02/01/17 LECTURE 7

Announcements:

→ Homework 2 is due today at 7:00 PM

→ Quiz 4 will be up today at 6
(until Friday at 10)

→ Exam 1 on Friday.

→ Attend in your official Section.

→ Cannot enter/take the exam
10mins from start.

NO EXCEPTIONS

→ Cannot leave until 2:00 PM

Pattern / Syllabus:

→ First 7 lectures.

→ Pattern: → 4/5 "problems" (like the ones in the homework)

→ Objective / "small" questions
(like the ones in the quizzes)

→ No notes / texts / etc; . .

Greedy Algorithms

"Greed is Good... Greed works..."

— Gordon Gecko

"Start with some solution and refine it using a greedy rule".

→ Often very easy/fast/simple to state or implement.

→ Can be "wrong" very easily.

→ "Greed does not always work!"

→ Analyzing them is trickier.

Today: Shortest paths.

Weighted Directed Graphs:

$G = (V, E)$

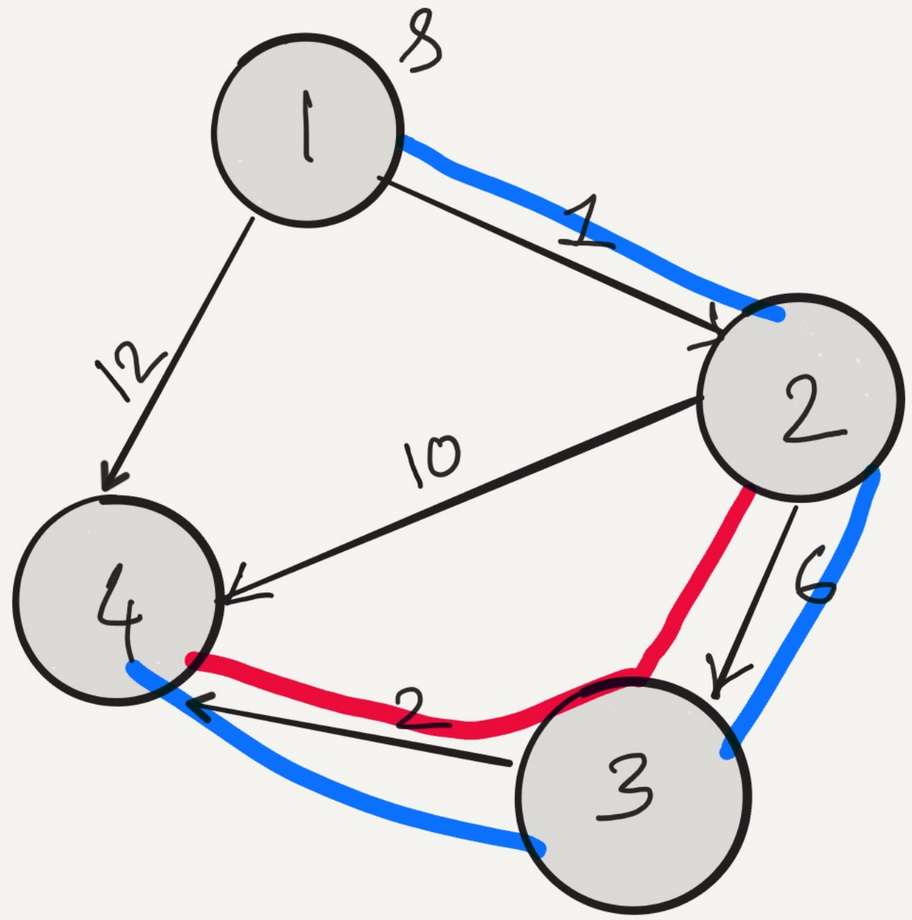
$V = \{v_1, v_2, \ldots, v_n\}$

$E = \{((v_1, v_2), 10), ((v_1, v_5), 1), ((v_1, v_4), 3), \ldots\}$

edges weights or lengths of edges.

If (u, v) is an edge: $l_{(u,v)}$ = length of the edge.

Ex:



$d(1,4) = 9$

$d(2,4) = 8$

→ Road networks with traffic information

→ Flight network with time to travel.

→ Routing networks with traffic information.

→ For us (for the time being), the lengths are positive.

Length of Paths: $P = (v_1, v_2, v_3, \ldots, v_k)$

$l(P)$ = Sum of the lengths of the edges on the path

Distance between vertices: Two vertices u, v from G

$d(u,v)$ = minimum length of a path connecting u to v

$d(u,v) = \infty$ if no path from u to v

Shortest Paths Problem

INPUT: $G = (V, E)$ Weighted directed graph with positive lengths

Vertex $s \in V$.

OUTPUT: Shortest paths from s to all other vertices.

(G is given in adjacency list representation.)

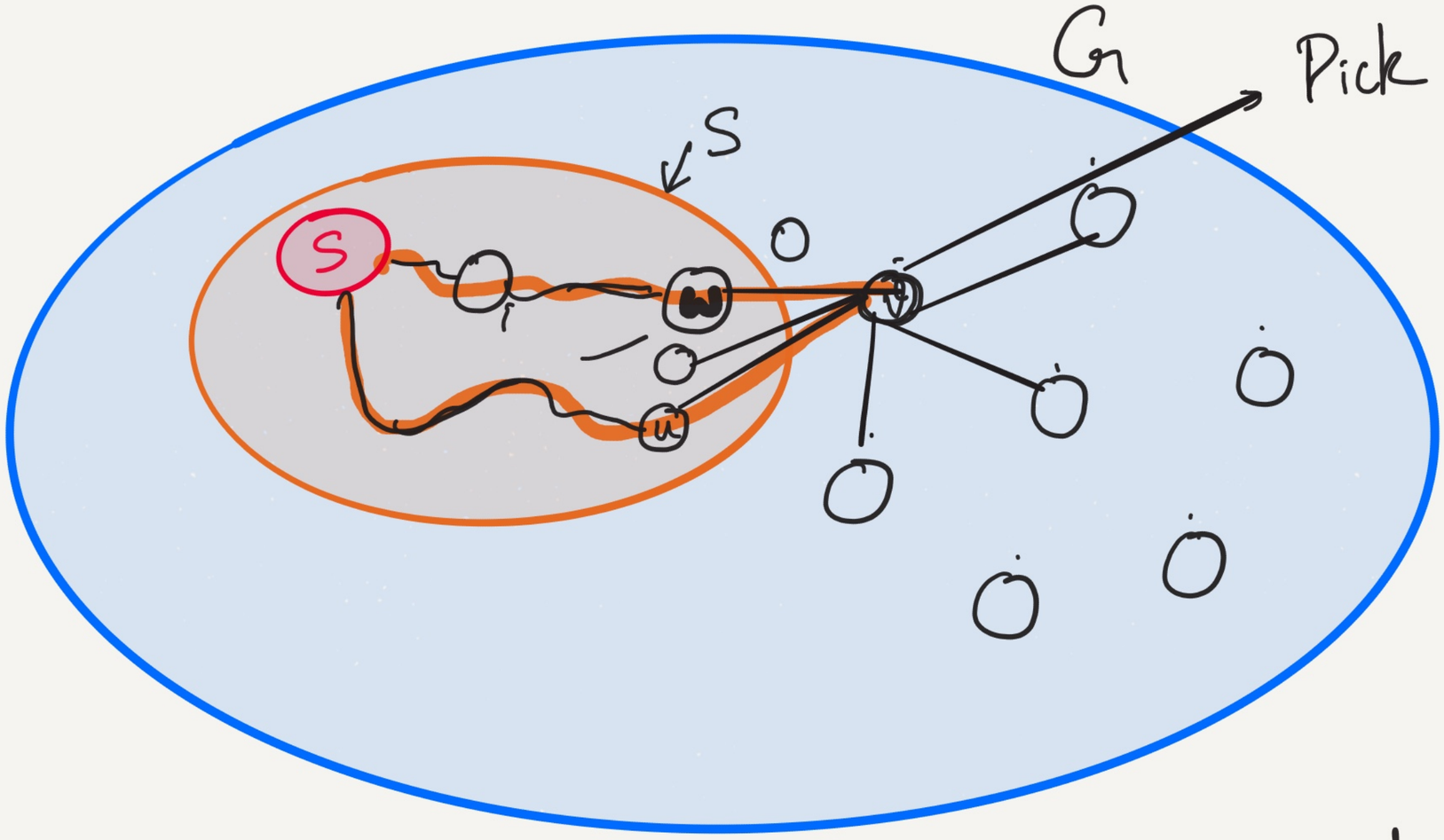
Greedy algorithm:

→ Gradually evolve the solution.

→ Suppose we have computed the shortest paths to a subset of vertices S.

→ Greedily update S.

→ Grow S by adding one new vertex $v \notin S$ that we also have the shortest path to.



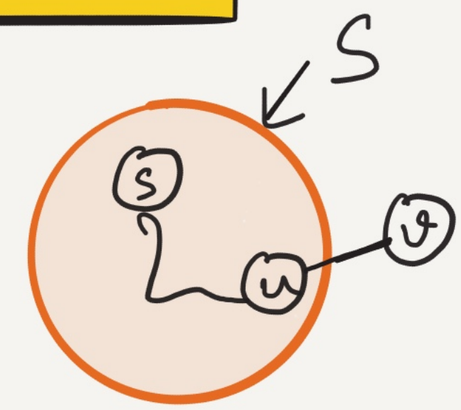
Pick the vertex that is closest to S

Every vertex in S, we already have the lengths of the shortest paths.

∀ vertex $v \notin S$, what is the length of the shortest path to v going from S?

Define

$$d'(v) = \min_{\substack{e=(u,v) \\ u \in S}} d(u) + l_{(u,v)}$$



Think of $d'(v)$ as measuring how close v is to the SET S.

→ Among all vertices $v \notin S$, pick the one with least $d'(v)$.

DIJKSTRA's ALGORITHM

Let S = Set of explored/discovered nodes. → $\boxed{d(s,u)}$

(For each vertex $u \in S$, we also store $d(u)$).

① $S = \{s\}$ and $d(s) = 0$. $d(v) = \infty$ $\forall$ $v \neq S$.

$\text{Parent}(v) = \phi$ $\forall$ $v \in S$

② While $S \neq V$

ⓐ For each vertex $v \notin S$

set $d'(v) = \min\{d(u) + l_{u,v} : u \in S \text{ and } (u,v) \text{ is an edge}\}$

ⓑ Find the vertex v with least $d'(v)$.

(i) ADD v to S.

(ii) $d(v) = d'(v)$.

Can we also find the shortest paths using the algorithm?
(and not just the lengths).

YES: Create a tree T and pointers sort of like what we did for BFS.

DIJKSTRA's ALGORITHM

Let S = Set of explored/discovered nodes. → $d(s,u)$

(For each vertex $u \in S$, we also store $d(u)$).

① $S = \{s\}$ and $d(s) = 0$. $d(v) = \infty$ $\forall$ $v \neq S$.

$\text{Parent}(v) = \phi$ $\forall$ $v \in S$. Tree $T = \phi$

② While $S \neq V$

ⓐ For each vertex $v \notin S$

set $d'(v) = \min\{d(u) + l_{u,v} : u \in S \text{ and } (u,v) \text{ is an edge}\}$

ⓑ Find the vertex v with least $d'(v)$.

(i) ADD v to S.

(ii) $d(v) = d'(v)$.

(iii) Find the vertex u that gets the min in the definition of $d'(v)$.

(iv) Set $\text{parent}(v) = u$.

(v) Add edge $(\text{parent}(v), v)$ to the tree.

Helpful for finding shortest paths

For any vertex u, we can reconstruct shortest path to u as follows:

Let P_u = path obtained by retracing parent links from u all the way to source s.

We will show that P_u's are shortest paths.

Analyzing the algorithm:

Strategy: "Greedy stays ahead".

Lemma1: At any stage, for any vertex $u \in S$, $d(u)$ is the length of the shortest path from s to u.

For a vertex $u \in S$, let P_u denote the path from s to u obtained by retracing the parent links.

Lemma2: $\forall u \in S$, P_u is a shortest path from s to u.

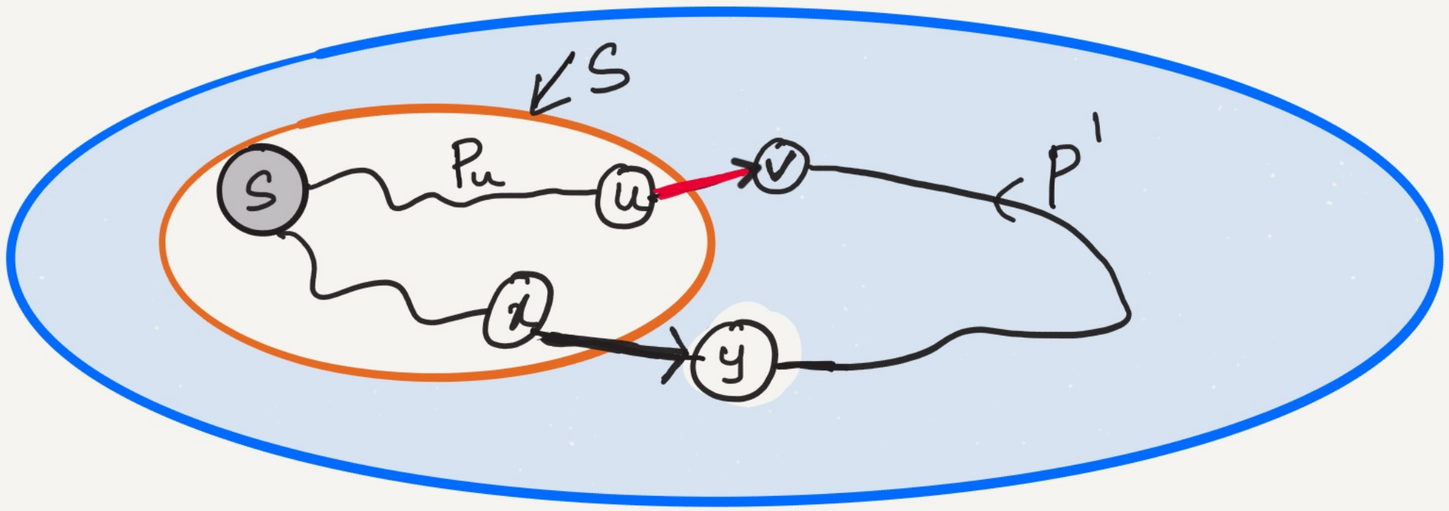
Proof is by induction on the size of the set S.

Base Case: $S = \{s\}$. $d(s) = 0$ (initialization) ✓

Induction step: Suppose $|S| = k$ and the statement is true.

We grew S by 1. Suppose we added a vertex v.

Goal: Show the statement holds for the new vertex v.



$|S| = k$.

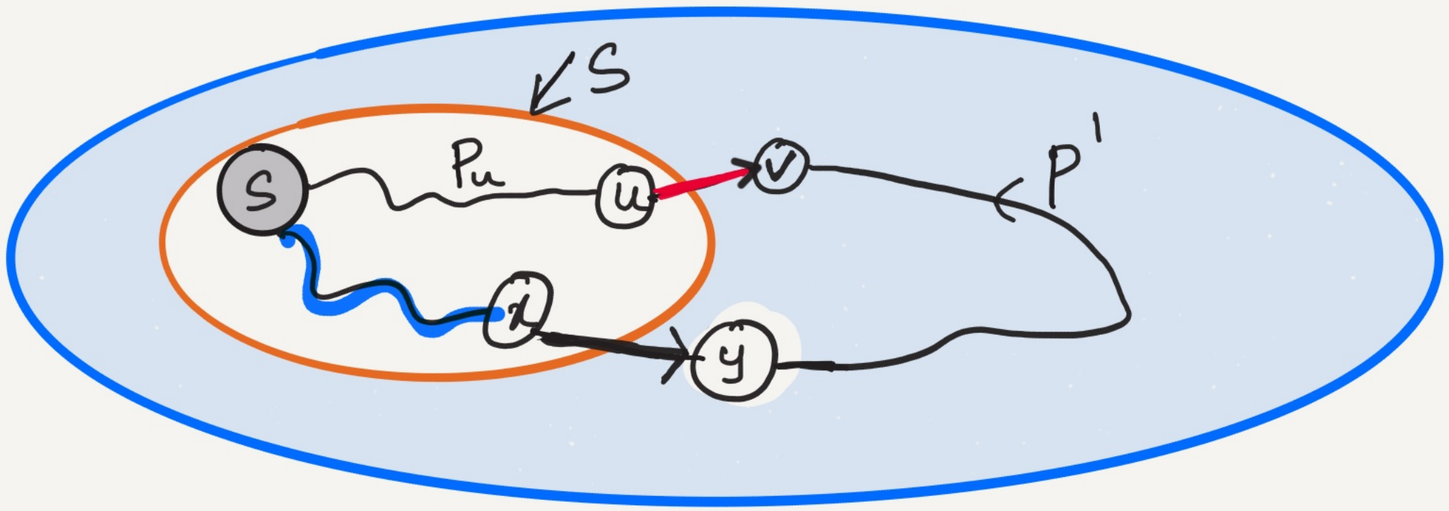
(u, v) is the edge used to add v.

Suppose the claim does not hold for v.

Suppose there is some other shorter path P'.

$$P_v = P_u + (u, v)$$

$$\ell(P') < \ell(P_v)$$



$|S| = k$.

(u,v) is the edge used to add v.

$$l(P') \geq l(P_x) + l_{(x,y)} \geq d(x) + l_{(x,y)}$$

By the induction hypothesis

$$l(P_x) \geq d(x)$$

$$l(P_v) = l(P_u) + l_{(u,v)} = d(u) + l_{(u,v)}$$

$$l(P') < l(P_v) \Rightarrow d(x) + l_{(x,y)} < d(u) + l_{(u,v)}.$$

$$d(x) + l_{(x,y)} < d(u) + l_{(u,v)} \quad \text{—} (\star)$$

is a contradiction !!

Why? The vertex v achieves the least value of $d'(u)$ among all vertices not in S.

But $(\star)$ says that y is closer to S:

$$d'(y) \leq d(x) + l_{(x,y)} < d'(v).$$

$\Rightarrow\Leftarrow$

$\Rightarrow$ P_v is a <u>shortest path</u> to v.

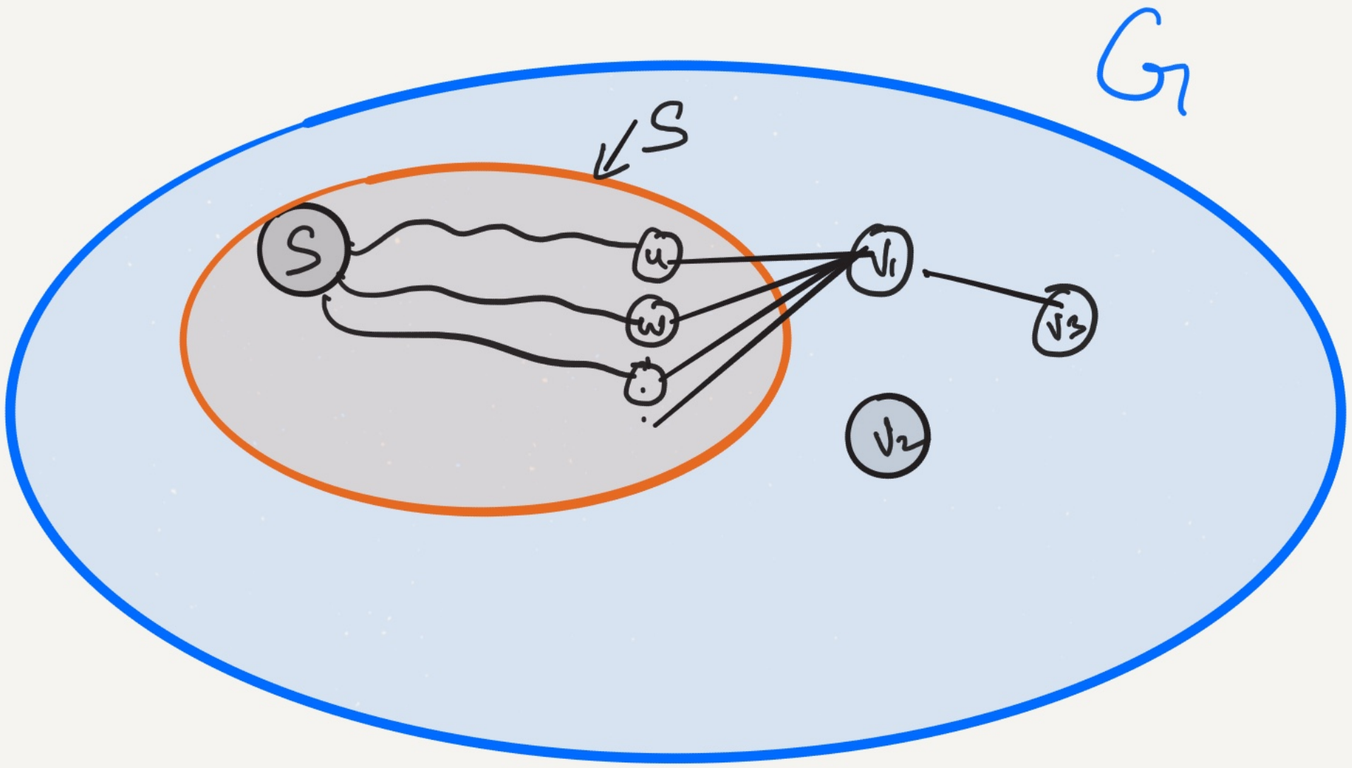
$\Rightarrow$ By induction the claim is true!!

(End of proof of Lemma 2.)

Theorem: Algorithm computes shortest paths from source s to all vertices

Proof: We terminate when $S = V$.

Apply Lemma 2 now.


G
S
S
u
w
v1
v2
v3