

## Lecture23

---

# CSE 417: Algorithms with Complexity

## Lecture 23

### Shortest Paths Problem and Dynamic Programming

11/22/2023

CSE 417

1

# Announcements

- No class on Friday



# Shortest Path Problem

- **Dijkstra's Single Source Shortest Paths Algorithm**
  - $O(m \log n)$  time, positive cost edges
- **Bellman-Ford Algorithm**
  - $O(mn)$  time for graphs which can have negative cost edges

# Dynamic Programming

- Express problem as an optimization

0  
→



$$d_s[t] = \min_{v \in N^-(t)}$$

$$d_s[v] + c_{vt}$$

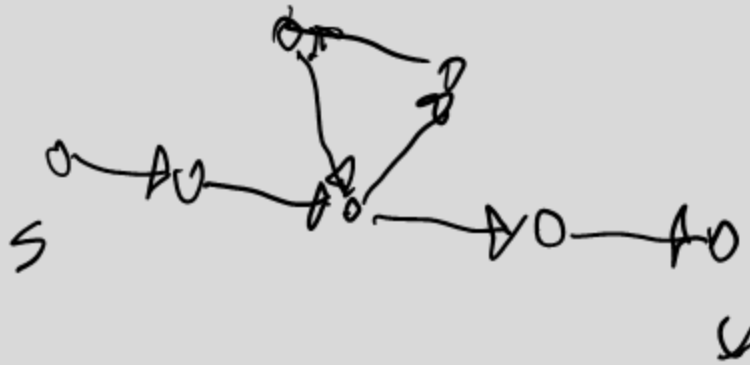
- Order subproblems so that results are computed in proper order

# Shortest Paths as DP

- $\text{Dist}_s[s] = 0$
- $\text{Dist}_s[v] = \min_w [\text{Dist}_s[w] + c_{wv}]$
- How do we order the computation
- Directed Acyclic graph: Topological Sort
- Dijkstra's algorithm determines an order

# Lemma

- If a graph has no negative cost cycles, then the **shortest** paths are **simple** paths
- Shortest paths have at most  $n-1$  edges



# Shortest paths with a given number of edges

- Find the shortest path from  $s$  to  $w$  with exactly  $k$  edges

# Express as a recurrence

- Compute distance from starting vertex  $s$
- $\text{Opt}_k(w) = \min_x [\text{Opt}_{k-1}(x) + c_{xw}]$
- $\text{Opt}_0(w) = 0$  if  $w = s$  and infinity otherwise





# Algorithm, Version 1

for each  $w$

$M[0, w] = \text{infinity};$

$M[0, s] = 0;$

for  $i = 1$  to  $n-1$

for each  $w$

$M[i, w] = \min_x (M[i-1, x] + \text{cost}[x, w]);$

$M[i, w]$  — length of  
 s,pa from  
 s to w ~~is~~  
 with

exactly  $i$   
 edges

at most

# Algorithm, Version 2



for each w

$M[0, w] = \text{infinity};$

$M[0, s] = 0;$

for i = 1 to n-1

for each w

$M[i, w] = \min(M[i-1, w], \min_x(M[i-1, x] + \text{cost}[x, w]));$

Bellman - Ford

# Algorithm, Version 3

for each w

$M[w] = \text{infinity};$

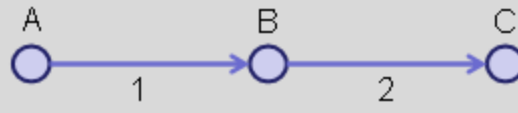
$M[s] = 0;$

for i = 1 to n-1

    for each w

$M[w] = \min(M[w], \min_x(M[x] + \text{cost}[x,w]));$

# Example:



Alg 1

A	B	C
0	-	-
-	1	-
-	-	3

Alg 2

A	B	C
0	-	-
0	1	-
0	1	3

Alg 3

A	B	C
0	-	-
0	1	3

A	B	C
0	-	-
0	1	-
0	1	3

# Correctness Proof for Algorithm 3

B-F

- Key lemmas, for all  $w$ :
  - There exists a path of length  $M[w]$  from  $s$  to  $w$
  - At the end of iteration  $i$ ,  $M[w] \leq M[i, w]$ ;

# Algorithm, Version 4

for each w

$M[w] = \text{infinity};$

$M[s] = 0;$

for i = 1 to n-1

    for each w

        for each x

            if ( $M[w] > M[x] + \text{cost}[x,w]$ )

$P[w] = x;$

$M[w] = M[x] + \text{cost}[x,w];$

11/22/2023

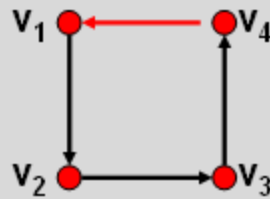
CSE 417

14

# Theorem



If the pointer graph has a cycle, then the graph has a negative cost cycle



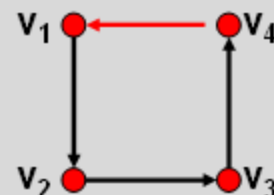
11/22/2023  
Proof: See text.

CSE 417

15

# If the pointer graph has a cycle, then the graph has a negative cost cycle

- If  $P[w] = x$  then  $M[w] \geq M[x] + \text{cost}(x, w)$ 
  - Equal when  $w$  is updated
  - $M[x]$  could be reduced after update
- Let  $v_1, v_2, \dots, v_k$  be a cycle in the pointer graph with  $(v_k, v_1)$  the last edge added
  - Just before the update
    - $M[v_j] \geq M[v_{j+1}] + \text{cost}(v_{j+1}, v_j)$  for  $j < k$
    - $M[v_k] > M[v_1] + \text{cost}(v_1, v_k)$
  - Adding everything up
    - $0 > \text{cost}(v_1, v_2) + \text{cost}(v_2, v_3) + \dots + \text{cost}(v_k, v_1)$





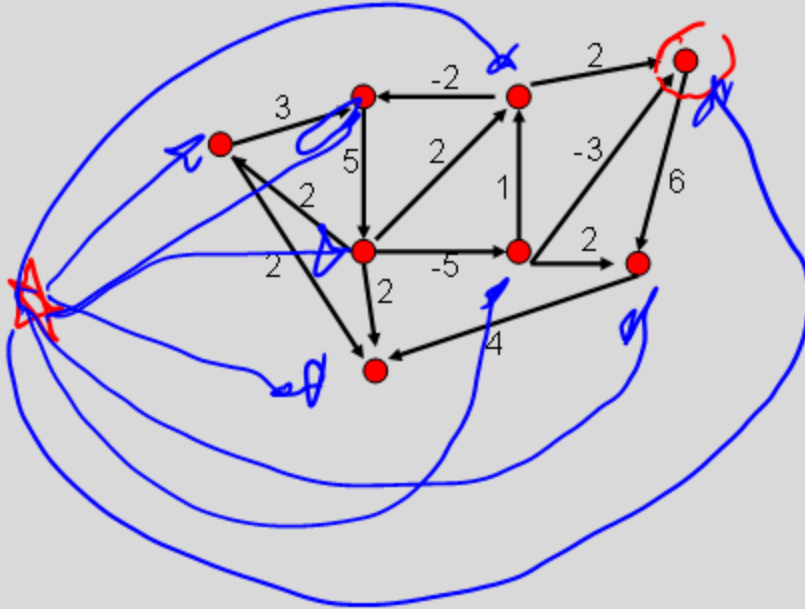
# Negative Cycles

- If the pointer graph has a cycle, then the graph has a negative cycle
- Therefore: if the graph has no negative cycles, then the pointer graph has no negative cycles

*Roadable from source S*

# Finding negative cost cycles

- What if you want to find negative cost cycles?

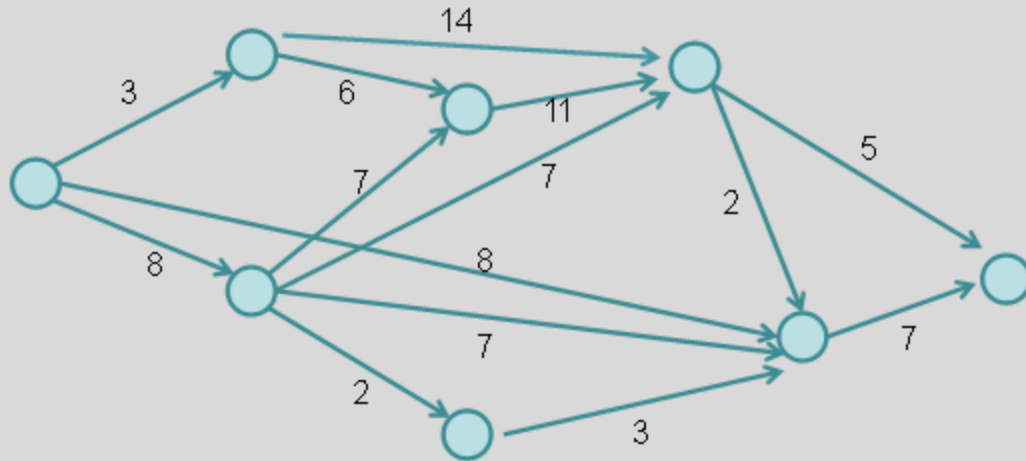


11/22/2023

CSE 417

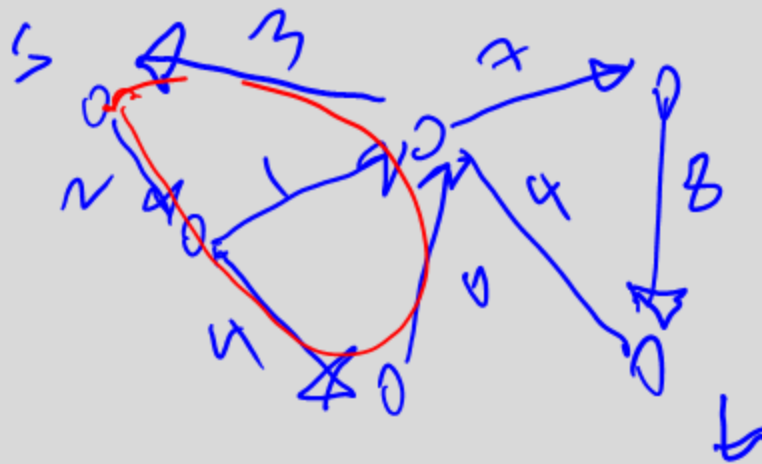
18

# Finding the longest Path in a DAG

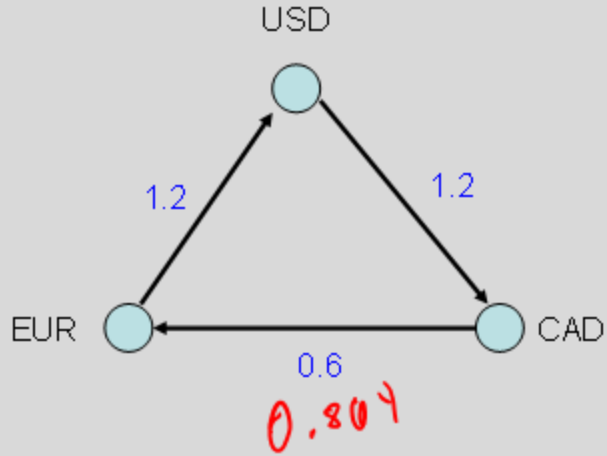


# What about finding Longest Paths in a directed graph

- Can we just change Min to Max?



# Foreign Exchange Arbitrage



	USD	EUR	CAD
USD	-----	0.8	1.2
EUR	1.2	-----	1.6
CAD	0.8	0.6	-----

