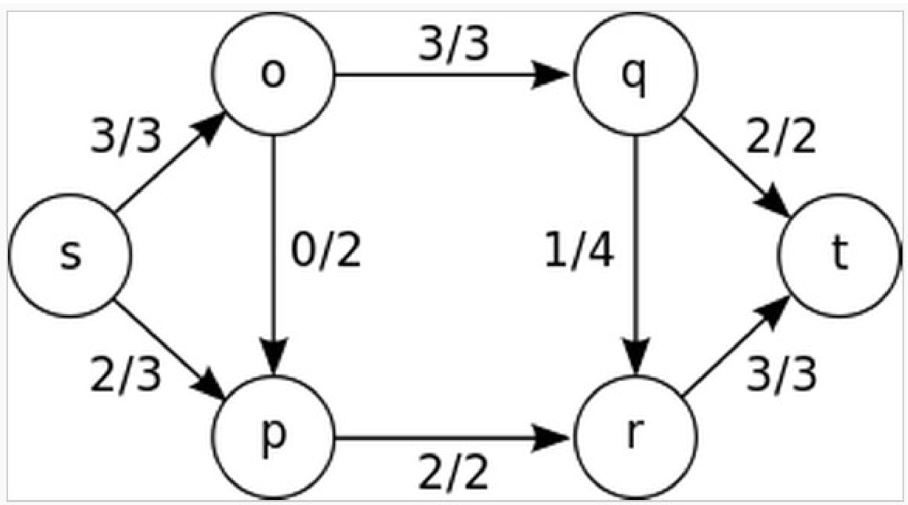
Algorithms: Lab

Implementation and Analysis of "Wave Algorithm"

By:
Ankush Shah
Institute of Computer Science
University of Bonn, Germany

Maximum Flow Problem

- Optimization problem
- finding a feasible flow through a single-source, single-sink flow network that is maximum.



An example of a flow network with a maximum flow. The source is *s*, and the sink *t*. The numbers denote flow and capacity.

souce: http://en.wikipedia.org/wiki/Maximum_flow_problem#Definition

Maximum Flow Problem: Algorithms

- Ford–Fulkerson algorithm: $O(E*max\_flow)$
- Edmonds–Karp algorithm: $O(VE^2)$
- Dinic's algorithm: $O(V^2E)$
  - **Blocking Flow**: $O(VE)$
  - Iteration: $O(V)$

Blocking Flow Problem

- Blocking Flow: A flow is blocking if there is a saturated edge on every a path from source to destination.

- Karzanov's algorithm: $O(V^2)$
  - Conceptually Complicated
- **Wave algorithm: $O(V^2)$**
  - **Conceptually Simple**

Wave Algorithm

- Repeat
  - Forward Pass: Push as much as possible
  - Backward Pass: Back up flow which is not reaching the destination. In the next forward pass this backup flow will sent through alternate paths, if available.

Wave Algorithm

- Terms
  - Preflow
  - Blocking preflow
  - Balanced / Unbalanced vertices
  - **Blocked / Unblocked vertices**

Wave Algorithm

- Topologically sort the vertices. Use DFS
  - show demo
- Initialise: Block s, and saturate (s,v)
- Repeat until all vertices are balanced
  - Increase Flow
    - Scan the vertices in topological order.
    - Try to balance the unblock vertices. Unable to balance, then block them.
    - If there is a blocked, unbalanced vertex, call DecreaseFlow
  - Decrease Flow
    - Scan the vertices in reverse topological order
    - Balance the blocked vertices.
    - If there is an unblocked unbalanced vertex, call IncreaseFlow

Analysis

- After increase flow, at least one vertex is blocked
- A blocked vertex which is balanced in a decrease flow remains balanced in subsequent steps
- (n iteration) * (n vertices)
- $O(n^2)$