

Lecture22

CSE 417: Algorithms with Complexity

Lecture 22

Shortest Paths Problem and Dynamic Programming

3/1/2023

CSE 417

1

Announcements

- **Lecture Schedule – Winter Quarter is short**
 - Last four lectures will cover NP-Completeness
- **HW 8 – Dynamic Programming**
- **HW 9 - DP and NP Completeness**
- **Final exam – Monday, March 13, 8:30 AM**
 - More info on its way . . .

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

recursive

- Express problem as an optimization

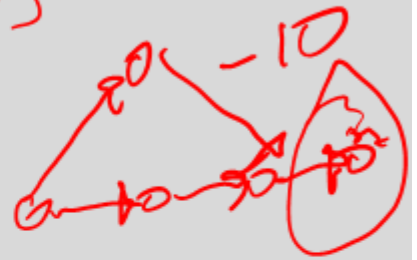
$$SP(s, u) = \min \{ SP(s, x) + \text{len}[x, u] \}$$

- 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

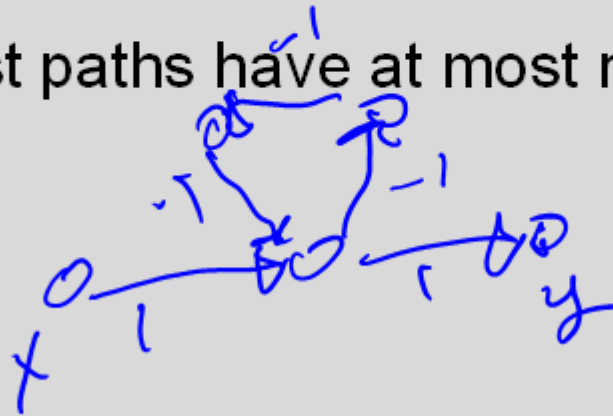
$$w \in N^-(v)$$



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

$SP(s, w, k)$

$SP(s, w, l)$

$SP(s, w, 2)$

$SP(s, w, n)$

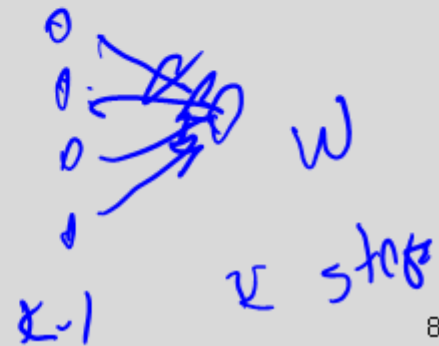
Express as a recurrence



- Compute distance from starting vertex s

$$\text{Opt}[k, w] \quad x \in N^-(w)$$

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



3/1/2023

CSE 417

8

Algorithm, Version 1

$$x \in N^-(w)$$

for each w

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

$M[0, s] = 0;$

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

 for each w

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

Algorithm, Version 2

for each w

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

$M[0, s] = 0;$

for i = 1 to n-1

for each w

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

paths $\leq k$
edges

Bellman Eqn.

Algorithm, Version 3

for each w

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

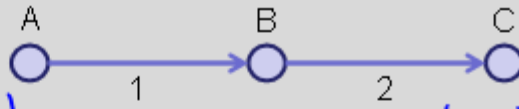
$M[s] = 0;$

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)
(B;C)

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

(B;C)
(A;B)

Correctness Proof for Algorithm 3

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

BF is at least as fast as Alg 2

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

$P[w] = x;$

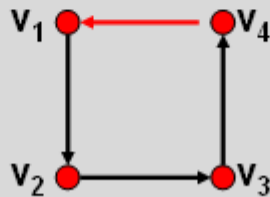
$M[w] = M[x] + \text{cost}[x,w] ;$

3/1/2023

14

Theorem

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



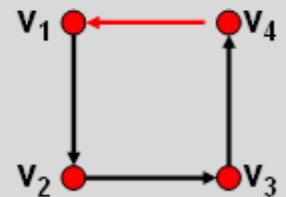
Proof: See text.

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
 - $0 > \text{cost}(v_1, v_2) + \text{cost}(v_2, v_3) + \dots + \text{cost}(v_k, v_1)$

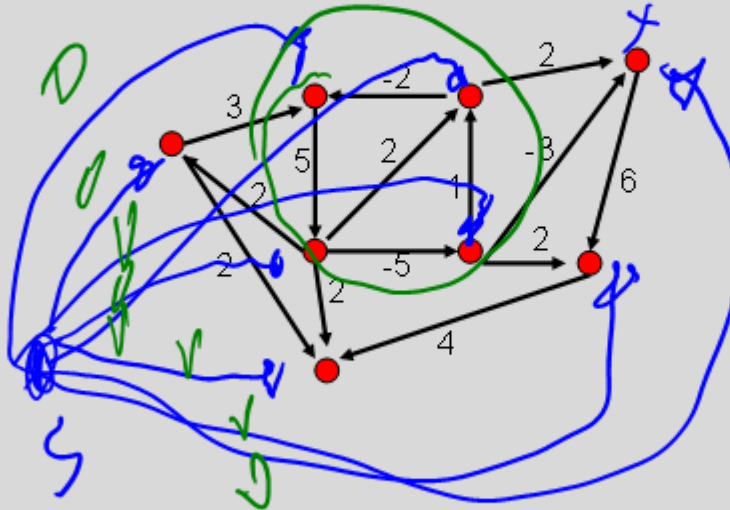


Negative Cycles

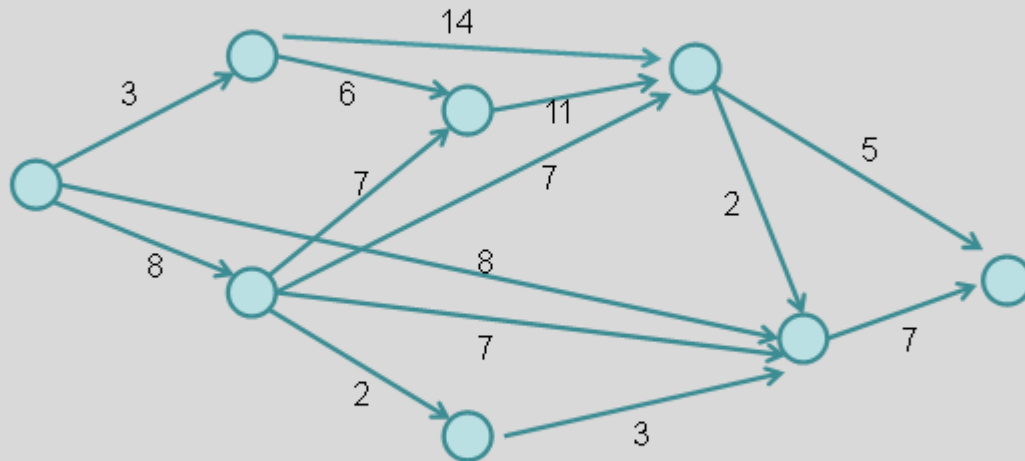
- If the pointer graph has a cycle, then the graph has a negative cycle
- Therefore: if the graph has no negative cycles, then the pointer graph has no negative cycles

Finding negative cost cycles

- What if you want to find negative cost cycles?



Finding the longest Path in a DAG

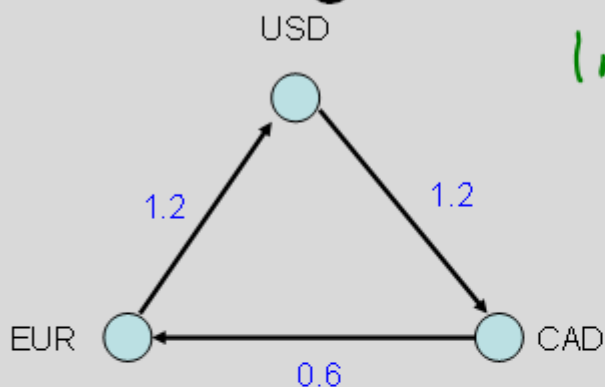


What about finding Longest Paths in a directed graph

- Can we just change Min to Max?

Foreign Exchange Arbitrage

$1.2 \times 1.2 \times 0.6$



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

$0.8 \times 0.8 \times 1.6 = 1.024$

