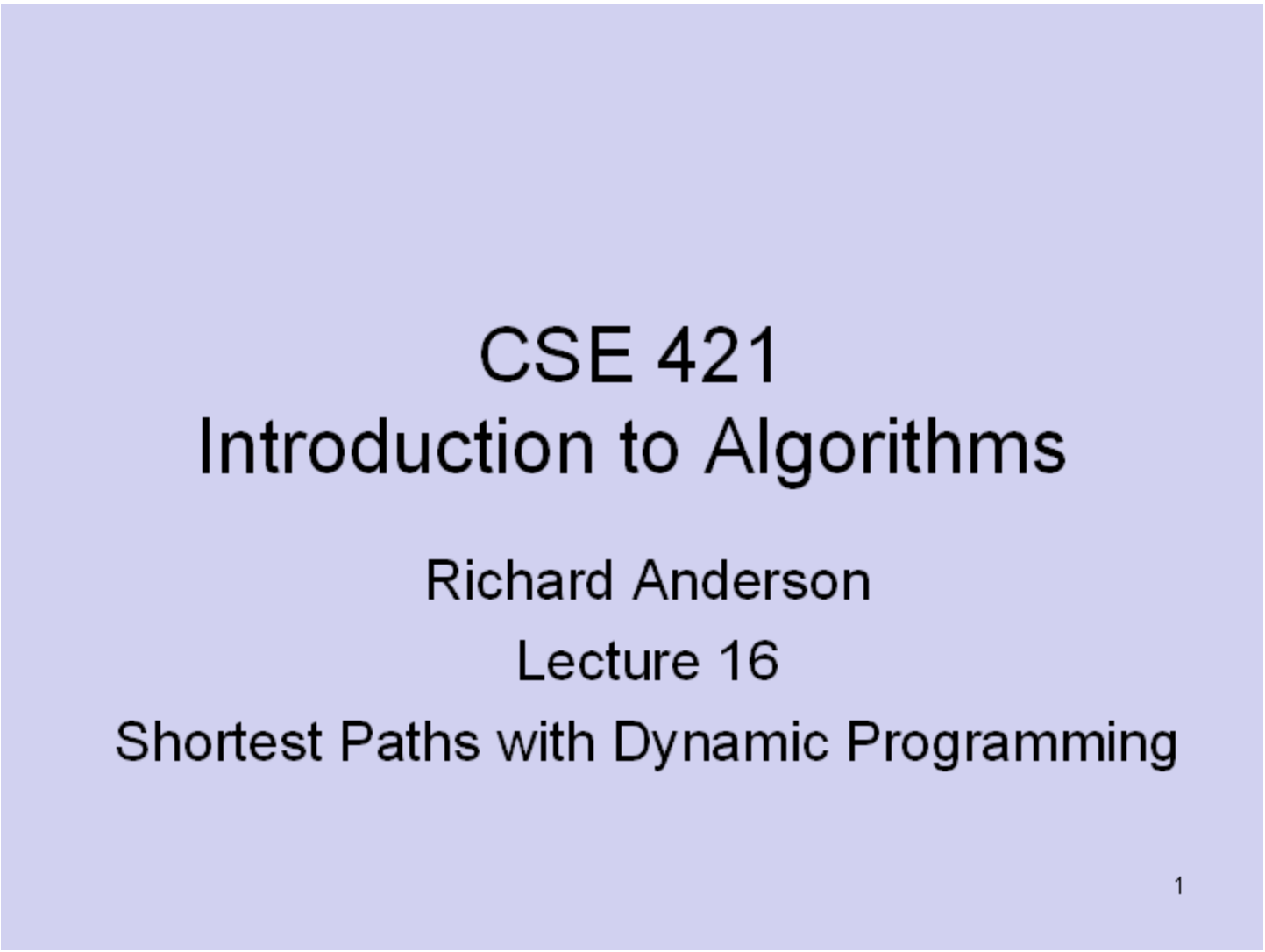
Lecture16


CSE 421
Introduction to Algorithms
Richard Anderson
Lecture 16
Shortest Paths with Dynamic Programming
1

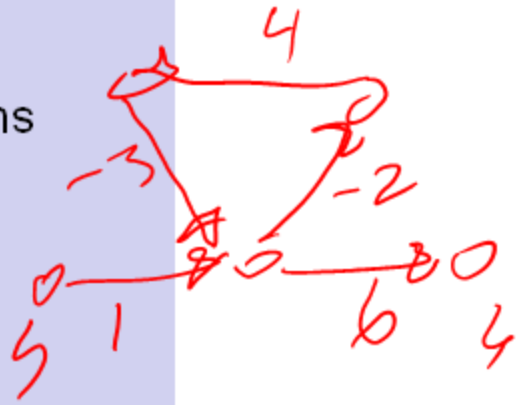
Announcements

- Dynamic Programming Reading:
  - 6.8 Shortest Paths (Bellman-Ford)
- Network Flow Reading
  - 7.1-7.3, Network Flow Problem and Algorithms
  - 7.5-7.12, Network Flow Applications

2

Shortest Path Problem

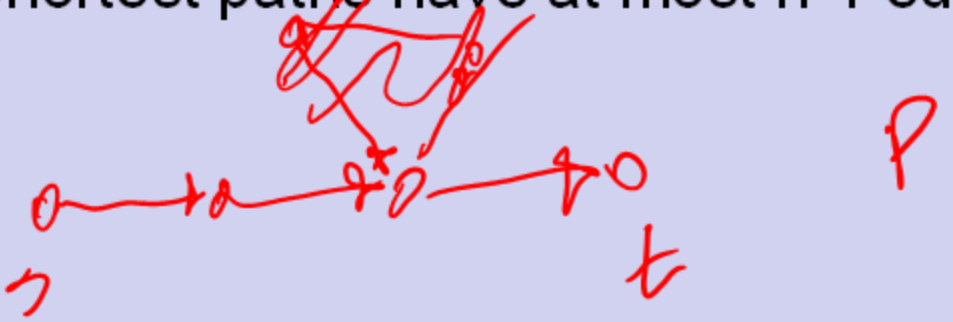
- Dijkstra's Single Source Shortest Paths Algorithm
  - O(m log n) time, positive cost edges
- Directed Acyclic Graphs
  - O(n + m), Topological Sort + DP
- Bellman-Ford Algorithm
  - O(mn) time for graphs which can have negative cost edges



3

Lemma

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



4

Shortest paths with a fixed number of edges

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

$$dist_k(s,w) = \min_{v \in N^-(w)} \left(dist_{k-1}(s,v) + c_{vw} \right)$$

$$dist_0(s,w) = \begin{cases} 0 & \text{if } s = w \\ \infty & \text{otherwise.} \end{cases}$$

5

Express as a recurrence

- Compute distance from starting vertex s

- $Opt_k(w) = \min_x [Opt_{k-1}(x) + c_{xw}]$
- $Opt_0(w) = 0$ if w = s and infinity otherwise

6

Algorithm, Version 1

space $O(n^2)$

runtime $O(n^3)$ $O(mn)$

```
for each w
    M[0, w] = infinity;
M[0, s] = 0;
for i = 1 to n-1
    for each w
        M[i, w] = min_x(M[i-1,x] + cost[x,w]);
```

7

paths of length i ⇒ paths of length ≤ i

Algorithm, Version 2

```
for each w
    M[0, w] = 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] + cost[x,w]));
```

8

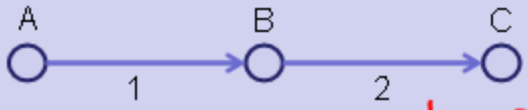
Bellman-Ford

Algorithm, Version 3

```
for each w
    M[w] = infinity;
M[s] = 0;
for i = 1 to n-1
    for each w
        M[w] = min(M[w], min_x(M[x] + cost[x,w]));
```

9

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	-
0	1	3

BC
AB

A	B	C
0	-	-
0	1	3

AB
BC

10

Correctness Proof for Algorithm 3

- Key lemma – at the end of iteration i, for all w, $M[w] \leq M[i, w]$;

11

Algorithm, Version 4

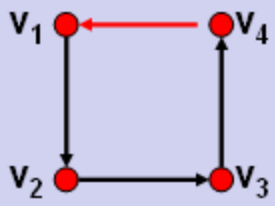
```
for each w
    M[w] = infinity;
M[s] = 0;
for i = 1 to n-1
    for each w
        for each x
            if (M[w] > M[x] + cost[x,w])
                P[w] = x;
                M[w] = M[x] + cost[x,w] ;
```

Check if cycle

12

Theorem

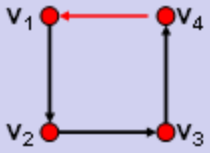
If the pointer graph has a cycle, then the graph has a negative cost cycle



13

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

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



14

M[1] ≥ M[2] + c_{12}

M[2] ≥ M[3] + c_{32}

M[3] ≥ M[4] + c_{43}

M[4] ≥ M[1] + c_{14}

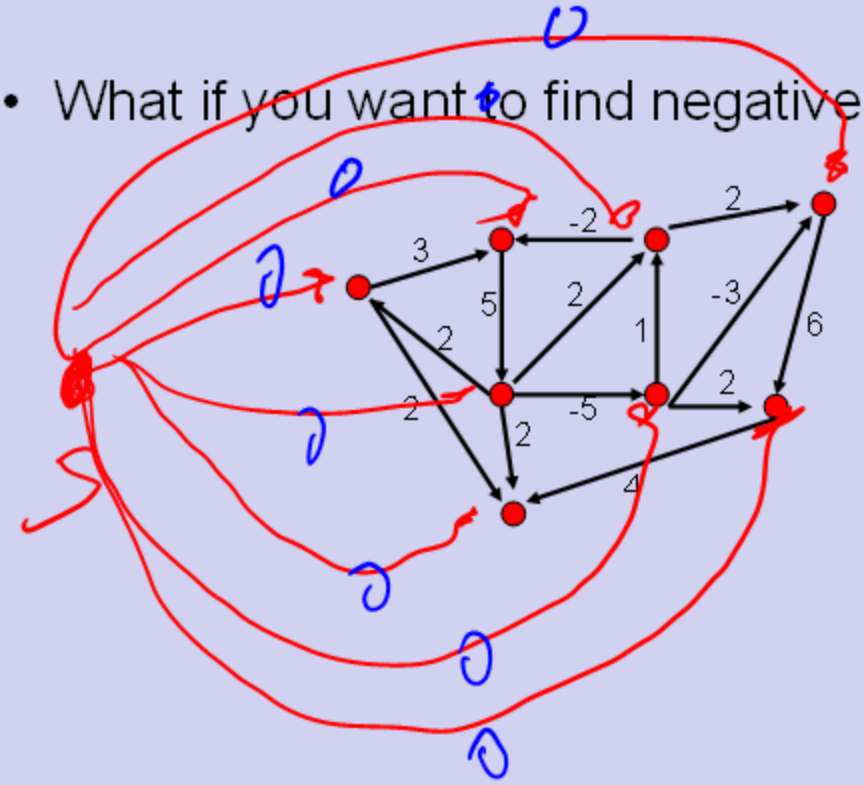
Negative Cycles

- 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

15

Finding negative cost cycles

- What if you want to find negative cost cycles?



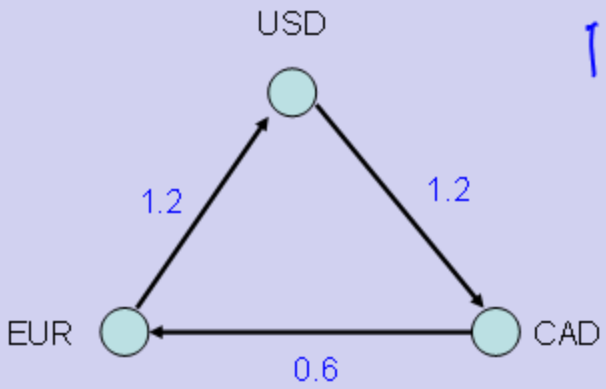
16

What about finding Longest Paths

- Can we just change Min to Max?

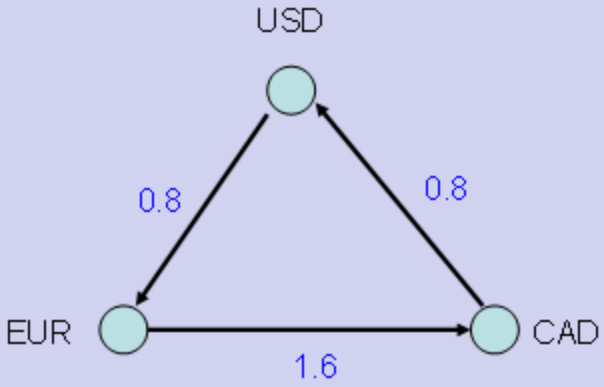
17

Foreign Exchange Arbitrage



1.44 x .6 = .08...

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



.64
1.6
1.024

18