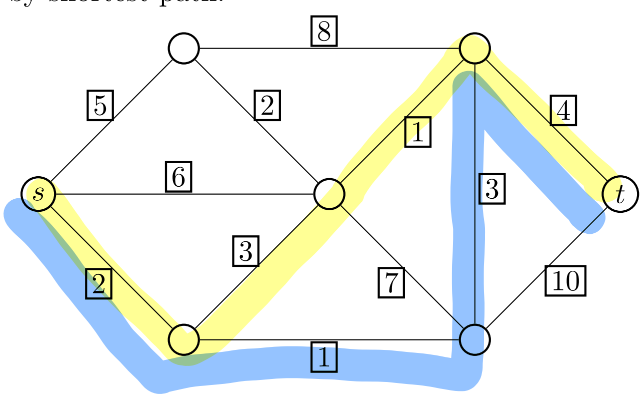
Tuesday, Oct. 11

1. Welcome!
2. Topic Submission by tonight
3. HW 5 up soon, due next Tuesday
4. Prof. Fox's DeWitt Wallace Lecture
   4:45 pm, Kagin Ballroom
5. Questions?
6. Shortest Path
7. Small Work → topic

Math 494: Discrete Optimization 10/11

Today in DiscOp we're continuing to talk about optimization problems on graphs. We spent all of last week carefully constructing linear programs for minimum spanning trees, creating an algorithm that'll solve the problem much faster, and then proving that algorithm yields an optimal solution to our linear program. We'll be doing something very similar today, just with a different graph problem: finding the "shortest path" between two vertices.

Example: Find the shortest path between s and t (use your intuition!), then formally state what we mean by shortest path.



→ length 10

Formalizing: the "length" of an $s-t$ path is the sum of the weights on the edges

Our first job today will be to construct a linear program for this question. We'll actually build two! The first will be more intuitive, but it will help to consider all graph as *directed*.

Definition: (directed graph) a *directed graph* $G = (V, E)$ is a graph on the vertices V where $E \subseteq V \times V$. Edge (u, v) is an ordered pair and points from u to v. uv

Question: How do we quickly make an undirected graph in to a directed graph?

make every undirected edge into 2 directed edges

Example: Write a linear program that solves the shortest path problem. A few things to consider along the way: (1) what are you making *decisions* about? That'll inform your variable and (2) what needs to happen at every vertex *along the way*? That'll inform your constraints.

decisions about edges

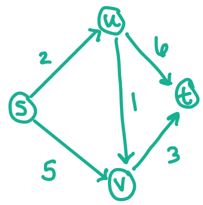
$$\begin{cases} \min & c^T x \\ \text{st} & \sum_w x_{sw} = 1 \\ & \sum_u x_{vu} - \sum_w x_{wv} = 0 \\ & \sum_u x_{ut} = 1 \end{cases} \rightarrow$$

one out of s

$$\begin{cases} \min & c^T x \\ \text{st} & \underbrace{\sum_v x_{vu}}_{\text{into } u} - \underbrace{\sum_w x_{uw}}_{\text{out of } u} = \begin{cases} -1 & u = s \\ 0 & u \neq s,t \quad * ! \\ 1 & u = t \end{cases} \\ & x \geq 0 \end{cases}$$

1 into t

Example: Write the linear program for a digraph G with directed edges $E = \{su, sv, uv, ut, vt\}$, where the costs are $c(su) = 2$, $c(sv) = 5$, $c(uv) = 1$, $c(ut) = 6$, $c(vt) = 3$.



$$\begin{cases} \min & 2x_{su} + 5x_{sv} + x_{uv} + 6x_{ut} + 3x_{vt} \\ \text{st} & -x_{su} - x_{sv} = -1 \\ & x_{su} - x_{uv} - x_{ut} = 0 \\ & x_{sv} + x_{uv} - x_{vt} = 0 \\ & x_{ut} + x_{vt} = 1 \\ & x \geq 0 \end{cases}$$

Let's dualize our general program and try to interpret what it does. It's actually going to inform our algorithm!

Primal $\min c^T x$ $\quad x_i \geq 0$

$$\begin{array}{c|c} & \text{edges } ab \\ \text{vert. } v,\ s,\ t & \left[\ A\ \right] \end{array} \left[\, x \,\right] = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

$$A_{v,ab} = \begin{cases} 0 & \text{if } v \notin \{a,b\} \\ -1 & \text{if } v = a \\ 1 & \text{if } v = b \end{cases}$$

Dual $\max y_t - y_s$ $\quad\leftarrow$ we can set $y_s = 0$

$$\begin{array}{c|c} & \text{a vert } b \\ \text{edges } ab & \left[\ -1 \quad A^T \quad 1\ \right] \end{array} \left[\, y \,\right] \leq \begin{bmatrix} \text{cost} \\ \vdots \\ \vdots \end{bmatrix}$$

y free.

$\underline{y_b - y_a \leq c_{ab}}$

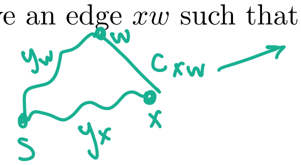
Feasible solutions to this dual program are called *feasible potentials*.

Claim: Let y be a feasible potential and let P be a path from our initial node s to any vertex v. Then $\underline{c(P) \geq y_v}$. Suppose $P = \{e_1, e_2, \ldots e_k\}$

$$c(P) = \sum_{i=1}^{k} c_{e_i} \geq \sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) = y_{v_k} - y_{v_0} = y_v - 0$$

So now that we've got a linear program, we notice once again that it's big. A more specialized algorithm also once again saves the day. First, let's develop two big ideas.

Big idea 1: Suppose we have a path from s to any v with length y_v. Moreover, suppose we have an edge xw such that $y_x + c_{xw} < y_w$. How do we make a shorter path to w?

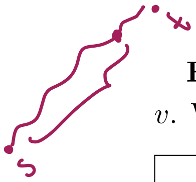


we can travel via x to w and get a shorter path!

Big idea 2: Suppose y_v is the length of the shortest directed path from s to v. What must be true about all edges vw and their relationship to y_w and y_v?

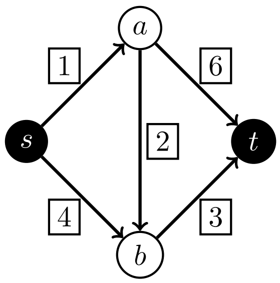
gotta be a feas. potential

$$y_w \leq y_x + c_{xw} \quad \forall\ xw \in E$$



Ford's Algorithm: Initialize a *infeasible* potential of $y_s = 0$ and $y_v = \infty$ for all other vertices v. While y is an infeasible potential, find a problematic edge and correct it.

Example: Run Ford's algorithm on the small graph below. We will also keep track of the *predecessor* of a vertex, essentially which edge helped us update the potential.



			sa		sb		at		ab		bt	
	y	p	y	p	y	p	y	p	y	p	y	p
s	0	0	0	0	0	0	0	0	0	0	0	0
a	∞	-1	1	s	1	s	1	s	1	s	1	s
b	∞	-1	∞	-1	4	s	4	s	3	a	3	a
t	∞	-1	∞	-1	∞	-1	7	a	7	a	6	b

Follow Up: Once your algorithm has terminated, take a look at the vector y. What do you notice about its entries and shortest paths from s?

y_v = length of the shortest path from s to v, p tracks how to get there.

Alright, time to prove this is going to work. We'll do this in two steps.

Claim. *Suppose G has no negative directed cycles. At any stage of the algorithm, we know the following:*

(1) *If $y_v \neq \infty$, then it is the length of a simple directed path from s to v.*
(2) *If $p_v \neq -1$, then p defines a simple directed path from s to v of length at most y_v.*

Pf (sketch):

(1). by construction, y_v will always be a length of a directed path from s to v. S/p its not simple (we used an edge twice)

where we updated v_i

this means there is a cycle in P: $v_0, v_1, \dots v_k = v_0$ with iteration numbers $q_0, q_1, \dots q_k$ such that

notation y_v^j = val of y_v @ step j.

$$\left\{ y_{v_{i-1}}^{q_i - 1} + c_{v_{i-1}v_i} = y_{v_i}^{q_i} \right.$$

if we update v_k, we decrease it from y_{v_0}!

$$\sum_{i=1}^{k} c_{v_{i-1}v_i} = \sum_{i=1}^{k} y_{v_i}^{q_i} - y_{v_{i-1}}^{q_i - 1} = y_{v_k}^{q_k} - y_{v_0}^{q_0} < 0 \quad ↯.$$

Claim. *Suppose G has no negative directed cycles. Then Ford's Algorithm terminates in a finite number of steps and p defines a least cost directed path from r to v of cost y_v.*

Pf: only finitely many paths in G b/t any 2 vertices, so b/c y, p track paths, only finitely many options for y, p. This terminates. previous claim.

by our previous claim, $y_v \geq C(P)$, but from earlier, any feasible potential has $y_v \leq C(P)$

So at this stage, we know the algorithm terminates at the best (max value at y_t) feasible potential. By duality, we can state the following theorem.

Theorem: Let G be a digraph $s, t \in V$, and $c \in \mathbb{R}^E$. If there exists a least cost directed path from s to v for all $v\ inV$,

$$\min\{c(P) : P \text{ a directed } s-t \text{ path}\} = \max\{y_t : y \text{ a feasible potential}\}$$

We'll close out our discussion of shortest path problems with two modifications of Ford's Algorithm. The first will handle the case of a negative cost cycle (which we had to forbid).

Algorithm 1 Moore-Bellman-Ford Algorithm

```
Initialize y(s) = 0, y(v) = ∞ for all v ≠ s
Initialize p(s) = 0, p(v) = −1 for all v ≠ s
repeat n − 1 times        } *
    for vw ∈ E do         }
        if y(w) > y(v) + c_vw then
            y(w) := y(v) + c_vw
            p(w) = v
        end if
    end for
until
for vw ∈ E do                      } **
    if y(w) > y(v) + c_vw then     }
        Found a negative cycle, return it.
    end if
end for
```

Space for notes...

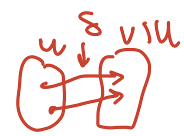
* every shortest path b/t 2 vert. is length $\leq n-1$

So updating all edges that many times will consider all paths

** Still not feas? → neg. cycle

Our second extension will consider Dijkstra's Algorithm, which processes the vertices instead of the edges. To do this, we'll consider another version of the shortest path LP.

Example: The shortest path program has a few formulations. Check out the one below and explain why it does the same thing. What might the dual of this program look like?

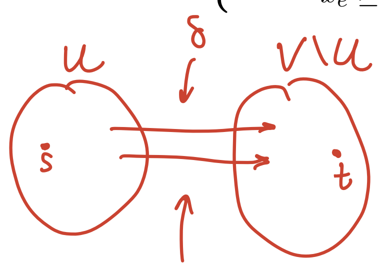


$$\begin{cases} \min & c^T x \\ \text{s. t.} & \sum_{e \in \delta(U)} x_e \geq 1 \quad \forall U \subseteq V,\ s \in U, t \notin U \\ & x_e \geq 0, x_e \in \mathbb{Z} \end{cases}$$

$\delta(U) = \{uv : u \in U, v \notin U\}$

$\boxed{P}$ edges

cuts



pick at least one of these!

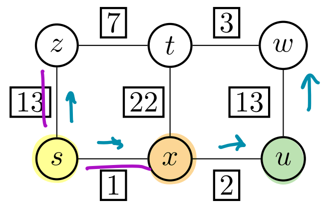
if our path goes from s to t, it needs to cross $\delta(U)$ $\forall$ U

Algorithm 2 Dijkstra's Algorithm

Initialize $y(s) = 0,\ y(v) = \infty$ for all $v \neq s$
Initialize $p(s) = 0,\ p(v) = -1$ for all $v \neq s$
Initialize $R = \emptyset$
while $R \neq V$ **do**
    Find a vertex $v \in V \setminus R$ such that $y(v) = \min_{w \in V \setminus R} y(w)$.
    Set $R = R \cup \{v\}$
    for all $w \in V \setminus R$ such that $(v, w) \in E$ **do**
        if $y(w) > y(v) + c_{vw}$ **then**
            Set $y(w) := y(v) + c_{vw}$
            Set $p(w) := v$
        end if
    end for
end while

Space for notes...

Example: Run Dijkstra's Algorithm on the graph below, tracking your steps as you go!



R	$y(s)$	$p(s)$	$y(z)$	$p(z)$	$y(x)$	$p(x)$	$y(u)$	$p(u)$	$y(w)$	$p(w)$	$y(t)$	$p(t)$
$\emptyset$	0	0	∞	-1	∞	-1	∞	-1	∞	-1	∞	-1
$\{s\}$	0	0	$\boxed{13}$	s	$\boxed{1}$	s	∞	-1	∞	-1	∞	-1
$\{s,x\}$	0	0	$\boxed{13}$	s	1	s	$\boxed{3}$	x	∞	-1	$\boxed{23}$	x
$\{s,x,u\}$	0	0	$\boxed{13}$	s	1	s	3	x	$\boxed{16}$	u	$\boxed{23}$	x
$\{s,x,u,z\}$	0	0	13	s	1	s	3	x	$\boxed{16}$	u	20	t
$\{s,x,u,z,w\}$	0	0	13	s	1	s	3	x	16	u	$\boxed{19}$	w
$\{s,x,u,z,w,t\}$	0	0	13	s	1	s	3	x	16	u	19	w

Claim. *Dijkstra's algorithm works correctly.*

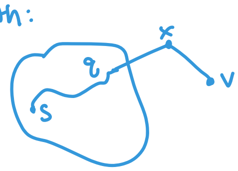
Pf: we consider 2 claims that are true @ any stage.

$\boxed{1}$ for each $v \in V \setminus \{s\}$ w/ $y(v) < \infty$, we know $p(v) \in V$, $y(v) = y(p(v)) + c_{p(v),v}$ and the seq. $v, p(v), p(p(v)) \ldots$ contains s.

↳ pf: if $y(v) < \infty$, $v \in R$ or we just reached it. Thus $p(v)$ is updated, so $p(v) \in R$ and the rest follows by the construction.

$\boxed{2}$ $\forall \; v \in R$, $y(v) = \text{dist}_{(G,c)}(s,v)$

S/p $v \in V \setminus \{s\}$ is added to R and assume (for $\lightning$) $\exists$ s-v path of length $< y(v)$. Let x be the first vertex on the path not in R, and let q be its predecessor on the path:



Then $y(x) \leq y(q) + c_x$