CSE 417: Algorithms with Complexity

Lecture 23 – Autumn 2024

Shortest Paths Problem and Dynamic Programming

1

Announcements

- Lecture plans
  - Today: Shortest Paths
  - Friday-Wednesday: Network Flow
  - After Thanksgiving: NP Completeness
- HW 8 Available

2

Shortest Path Problem

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

3

Dynamic Programming

- Express problem as an optimization
- Order subproblems so that results are computed in proper order

4

Shortest Paths as DP

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

5

Lemma

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

6

Shortest paths with a given number of edges

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

7

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

8

Algorithm, Version 1

```
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]);
```

9

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]));
```

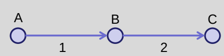
10

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]));
```

11

Example:



A	B	C

A	B	C

A	B	C

A	B	C

12

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] ≤ M[i, w];

13

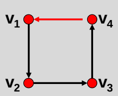
Algorithm, Version 4

```
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] ;
```

14

Theorem

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

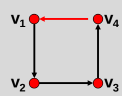


Proof: See text.

15

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
    - $0 > cost(v_1, v_2) + cost(v_2, v_3) + \ldots + cost(v_k, v_1)$



16

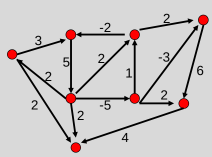
Negative Cycles

- 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

17

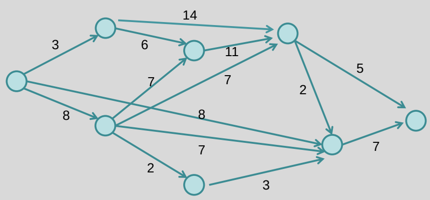
Finding negative cost cycles

- What if you want to find negative cost cycles?



18

Finding the longest Path in a DAG



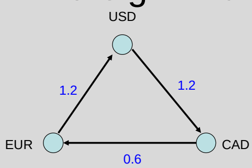
19

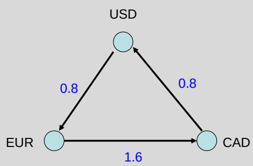
What about finding Longest Paths in a directed graph

- Can we just change Min to Max?

20

Foreign Exchange Arbitrage





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

21