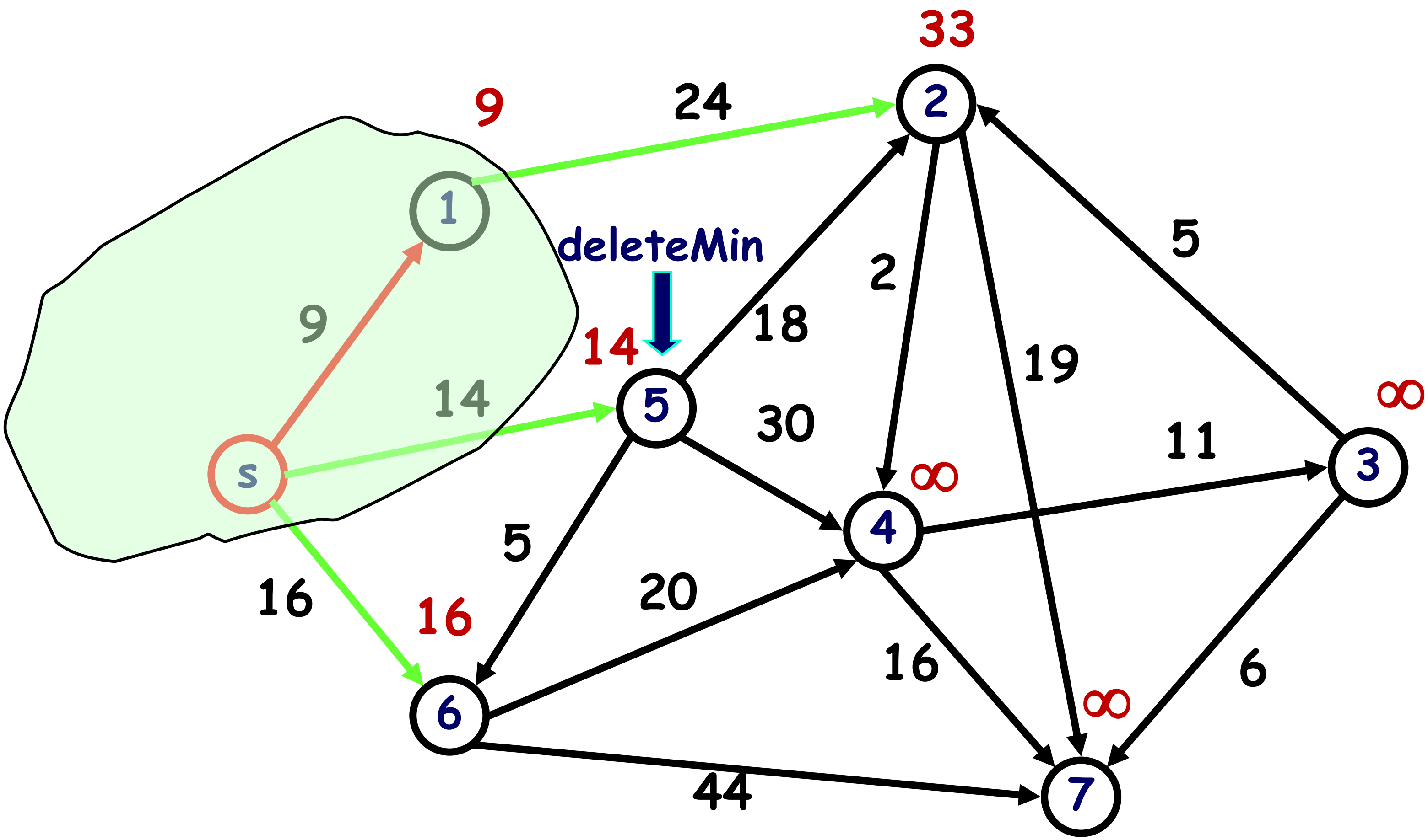
solution tree = { s, 1 }
heap = {2, 3, 4, 5, 6, 7}



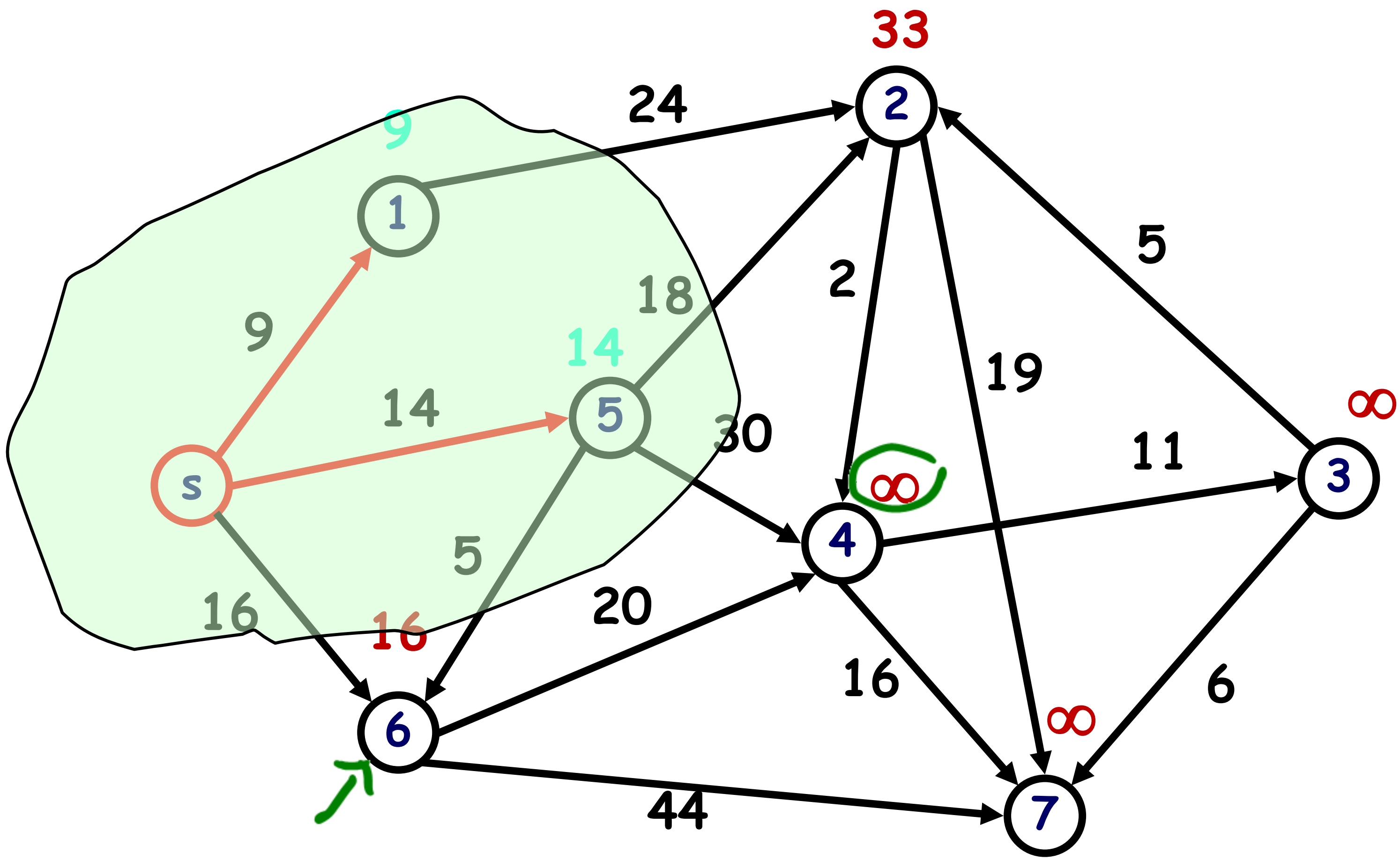
solution tree = { s, 1 }
heap = {2, 3, 4, 5, 6, 7}



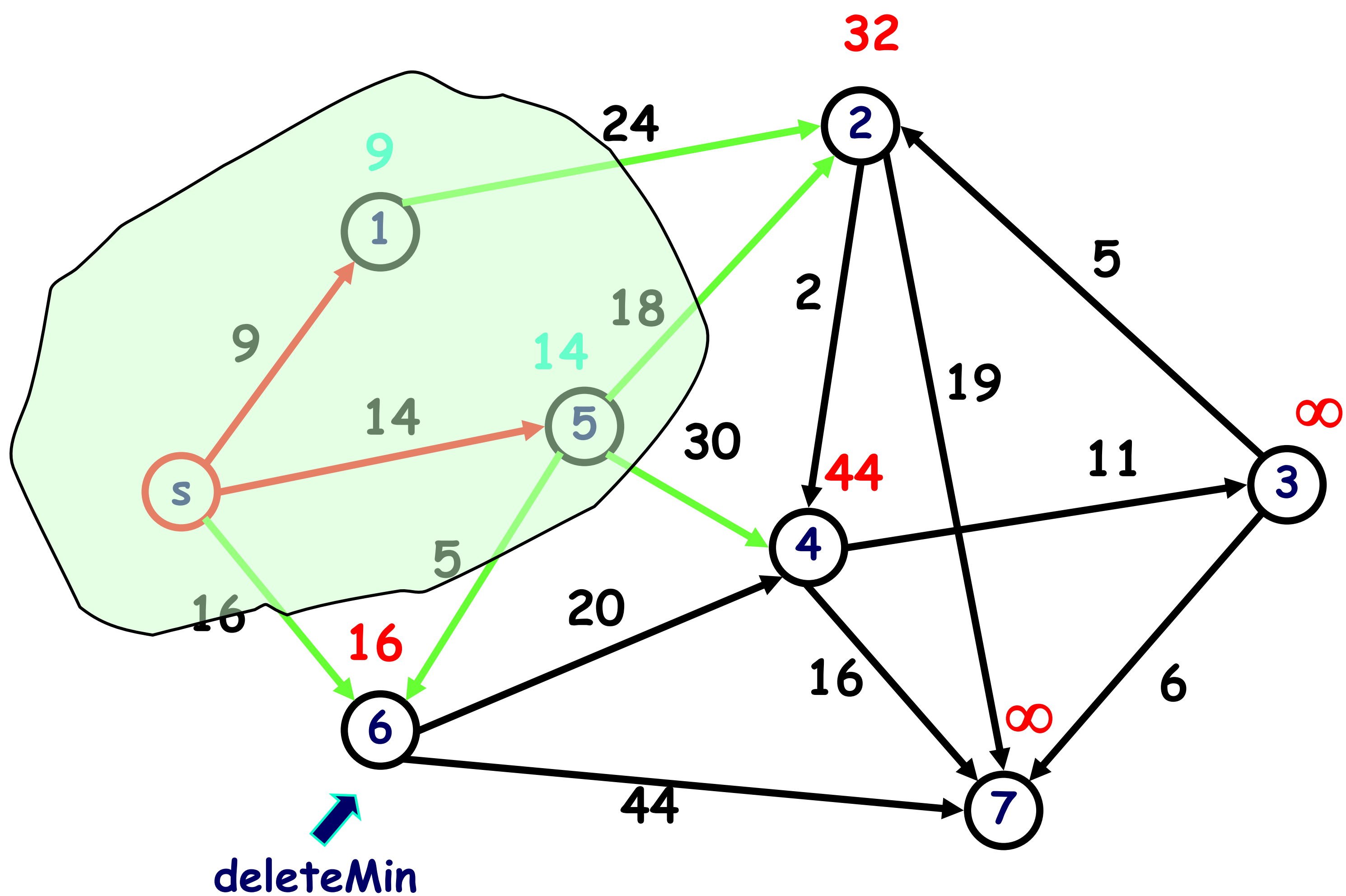
solution tree = { s, 1 }
heap = {2, 3, 4, 5, 6, 7}



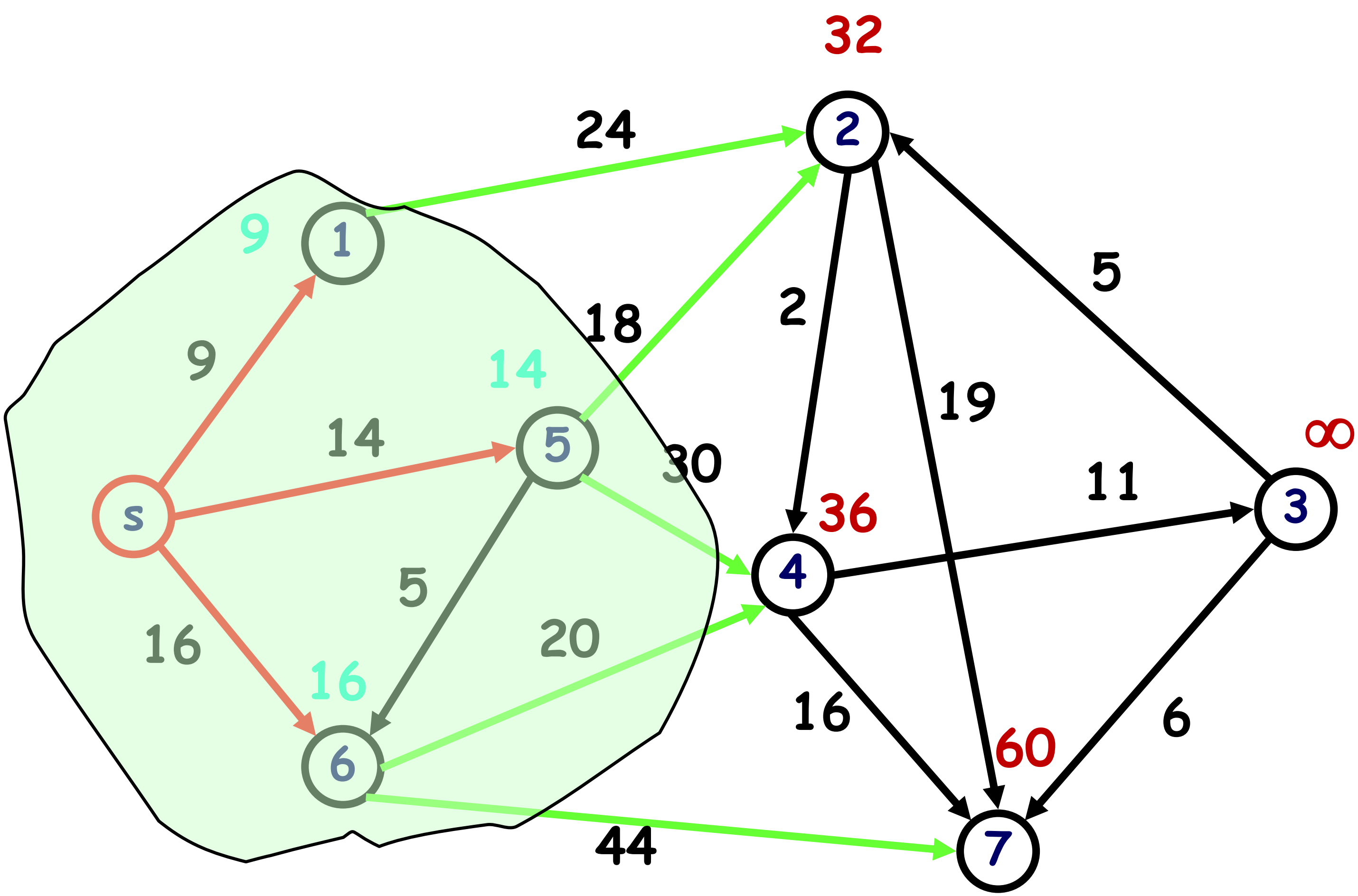
solution tree = { s, 1, 5 }
heap = {2, 3, 4, 6, 7}



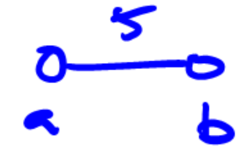
solution tree = { s, 1, 5 }
heap = {2, 3, 4, 6, 7}



solution tree = { s, 1, 5, 6 }
heap = {2, 3, 4, 7}



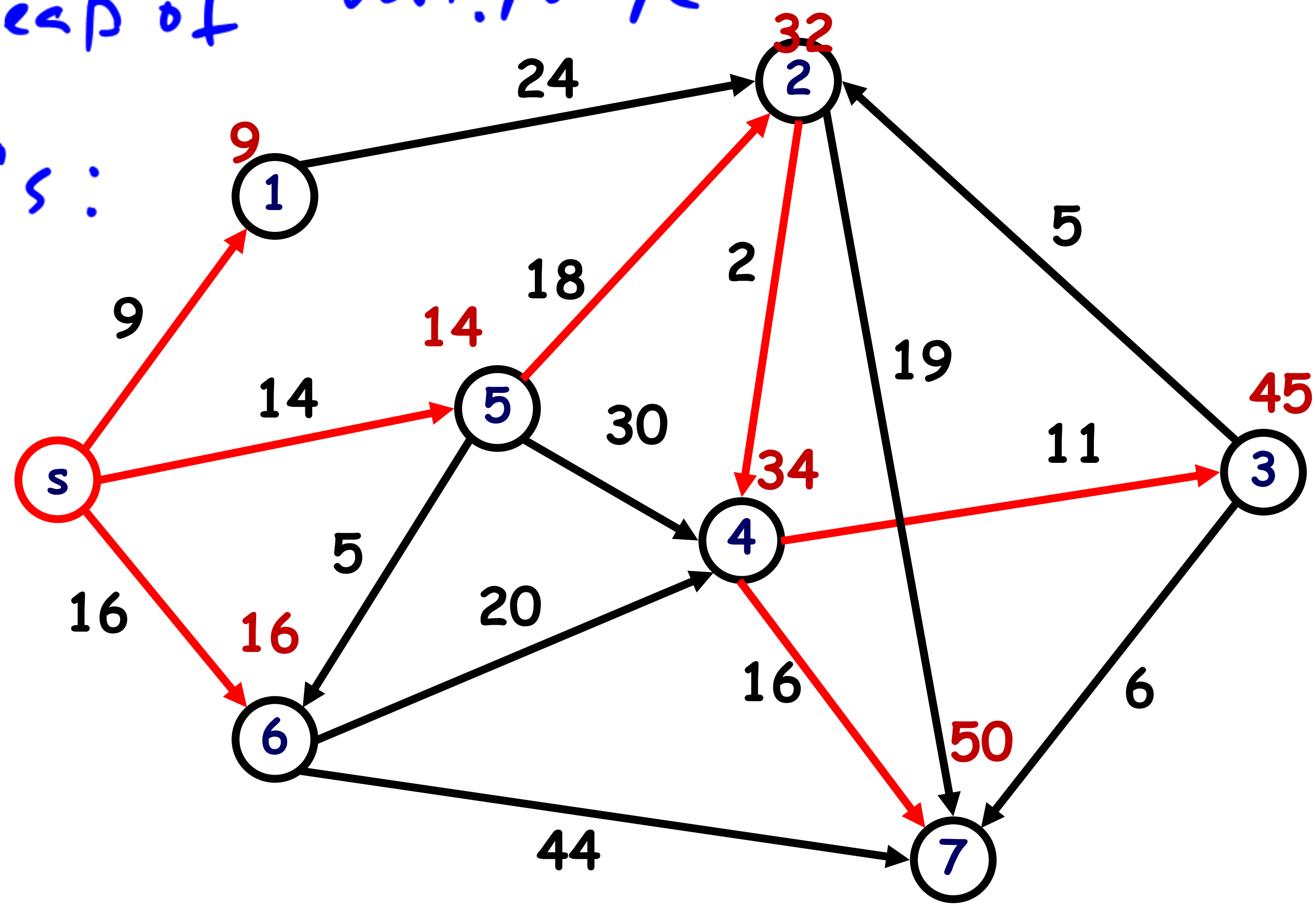
solution tree = { s, 1, 5, 6, 2, 4, 3, 7 }
heap = {}



spanningtree ≠ MST

Prim's: heap of vert./edge

Dijkstra's: heap of vert./dist.



Complexity

Let D(v) denote a length from the source s to vertex v.
We store distances D(v) in a binary heap.

INIT: $D(s) = 0$, $D(v) = \infty$ for $\forall v \neq s$

LOOP: deleteMin, $O(\log V)$ for each vertex

update distances in the heap, $O(\log V)$ for each edge

for all u in adj(v)

$\min(D(u), D(v) + \text{weight}(u,v))$

Runtime: $O(V \log V + E \log V)$, same as Prim's

Fibonacci: $O(V \log V + E \cdot 1)$ amortized

Discussion Problem

Assume that an unsorted array is used instead of a binary heap. What would the running time of the Dijkstra algorithm? $O(V \cdot V + E \cdot 1)$

dense graph, $E = O(V^2)$

$O(V^2)$

Fibonacci $O(V \log V + E)$, use a heap

dense graph

Proof of Correctness

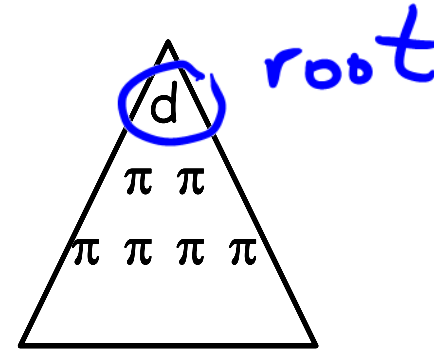
Lemma. For each node $u \in S$ (solution tree), $d(u)$ is the shortest s-u path.

Induction on $|S|$

Base case: $|S| = 1$, $d(s) = 0$

IH: $|S| = k$ vertices

IS: prove it for $(k+1)$ vertices



$\pi(v) \geq d(v)$

$$\pi(x) = \min\big(d(v),\ d(v) + w(v,x)\big)$$

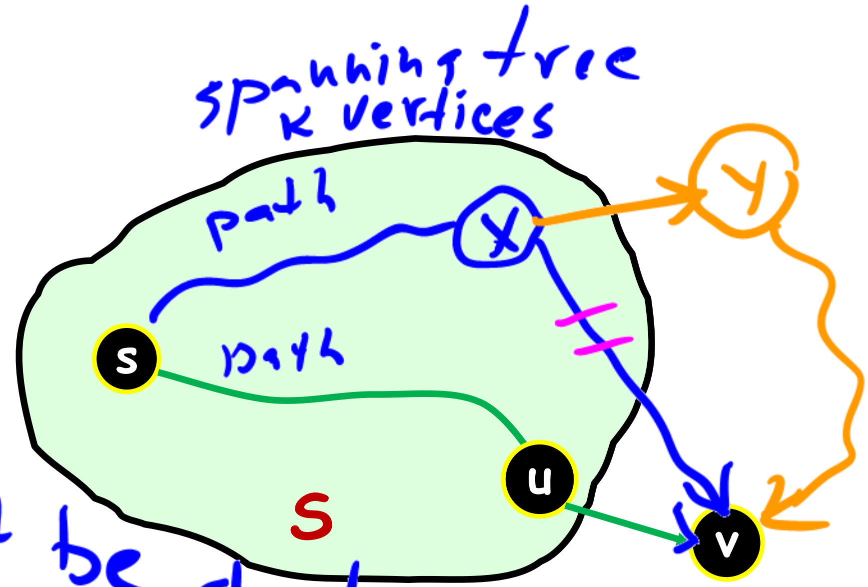
weights $\in \mathbb{R}^+$

Proof of Correctness

$d(u)$ is the shortest

$\pi(v) = d(u) + \text{edge}(u,v)$

Assume $\pi(v) \neq d(v)$



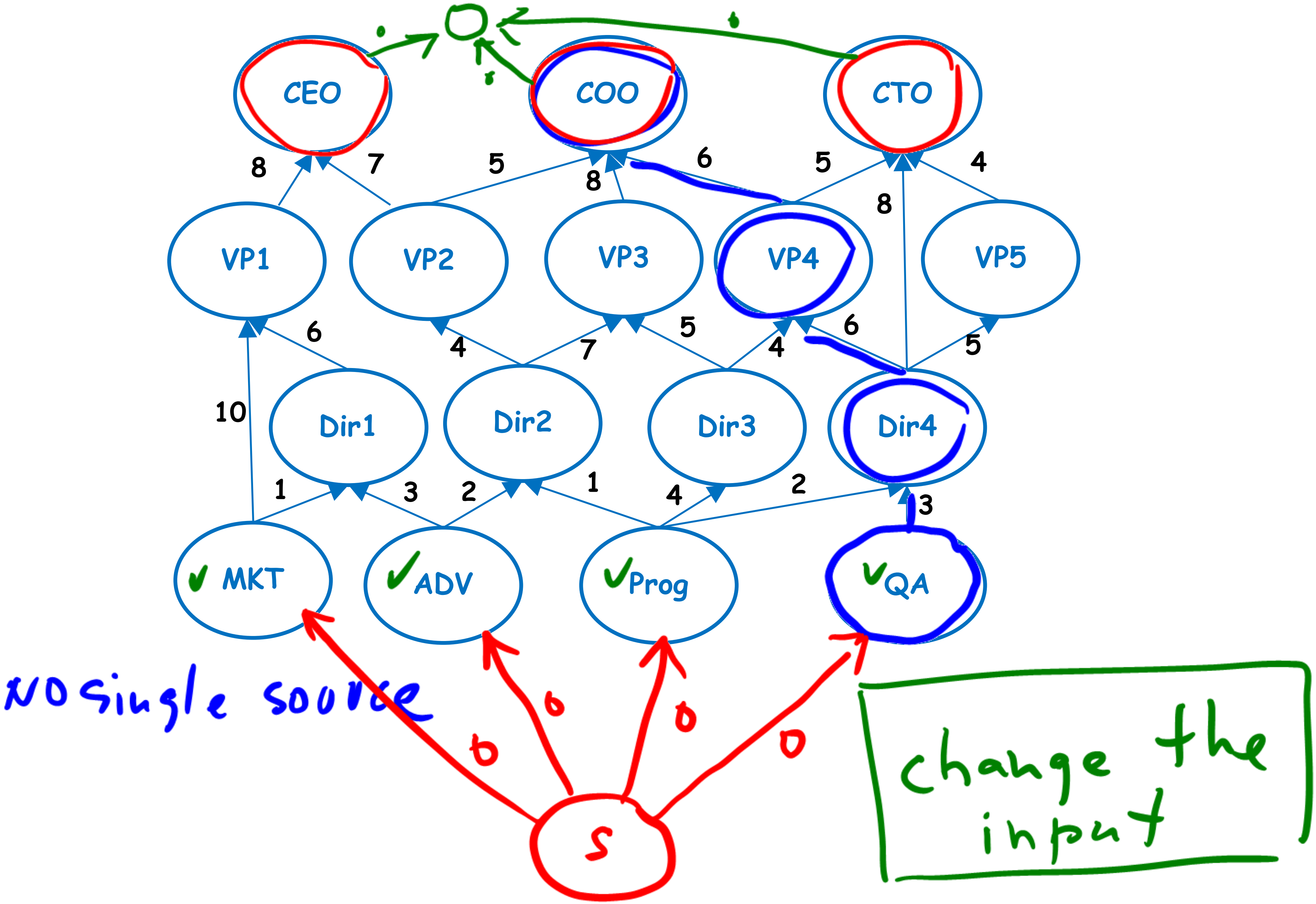
Case 1.
distanse through X cannot be shorter

Case 2. path s-x-y-v
is it the shortest?
our ALG would choose edge (x, Y)
but (u, v)

Discussion Problem 1

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 pre-requisite for u. Top positions are the ones which are not pre-requisites for any positions. Start positions are the ones which have no pre-requisites. The cost of an edge (v,u) is the effort required to go from one position v to position u. Salma 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 algorithm?

SSSP


CEO
COO
CTO
VP1
VP2
VP3
VP4
VP5
Dir1
Dir2
Dir3
Dir4
MKT
ADV
Prog
QA
S
no single source
change the input

Discussion Problem 2

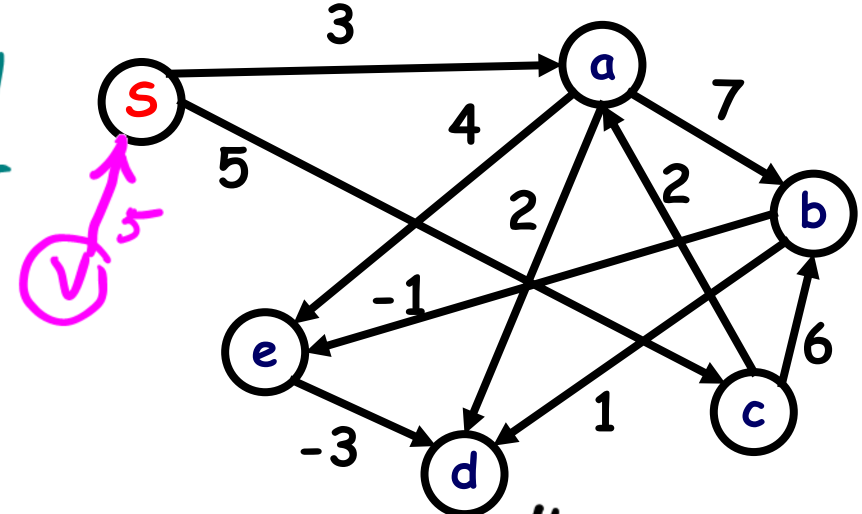
No Dijkstra

Design a linear time algorithm to find shortest distances in a DAG.

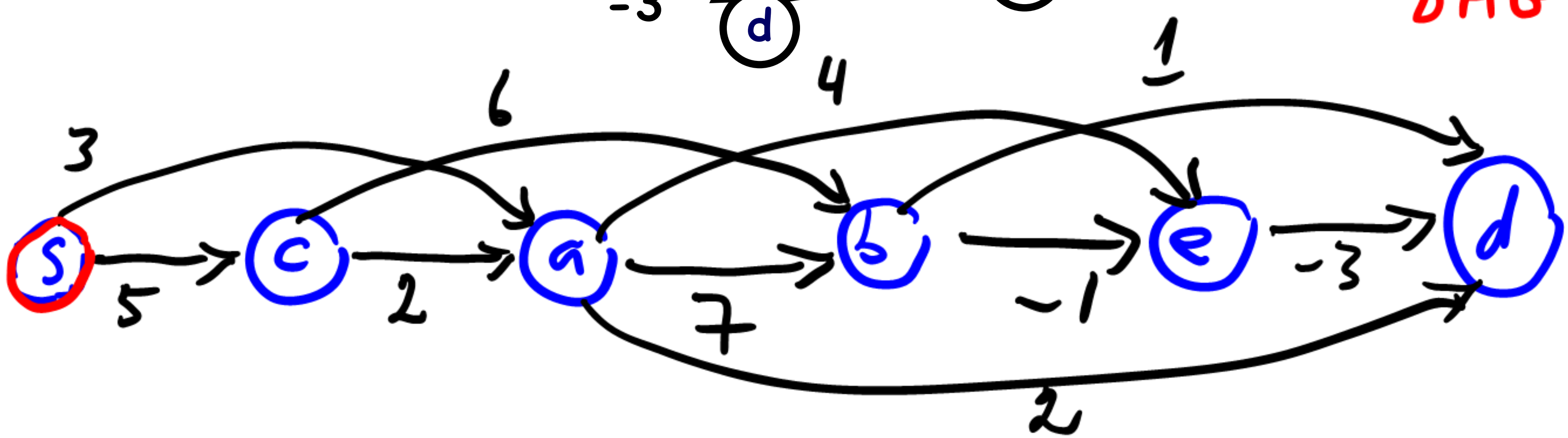
~~a) traversal, BFS~~

b) topological sort

How topological sort help us to solve the problem?



DAG

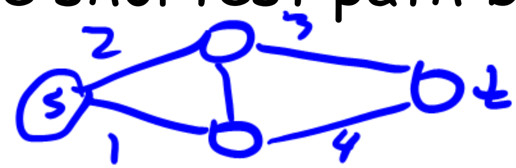


s	a	b	c	d	e	
0	3	–	5	–	–	s
	3	5+6	5	–	–	s, c
	3	10	5	5	7	s, c, a
	3	10	6	5	7	s, c, a, b
0	3	10	5	4	7	s, c, a, b, e

Runtime
$O(E)$

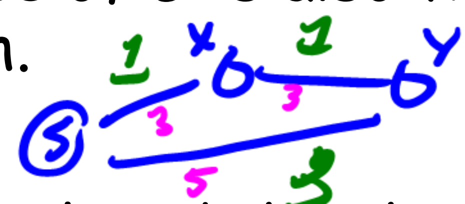
Review Questions

(T/F) If all edges in a connected undirected graph have distinct positive weights, the shortest path between any two vertices is unique. example



(T/F) Suppose we have calculated the shortest paths from a source to all other vertices. If we modify the original graph, *G*, such that weights of all edges are increased by 2 then the shortest path tree of *G* is also the shortest path tree of the modified graph.

+2



s-x-y , s-y

(T/F) Suppose we have calculated the shortest paths from a source to all other vertices. If we modify the original graph *G* such that weights of all edges are doubled, then the shortest path tree of *G* is also the shortest path tree of the modified graph.

$x+y+z < a+b$

$2(x+y+z) < 2(a+b)$

Discussion Problem 3

Why Dijkstra's greedy algorithm does not work on graphs with negative weights?

Run Dijkstra

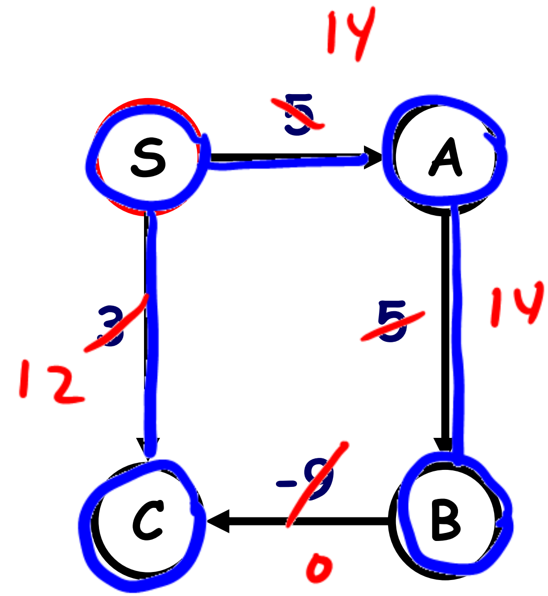
$d(S)=0$, $\boxed{d(C)=3}$

$d(A)=5$, $d(B)=10$

STOP

OPT: $d(C)=\underline{1}$

S-A-B-C



Run Dijkstra

$d(C)=12$

How to fix Dijkstra?

a) Re-weight

add 9

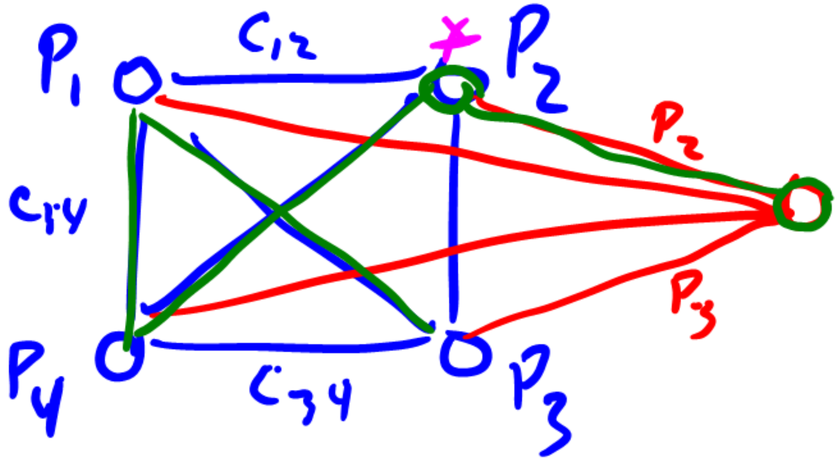
it does not work!

b) DP

Discussion Problem 4

In this problem you are to find the most efficient way to deliver power to a network of n cities. It costs p_i to open up a power plant at city i. It costs $c_{ij} \geq 0$ to put a cable between cities i and j. A city is said to have power if either it has a power plant, or it is connected by a series of cables to some other city with a power plant. Devise an efficient algorithm for finding the *minimum cost* to power all the cities.

MST



Run MST algo.

Why our solution will have at least one power plant?

The solution must have at least one red edge ...

Two ways to solve (in general)

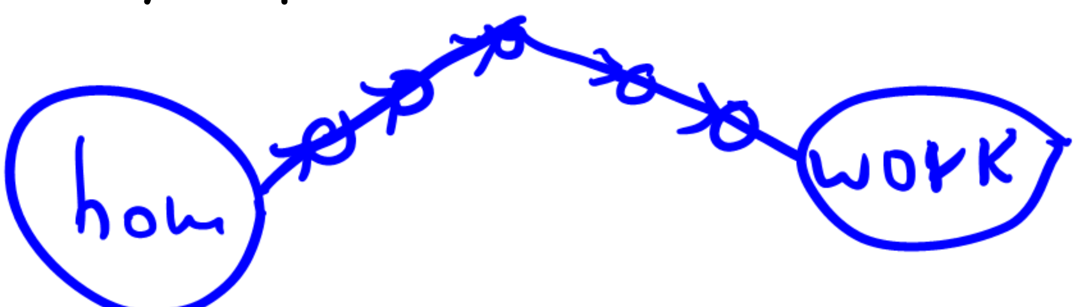
① change the input

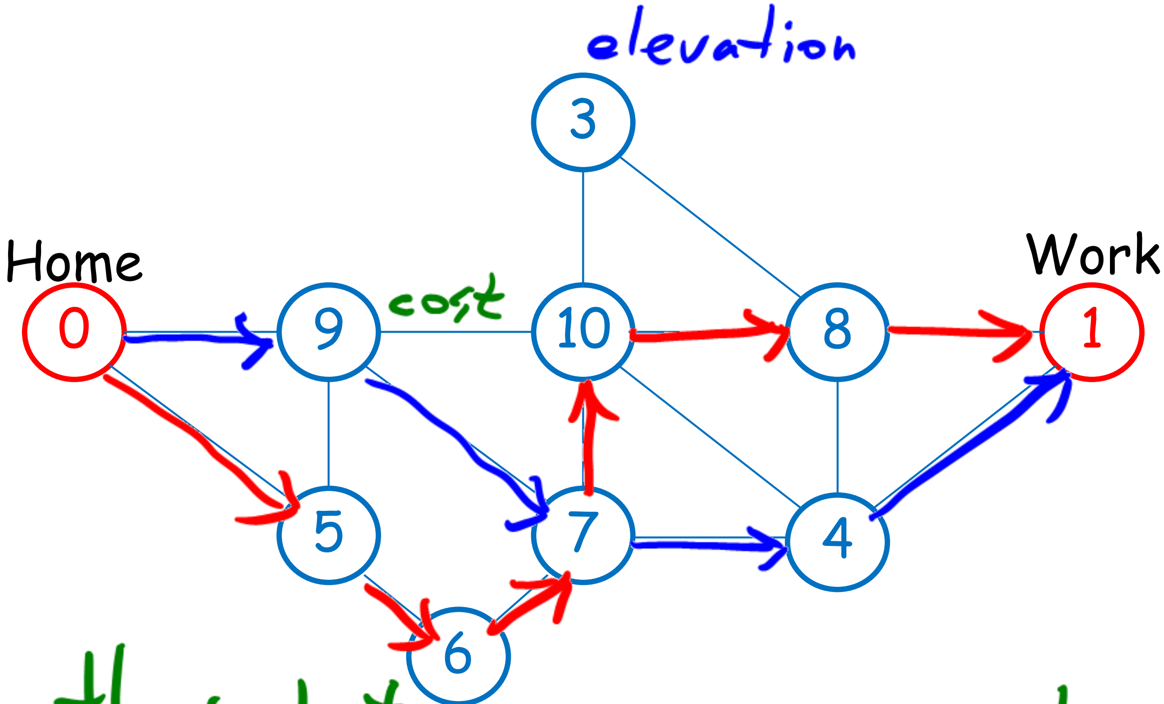
② change the algorithm

Discussion Problem 5

Hardy decides to start running to work in San Francisco to get in shape. He prefers a route to work that goes first entirely uphill and then entirely downhill. To guide his run, he prints out a detailed map of the roads between home and work. Each road segment has a positive length, and each intersection has a distinct elevation. Assuming that every road segment is either fully uphill or fully downhill, give an efficient algorithm to find the shortest path that meets Hardy's specifications.

Dijkstra

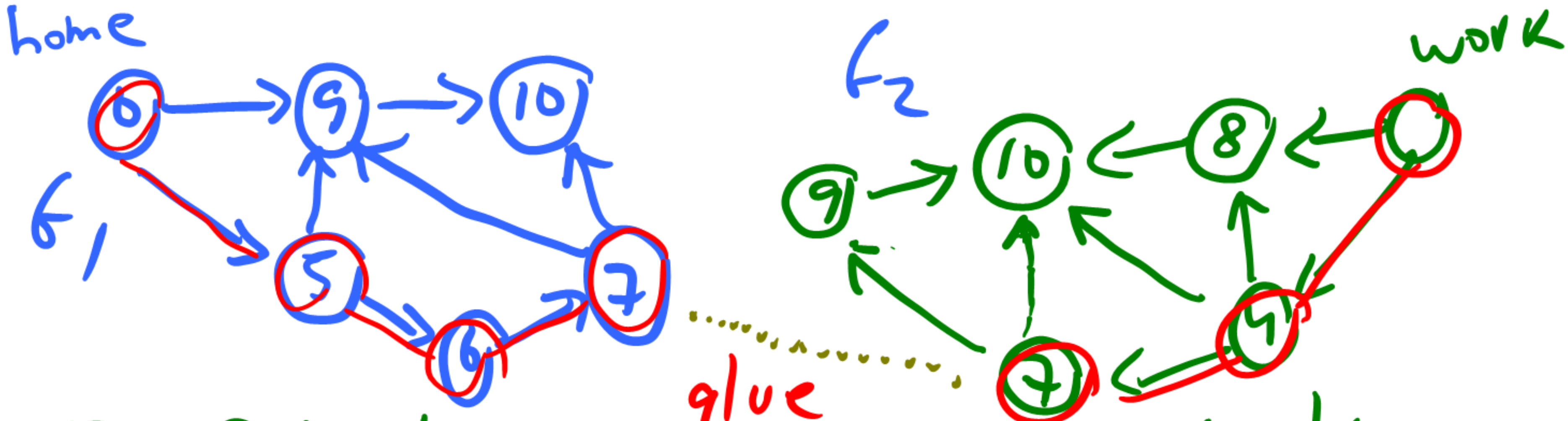




Change the input:

1. create uphill graph from home
2. create uphill graph from work

Uphill graph



Run Dijkstra from home

glue two paths

Run Dijkstra from work

Complexity: 1. construction G_1, G_2

2. run Dijkstra (twice)

3. find common vertices

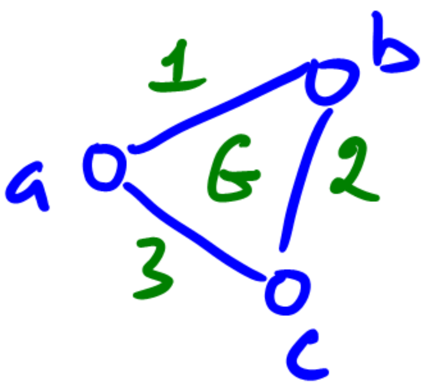
Discussion Problem 6

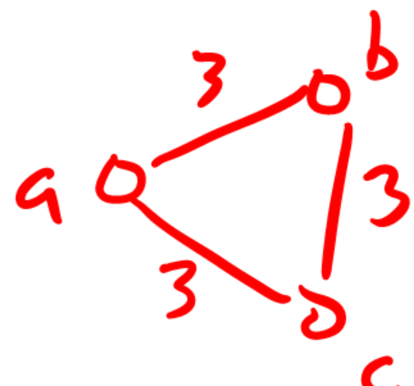
$V \to \infty$, $E \to \infty$

Given a graph G=(V,E) whose edge weights are integers in the range [0, W], where W is a relatively small integer number compare to the number of vertices, W = O(1). We could run Dijkstra's algorithm to find the shortest distances from the start vertex to all other vertices. Design a new linear time algorithm that will outperform ~~Dijkstra's~~ algorithm.

traversal
BFS

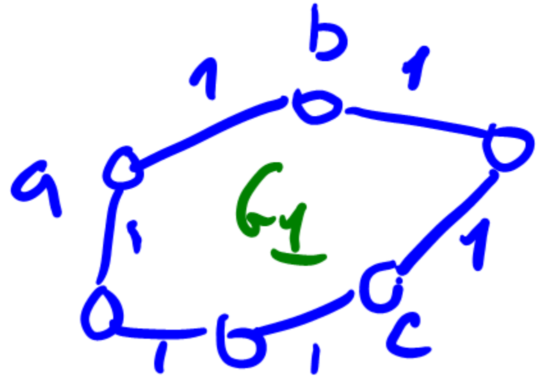
Change the input !!





W = 3, worst-case

Solution



Run BFS!

Runtime,

$G=(V,E)$ ← input

$G_1=(V_1,E_1)$

$BFS(G_1)=O(V_1+E_1)$

DIY verify

$V_1 = E + w\cdot V$, $E_1 = w\cdot E$ verify!

$BFS(G_1)=O(w\cdot V + w\cdot E) = O(V+E)$ if $w = O(1)$