Admin

Assignment 7 graded

Assignment 9

Checkpoint 3

Greedy algorithms -> all pairs shortest paths

Could ask to decide greedy vs. DP, but no DP solutions/algorithms

(will not include network flow)

2 pages of notes

Student networking

You decide to create your own computer network:

- You get three of your friends and string some network cables
- Because of capacity (due to cable type, distance, computer, etc) you can only send a certain amount of data to each person
- If edges denote capacity, what is the maximum throughput you can send from S to T?
Student networking

You decide to create your own campus network:
- You get three of your friends and string some network cables
- Because of capacity (due to cable type, distance, computer, etc) you can only send a certain amount of data to each person
- If edges denote capacity, what is the maximum throughput you can you send from S to T?

Another flow problem

How much water flow can we continually send from s to t?

Another flow problem

Flow graph/networks

Flow network
- directed, weighted graph (V, E)
- positive edge weights indicating the “capacity” (generally, assume integers)
- contains a single source s ∈ V with no incoming edges
- contains a single sink/target t ∈ V with no outgoing edges
- every vertex is on a path from s to t
What are the constraints on flow in a network?

Flow: in-flow = out-flow for every vertex (except s, t)

Flow along an edge cannot exceed the edge capacity

Flows are positive

Max flow problem

Given a flow network: what is the maximum flow we can send from s to t that meet the flow constraints?

Applications?

- Network flow
  - Water, electricity, sewage, cellular...
  - Traffic/transportation capacity
- Bipartite matching
- Sports elimination
- …
Max flow origins

Rail networks of the Soviet Union in the 1950’s

The US wanted to know how quickly the Soviet Union could get supplies through its rail network to its satellite states in Eastern Europe.

In addition, the US wanted to know which rails it could destroy most easily to cut off the satellite states from the rest of the Soviet Union.

These two problems are closely related: solving the max flow problem also solves the min cut problem of figuring out the cheapest way to cut off the Soviet Union from its satellites.

Source: Ibackstrom, The Importance of Algorithms, at www.topcoder.com

Algorithm idea

Algorithm idea

Algorithm idea

Algorithm idea

Source: Ibackstrom, The Importance of Algorithms, at www.topcoder.com
Algorithm idea

reroute some of the flow

Total flow?

Algorithm idea

send some flow down a path
Algorithm idea

send some flow down a path

Algorithm idea

send some flow down a path

Algorithm idea

reroute some of the flow

Algorithm idea

Are we done?
Is this the best we can do?
A cut is a partitioning of the vertices into two sets $S_s$ and $S_t = V - S_s$.

Flow across cuts

The flow “across” a cut is the total flow from nodes in $S_s$ to nodes in $S_t$, minus the total from nodes in $S_t$ to $S_s$.

What is the flow across this cut?

$$10 + 10 - 6 = 14$$
Flow across cuts

Consider any cut where \( s \in S \) and \( t \in S_t \), i.e. the cut partitions the source from the sink.

What do we know about the flow across the any such cut?

The flow across ANY such cut is the same and is the current flow in the network.

\[ 4 + 10 = 14 \]

\[ 4 + 6 + 4 = 14 \]
Consider any cut where $s \in S$, and $t \in S$, i.e. the cut partitions the source from the sink.

10 + 10 - 6 = 14

The flow across ANY such cut is the same and is the current flow in the network.

Why? Can you prove it?

Inductively?

- every vertex is on a path from $s$ to $t$
- in-flow = out-flow for every vertex (except $s$, $t$)
- flow along an edge cannot exceed the edge capacity
- flows are positive

Base case: $S = s$

- Flow is total from from $s$ to $t$; therefore the total flow out of $s$ should be the flow
- All flow from $s$ gets to $t$
  - every vertex is on a path from $s$ to $t$
  - in-flow = out-flow
Flow across cuts

The flow across ANY such cut is the same and is the current flow in the network

Inductive case: Consider moving a node $x$ from $S_t$ to $S_s$, how do the flows change?

Is the flow across the difference partitions the same?

Inductive case: Consider moving a node $x$ from $S_t$ to $S_s$. How do the flows change?

Flow across cuts

Consider any cut where $s \in S_s$ and $t \in S_t$, i.e. the cut partitions the source from the sink

The flow across ANY such cut is the same and is the current flow in the network

Capacity of a cut

The “capacity of a cut” is the maximum flow that we could send from nodes in $S_s$ to nodes in $S_t$ (i.e. across the cut)

How do we calculate the capacity?
Capacity of a cut

The “capacity of a cut” is the maximum flow that we could send from nodes in \( S \) to nodes in \( S' \), (i.e. across the cut)

Capacity is the sum of the edges from \( S \) to \( S' \).

\[ 10 + 9 = 19 \]

Why?

Any more and we would violate the edge capacity constraint
Any less and it would not be maximal, since we could simply increase the flow
Quick recap

A cut is a partitioning of the vertices into two sets $S_s$ and $S_t = V - S_s$.

For any cut where $s \in S_s$ and $t \in S_t$, i.e., the cut partitions the source from the sink:
- the flow across any such cut is the same
- the maximum capacity (i.e., flow) across the cut is the sum of the capacities for edges from $S_s$ to $S_t$.

Maximum flow

For any cut where $s \in S_s$ and $t \in S_t$:
- the flow across the cut is the same
- the maximum capacity (i.e., flow) across the cut is the sum of the capacities for edges from $S_s$ to $S_t$.

Are we done?
Is this the best we can do?

Maximum flow

What is the minimum capacity cut for this graph?

Capacity = $10 + 4$

Is this the best we can do?
Maximum flow

What is the minimum capacity cut for this graph?

Capacity = 10 + 4

flow = minimum capacity, so we can do no better

Algorithm idea

send some flow down a path

send some flow down a path

Search for a path with remaining capacity from s to t

reroute some of the flow

How do we determine the path to send flow down?

How do we handle "rerouting" flow?
During the search, if an edge has some flow, we consider “reversing” some of that flow.

The residual graph $G_f$ is constructed from $G$.

For each edge $e$ in the original graph $G$:
- if $\text{flow}(e) < \text{capacity}(e)$:
  - introduce an edge in $G_f$ with capacity $= \text{capacity}(e) - \text{flow}(e)$
  - this represents the remaining flow we can still push
- if $\text{flow}(e) > 0$:
  - introduce an edge in $G_f$ in the opposite direction
  - capacity $= \text{flow}(e)$
  - this represents the flow that we can reroute/reverse

Find a path from $s$ to $t$ in $G_f$. 
Algorithm idea

Find a path from $s$ to $t$ in $G_f$

None exist... done!
Find a path from $s$ to $t$ in $G_f$.
Ford-Fulkerson:

Ford-Fulkerson(G, s, t)

\[ \text{flow} = 0 \text{ for all edges} \]

\[ G_f = \text{residualGraph}(G) \]

while a simple path exists from s to t in \( G_f \),

send as much flow along the path as possible

\[ G_f = \text{residualGraph}(G) \]

return flow

Ford-Fulkerson: is it correct?

Does the function terminate?
- Every iteration increases the flow from s to t
- The flow is bounded by the min-cut

Ford-Fulkerson(G, s, t)

\[ \text{flow} = 0 \text{ for all edges} \]

\[ G_f = \text{residualGraph}(G) \]

while a simple path exists from s to t in \( G_f \),

send as much flow along the path as possible

\[ G_f = \text{residualGraph}(G) \]

return flow
Ford-Fulkerson: is it correct?

When it terminates is it the maximum flow?

Assume it didn’t

- We know then that the flow < min-cut
- therefore, the flow < capacity across EVERY cut
- therefore, across each cut there must be a forward edge in $G_f$
- thus, there must exist a path from s to t in $G_f$
  - start at s (and $A = s$)
  - repeat until t is found
    - pick one node across the cut with a forward edge
    - add this to the path
    - add the node to A (for argument sake)
- However, the algorithm would not have terminated… a contradiction

Ford-Fulkerson: runtime?

- traverse the graph
- at most add 2 edges for original edge
  - $2V + E$

Can we simplify this expression?
Ford-Fulkerson: runtime?

Ford-Fulkerson\((G, s, t)\)
flow = 0 for all edges
\(G_f = \text{residualGraph}\(G)\)
while a simple path exists from s to t in \(G_f\)
send as much flow along path as possible
\(G_f = \text{residualGraph}\(G)\)
return flow

Can we bound the number of times the loop will execute?

74

75

Ford-Fulkerson: runtime?

Ford-Fulkerson\((G, s, t)\)
flow = 0 for all edges
\(G_f = \text{residualGraph}\(G)\)
while a simple path exists from s to t in \(G_f\)
send as much flow along path as possible
\(G_f = \text{residualGraph}\(G)\)
return flow

- traverse the graph
- at most add 2 edges
- integer capacities, so integer increases

BFS or DFS
\(O(V + E) = O(E)\)

76

77

Ford-Fulkerson: runtime?

Ford-Fulkerson\((G, s, t)\)
flow = 0 for all edges
\(G_f = \text{residualGraph}\(G)\)
while a simple path exists from s to t in \(G_f\)
send as much flow along path as possible
\(G_f = \text{residualGraph}\(G)\)
return flow

- max-flow
- increases ever iteration
- integer capacities, so integer increases

Overall runtime? \(O(\text{max-flow} \times E)\)
Can you construct a graph that could get this running time?

Hint:

O(max-flow * E)
Can you construct a graph that could get this running time?

What is the problem here? Could we do better?
Faster variants

Edmunds-Karp

- Select the shortest path (in number of edges) from s to t in G;
  - How can we do this?
    - use BFS for search
- Running time: O(V E^2)
  - avoids issues like the one we just saw
  - see the book for the proof

preflow-push (aka push-relabel) algorithms
- O(V^3)

Other variations...

Other variations...

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ford-Fulkerson</td>
<td>O(V^2 E)</td>
</tr>
<tr>
<td>Edmond-Karp</td>
<td>O(V^3)</td>
</tr>
<tr>
<td>Dinitz-Tarjan</td>
<td>O(V^3)</td>
</tr>
<tr>
<td>Dinic</td>
<td>O(V^2 E)</td>
</tr>
<tr>
<td>edmondskarp.pdf</td>
<td></td>
</tr>
</tbody>
</table>

http://akira.ruc.dk/~keld/research/algoritmeforlamodel_03/Artikler/08/Goldberg88.pdf

http://en.wikipedia.org/wiki/Maximum_flow