CSE 421
Algorithms

Richard Anderson

Lecture 21

Shortest Paths and Network Flow

Shortest Paths with Dynamic Programming

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 with negative cost edges

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 fixed number of edges

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

Express as a recurrence

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

Algorithm, Version 1

foreach w
    M[0, w] = infinity;
M[0, v] = 0;
for i = 1 to n-1
    foreach w
        M[i, w] = $\min_x$(M[i-1,x] + cost[x,w]);

Algorithm, Version 2

foreach w
    M[0, w] = infinity;
M[0, v] = 0;
for i = 1 to n-1
    foreach w
        M[i, w] = min(M[i-1, w], $\min_x$(M[i-1,x] + cost[x,w]))

Algorithm, Version 3

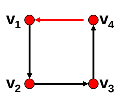
foreach w
    M[w] = infinity;
M[v] = 0;
for i = 1 to n-1
    foreach w
        M[w] = min(M[w], $\min_x$(M[x] + cost[x,w]))

Correctness Proof for Algorithm 3

- Key lemma – at the end of iteration i, for all w, M[w] <= M[i, w];
- Reconstructing the path:
  - Set P[w] = x, whenever M[w] is updated from vertex x

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
    - $M[v_j] >= M[v_{j+1}] + cost(v_{j+1}, v_j)$ for j < k
    - $M[v_k] > M[v_1] + cost(v_1, v_k)$
  - Adding everything up
    - $0 > cost(v_1,v_2) + cost(v_2,v_3) + \ldots + cost(v_k, v_1)$

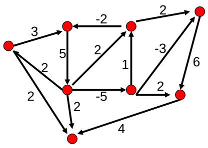


Negative Cycles

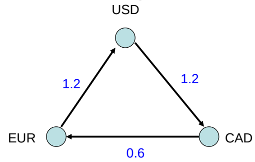
- If the pointer graph has a cycle, then the graph has a negative cycle
- Therefore: if the graph has no negative cycles, then the pointer graph has no negative cycles

Finding negative cost cycles

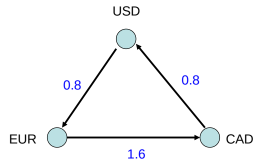
- What if you want to find negative cost cycles?



Foreign Exchange Arbitrage



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



Network Flow

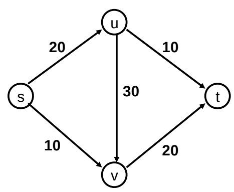
Outline

- Network flow definitions
- Flow examples
- Augmenting Paths
- Residual Graph
- Ford Fulkerson Algorithm
- Cuts
- Maxflow-MinCut Theorem

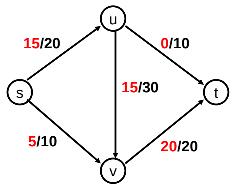
Network Flow Definitions

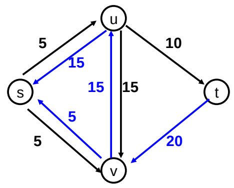
- Capacity
- Source, Sink
- Capacity Condition
- Conservation Condition
- Value of a flow

Flow Example



Flow assignment and the residual graph

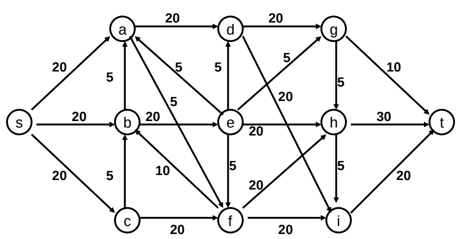




Network Flow Definitions

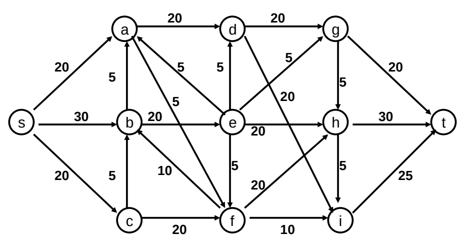
- Flowgraph: Directed graph with distinguished vertices s (source) and t (sink)
- Capacities on the edges, c(e) >= 0
- Problem, assign flows f(e) to the edges such that:
  - 0 <= f(e) <= c(e)
  - Flow is conserved at vertices other than s and t
    - Flow conservation: flow going into a vertex equals the flow going out
  - The flow leaving the source is a large as possible

Flow Example



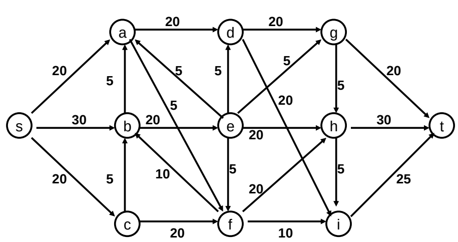
Find a maximum flow

Value of flow:



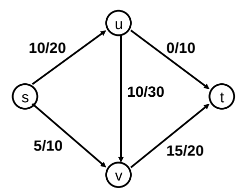
Construct a maximum flow and indicate the flow value

Find a maximum flow

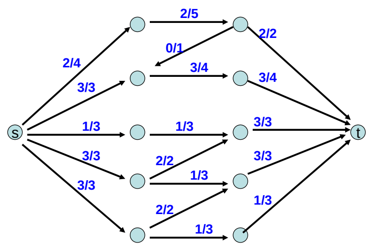


Augmenting Path Algorithm

- Augmenting path
  - Vertices $v_1, v_2, \ldots, v_k$
    - $v_1 = s$, $v_k = t$
    - Possible to add b units of flow between v_j and v_{j+1} for j = 1 … k-1



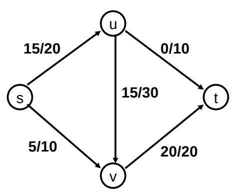
Find two augmenting paths

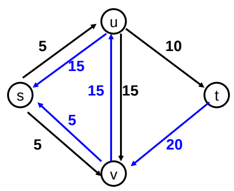


Residual Graph

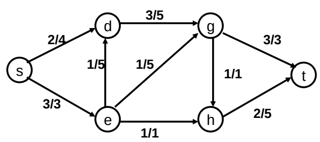
- Flow graph showing the remaining capacity
- Flow graph G, Residual Graph G_R
  - G: edge e from u to v with capacity c and flow f
  - G_R: edge e' from u to v with capacity c – f
  - G_R: edge e'' from v to u with capacity f

Residual Graph

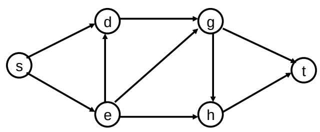




Build the residual graph

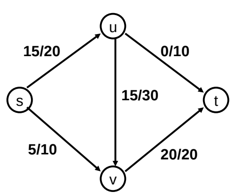


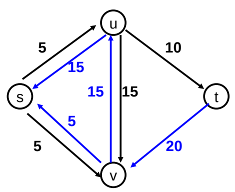
Residual graph:



Augmenting Path Lemma

- Let P = v_1, v_2, …, v_k be a path from s to t with minimum capacity b in the residual graph.
- b units of flow can be added along the path P in the flow graph.





Proof

- Add b units of flow along the path P
- What do we need to verify to show we have a valid flow after we do this?
  –
  –

Ford-Fulkerson Algorithm (1956)

while not done

  Construct residual graph G_R

  Find an s-t path P in G_R with capacity $b > 0$

  Add b units along in G

If the sum of the capacities of edges leaving S is at most C, then the algorithm takes at most C iterations