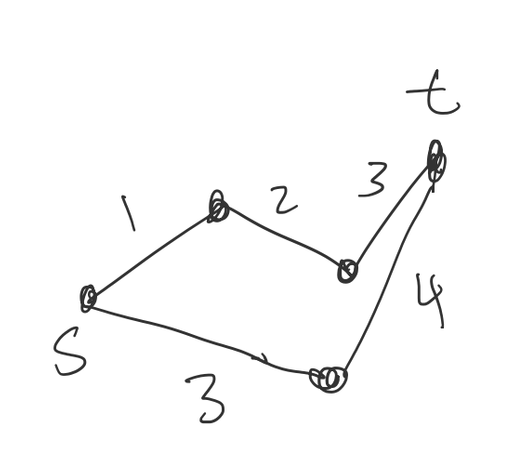
10: Bellman-Ford (Shortest Path)

Thursday, April 15, 2021 4:52 PM

Shortest Path Problem

Input: Graph $G=(V,E)$, $w: E \to \mathbb{R}$, $s,t \in V$, $|V|=n$

Output: Shortest path from s to t in G
↓
sum of edge weights on path



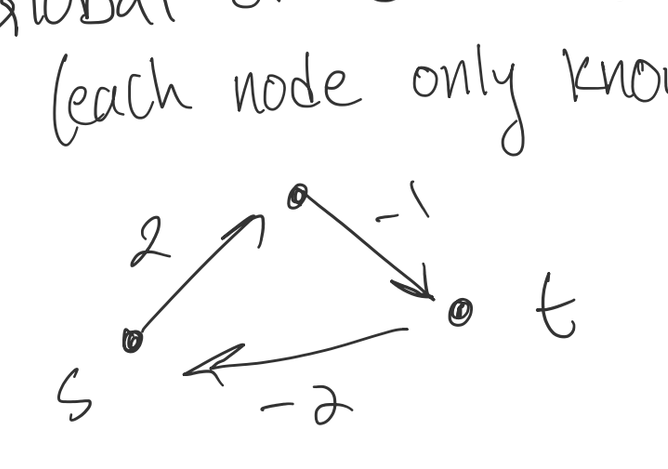
3 approaches

- Dynamic Programming
- Greedy
- Brute Force

} Which approach is used depends on type of graph

Bellman-Ford: used for graphs with
- directed or undirected edges
- positive + negative weights
- no negative cycles
- Global or distributed description of G (each node only knows neighbors)

Negative Cycle:



With modification: can create alg to detect neg. cycles

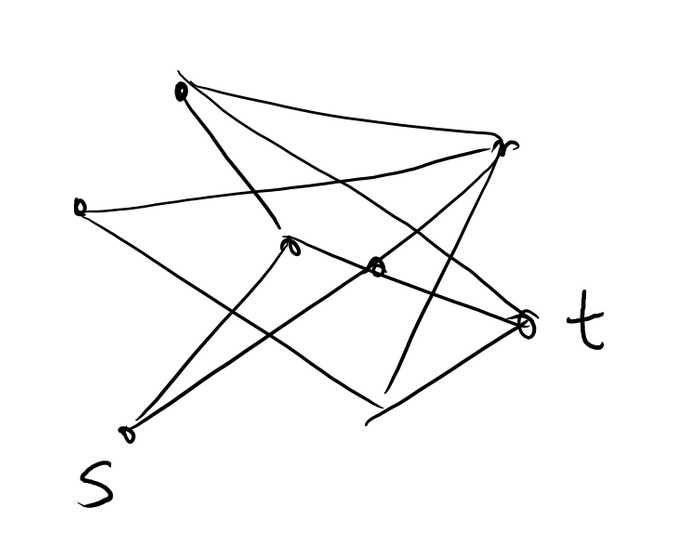
Applications

- Data routing in distributed networks
- Bartering: Apple –1→ orange, +1.5 → banana
  - Can get orange and $1 for apple
  - Can get banana for orange and $1.50
- arbitrage = find inefficiency in currency trading market to make $. (Neg cycle detection version)

Harm/Benefit?

Designing D.P.

Think about options for final choice in our solution/strategy.
Create a recurrence in terms of smaller subproblem + final choice.



Final Choice?
Last edge on path before t.

Let P_u = Shortest path from s to u

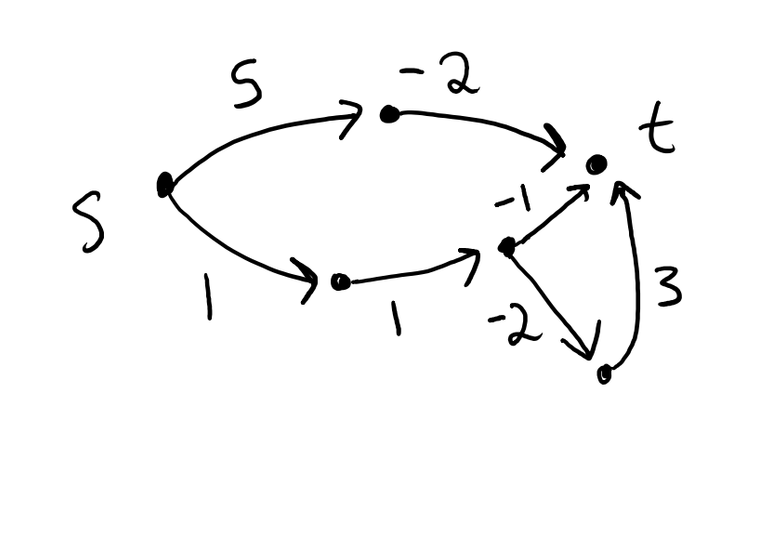
$$P_t = \begin{cases} P_u + (u,t) & \text{if } u \text{ was final vertex before } t \\ P_v + (v,t) & \text{if } v \\ \vdots & \\ (s,t) & \text{if } s \end{cases}$$

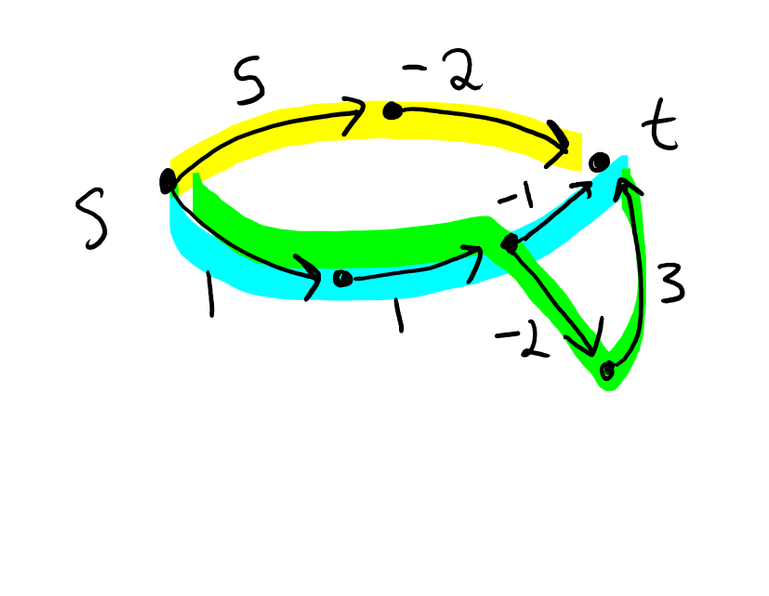
Difficulty: P_u is not a smaller subproblem than P_t
⇓
Never reach base case?!

Idea: Limit # of edges in path

What is the length of the shortest path from s to t with at most 2 edges? 3 edges? 4 edges?

A) 3, 1, 1 B) 3, 1, 3 C) 3, 2, 1 D) 2, 3, 4





A: 3, 1, 1
- 2 edges: 5−2=3
- 3 edges: 1+1−1=1
- 4 edges: 1+1−2+3=3 (not better than blue path, so use that instead.)

Define $P_{v,i}$ = Shortest s→v path with at most i edges
(⇑ add weights; ⇑ # edges)

[Note: a path with n or more edges must contain a cycle. Since no negative cycles → optimal path will never have cycle, so only need to go up to $i=n-1$.]

Group work

i gets smaller → recurrence relation, in terms of smaller subproblem!

1. $$P_{v,i} = \begin{cases} P_{u,i-1} + (u,v), & \text{if shortest path with } i \text{ edges goes through } u \text{ immediately prior to } v \\ P_{v,i-1}, & \text{If shortest } s \to v \text{ path with at most } i \text{ edges uses fewer than } i \text{ edges.} \end{cases}$$

$P_{s,0} = \emptyset$ $P_{v,0} = \text{Null}$ $(v \neq s)$ ← i decreases, so base case, make $i=0$. (Check later!)

2. Turn into objective function recurrence relation

$L(P_{v,i})$ = length of shortest s→v path with at most i edges

$L(P_{v,i}) = \min_{u \in V} \{ L(P_{u,i-1}) + w(u,v) \}$ ~~min { … , $L(P_{v,i-1})$ }~~ ($u \neq v$ struck out)

$\| \; L(P_{v,i-1}) + w(v,v)$ ← b/c = 0

$L(P_{s,0}) = 0$

$L(P_{v,0}) = \infty$ $(v \neq s)$

3. Write Pseudocode to Fill in A (don't need to work backward)

Input: Description of an n-vertex graph via an nxn array w, such that w[u,v] contains weight of edge (u,v). (Weight is infinity if no edge and 0 for w[v,v].) Starting vertex s

Output: Array A containing…? $A[v,i] = L(P_{v,i})$ (for all n vertices; for i from 0 to n−1)

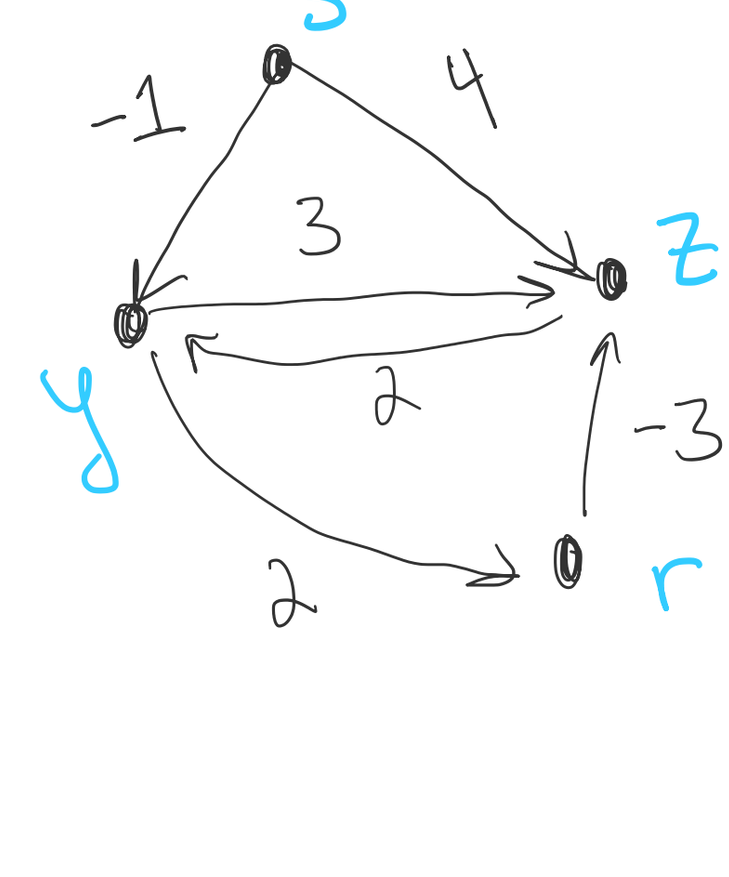
```
Initialize nxn array A containing ∞'s.
A[s,0]=0
for i = 1 to n-1:
    for v ∈ V:
        // Find minimum among options!
        for u ∈ V:
            if A[u,i-1]+w[u,v] < A[v,i]:
                A[v,i] ← A[u,i-1]+w[u,v]
```

A

vertices				
z	∞	∞	∞	∞
y	∞	∞	∞	∞
x	∞	∞	∞	∞
s	0	∞	∞	∞
	0	1	2	3

max edges in path

Example



r	∞	∞	∞, 1, ∞, ∞ → 1	
z	∞	4	4, 2, 4, ∞ → 2	
y	∞	-1	-1, -1, 6, ∞ → -1	
s	0	0	0	
	0	1	2	3

i

Runtime?

A) O(n) B) $O(n^2)$ C) $O(n^3)$ D) Need more info about graph

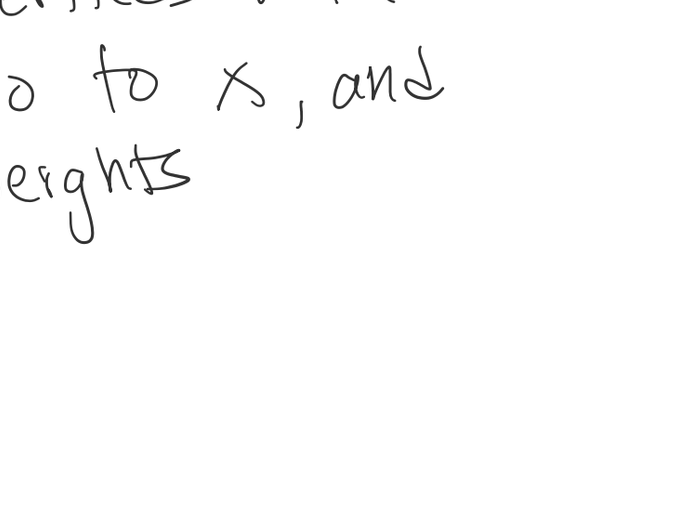
```
Initialize nxn array A containing ∞'s.      O(n^2)
A[s,0]=0
for i = 1 to n-1:                           ← O(n)
    for v ∈ V:                              ← O(n)
        // Find minimum among options!
        for u ∈ V:                          ← O(n)       } O(n^3)
            if A[u,i-1]+w[u,v] < A[v,i]:    } O(1)
                A[v,i] ← A[u,i-1]+w[u,v]
```

⇑ This is when input is array w ⇑ (adjacency matrix).

Only need consider this case if edge from u to v exists.

$$P_{v,i} = \begin{cases} P_{u,i-1} + (u,v), & \text{if shortest path with } i \text{ edges goes through } u \text{ immediately prior to } v \\ P_{v,i-1}, & \text{If shortest } s \to v \text{ path with at most } i \text{ edges uses fewer than } i \text{ edges.} \end{cases}$$

Reverse adjacency list:



W[x] = | y, w(y,x) | z, w(z,x) | ← list of all vertices with edges that go to x, and weights

W[z] = | q, w(q,z) | r, w(r,z) | y, w(y,z) |

```
Initialize nxn array A containing ∞'s.
A[s,0]=0
for i = 1 to n-1:                              ← O(n)
    for v ∈ V:                                 ← every vertex
        // Find minimum among options!         ← go through list W[v]
        for u ∈ V s.t. (u,v) ∈ E:              ← each edge going to v
            if A[u,i-1]+w[u,v] < A[v,i]:
                A[v,i] ← A[u,i-1]+w[u,v]
```

} Every edge in graph once
O(m)
↑
# of edges

Mini Amortized Analysis

In worst case: $O(m) = O(n^2)$, but if sparse graph can do better

$O(mn)$ runtime