Course Notes for

CS 1501
Algorithm Implementation

By
John C. Ramirez
Department of Computer Science
University of Pittsburgh

- These notes are intended for use by students in CS1501 at the University of Pittsburgh and no one else
- These notes are provided free of charge and may not be sold in any shape or form
- These notes are NOT a substitute for material covered during course lectures. If you miss a lecture, you should definitely obtain both these notes and notes written by a student who attended the lecture.
- Material from these notes is obtained from various sources, including, but not limited to, the following:
    - Algorithms in C++ by Robert Sedgewick
    - Algorithms, 4th Edition by Robert Sedgewick and Kevin Wayne
    - Introduction to Algorithms, by Cormen, Leiserson and Rivest
    - Various Java and C++ textbooks
    - Various online resources (see notes for specifics)

Previous Lecture

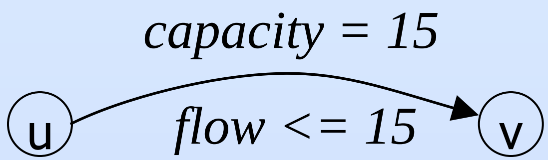
- Note that we did not cover all of the material from Lecture 18 in our previous synchronous lecture
- Specifically, we did not get to Djikstra's Shortest Path algorithm
- We will cover this in the first part of our synchronous Lecture 19, and there will be less new material in this leture
- Review the last part of Lecture 18 prior to our synchronous Lecture 19

- Definitions:
  - Consider a directed, weighted graph with set V of vertices and set E of edges, with each edge weight indicating a capacity, c(u,v) for that edge
    - c(u,v) = 0 means no edge between u and v
  - Consider a source vertex, s with no in-edges and a sink vertex, t with no out-edges
  - A FLOW on the graph is another assignment of weights to the edges, f(u,v) such that the following rules are satisfied:

1) For All (u,v) in V, f(u,v) <= c(u,v)

– No flow can exceed the capacity of the edge

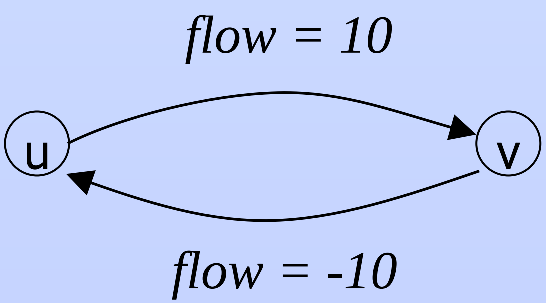
– Ex:



2) For All (u,v) in V, f(u,v) = – f(v,u)

– If a positive flow is going from u to v, than an equal weight negative flow is going from v to u

– Ex:

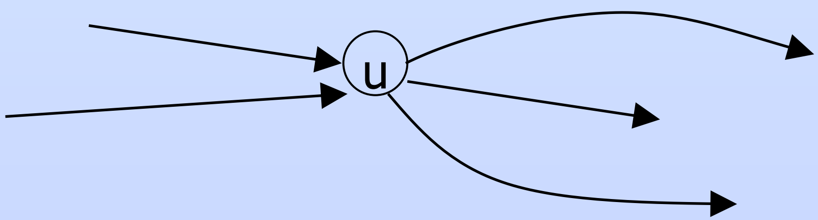


- Think about pouring water from some pitcher A into some other pitcher B at 50 ml/sec
- The flow from A to B is 50 ml / sec
- The flow from B to A is -50 ml / sec

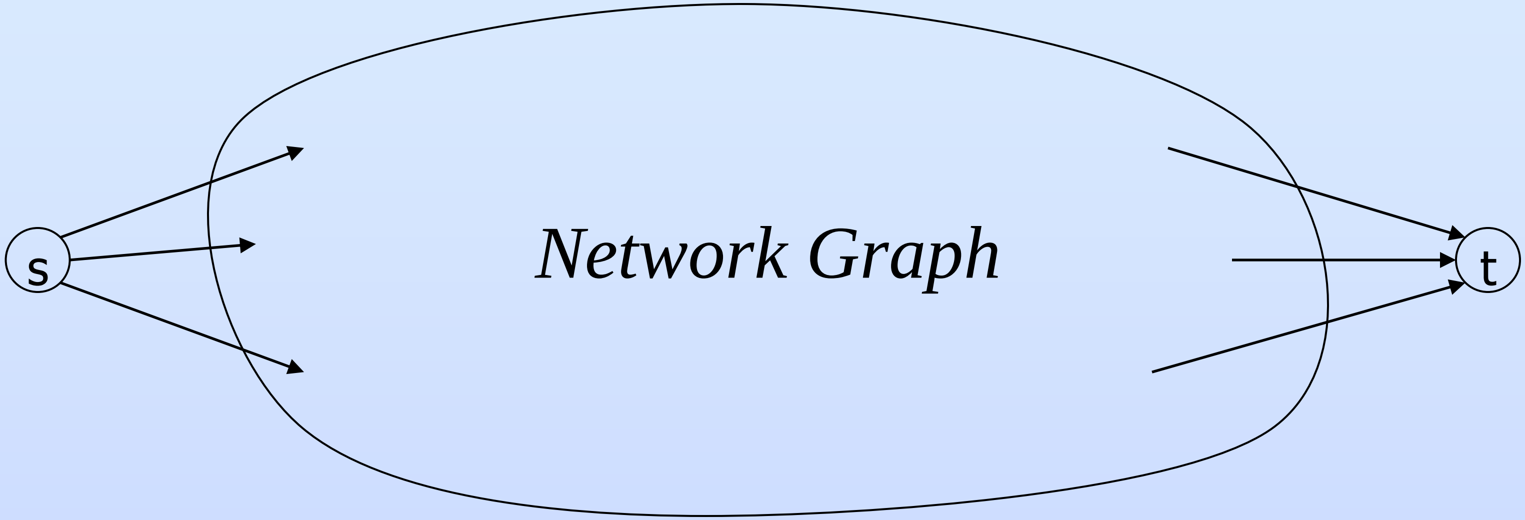
3) For All u in V – {s, t}, sum of (flow on edges from u) = 0

– All vertices except the source and sink have an overall "net" flow of 0 – the amount coming in is the same as the amount going out

– No vertices "keep" any of the flow



- Problem: What is the maximum flow for a given network and how can we determine it?

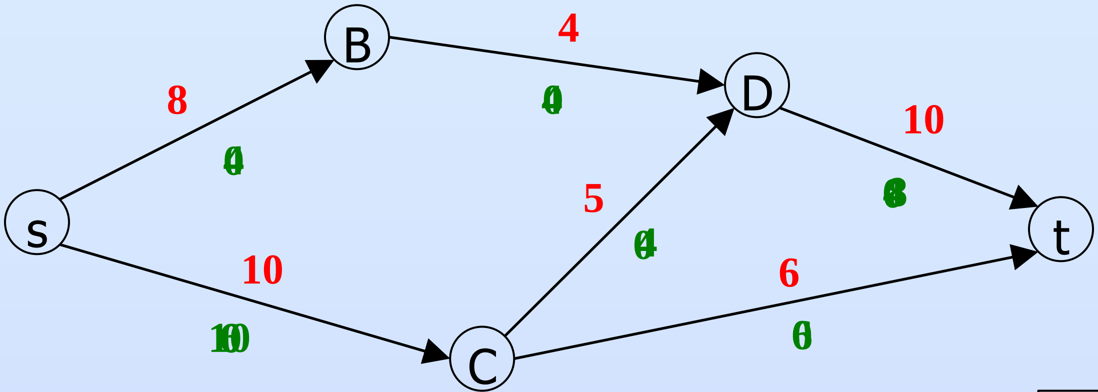


- Assume that an infinite amount of flow can come from s
- Assume that an infinite amount of flow can go into t
- The only restriction on the total is the capacities of the edges in the graph
- Given the capacities, how much flow can we push from s to t?

- Ford-Fulkerson approach:
  - For a given network with a given flow, we look for an augmenting path from the source to the sink
    - An augmenting path is a path from the source to the sink that provides additional flow along it
  - After finding an augmenting path, we update the residual network to reflect the additional flow
  - We repeat this process on the residual network until no augmenting path can be found
  - At this point, the maximum flow has been determined

red = capacity
green = flow

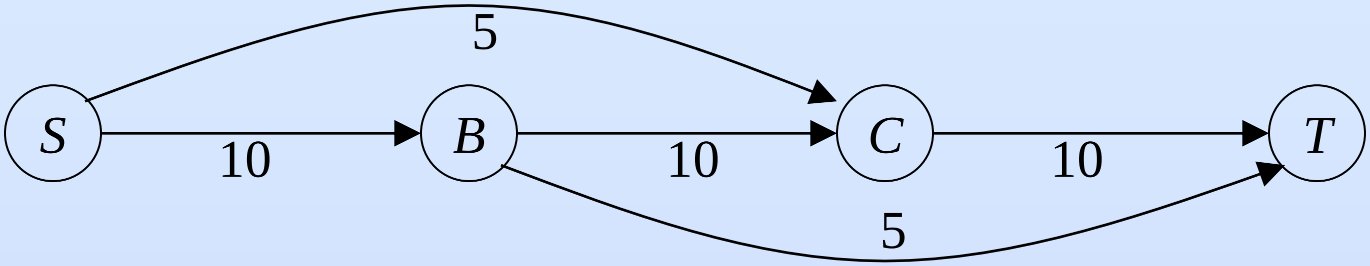
Network Flow



- Try path: sCt
  - This has an augment of 6 (min weight edge in path)
  - Update residual graph to reflect the flow
- Try path: sBDt
  - Update augment of 4 (min weight edge in path)
- Try path: sCDt
  - Update augment of 4 (min weight edge in path)
- No more augmenting paths
- Final flow is 6 + 4 + 4 = 14

- Note that after path sCDt is found we can no longer get from s to t
- Note that edges BD and sC are both saturated – used to capacity
- This creates a CUT in the graph, separating s from t

- Note the following example:



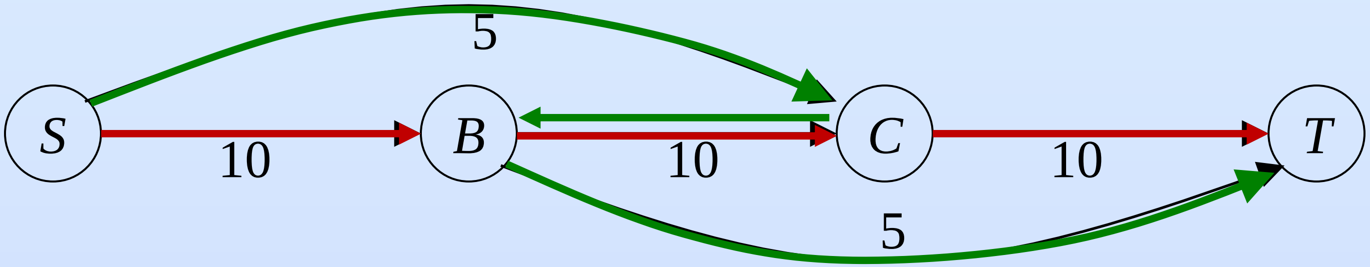
- Let's look at 2 possible sequences of augmenting paths:

1) SBT (5), SCT (5), SBCT (5)
   - Total Flow is 15
2) SBCT (10), ????
   - We seem to be stuck - there is no forward path from S to T
     - We could do SC but then what?
     - BT is open but how do we get to B?
   - Yet the choice of paths should not affect our max flow

Backward Flow

- How to resolve this problem?
  - The augmenting paths do not represent final flow values for given edges
    - They are simply a tool for determining the overall maximum flow
  - A given assignment for an edge in one path may be MORE than what the final network flow would indicate
    - Idea is that we overassigned it during an augmenting path, and the actual flow on that edge will in fact be less than the assigned value
  - Backward flow is a way to lessen the flow on a given edge while increasing the overall flow in the graph
  - Let's reconsider the previous example

Backward Flow



- We first find augmenting path SBCT with flow 10
- We next start an augmenting path with SC
  - At C we cannot move forward to T
  - But we CAN move backward to B
    - This would REDUCE the flow on BC from 10 to 5
    - Because a flow of 5 from C to B would be -5 from B to C
    - This is only possible because there is already a flow assigned to BC
    - We reduce it by 5 on this edge but overall the flow through the graph is still positive because…
  - We can now move forward from B to T
  - So overall we have path SCBT with flow 5

Backward Flow

  - We now have no more augmenting paths and our total flow is 10 + 5 = 15
    - The same flow we achieved with the other paths
- Note that the final assignment on the edges for both sequences of augmenting paths is the same
  - This is not always the case
  - If the assignment of maximum flow is unique, they will be the same
  - If more than one assignment gives the same value for maximum flow, difference choices for augmenting paths can result in different assignments
    - But the overall flow will always be the same

- To implement FF, we need
  - A way to represent the graph
  - We will call the graph a FlowNetwork
    - It will consist of an adjacency list of FlowEdge
      - Similar to DirectedEdge but…
    - Each FlowEdge will have:
      - a "from" vertex (v)
      - a "to" vertex (w)
      - a capacity value (how much flow can it take?)
      - a flow value (how much flow is actually on it?)
      - a method to determine residual capacity (how much more flow can it take?)

Implementing FF approach

- Note that the residual capacity is different depending on the "direction" from which we are looking at the edge
  - If we look in the forward direction, it is `(capacity - flow)`
    - How much capacity is left?
  - If we look in the backward direction, it is `flow`
    - We can "remove" up to the amount currently assigned to the edge

- See FlowEdge.java and FlowNetwork.java