Tutorial for Algorithm’s Theory
Problem Set 5

January 17, 2013
Exercise 1: Maximum Flow Algorithms

Consider the following flow network:

a) Solve the maximum flow problem on the above network by using the Ford Fulkerson algorithm.

b) Solve the maximum flow problem on the above network by using the preflow-push algorithm. Use the variant of the algorithm that always applies a push or relabel operation to a maximum height node with positive excess.

c) Give a minimum capacity $s - t$ cut of the given network.
Solution 1a: Ford Fulkerson algorithm

Improve flow using an augmenting path as long as possible:

1: Initially, \( f(e) = 0 \) for all edges \( e \in E \), \( G_f = G \)
2: while there is an augmenting s-t path \( P \) in \( G_f \) do
3: Let \( P \) be an augmenting path in \( G_f \)
4: \( f' := \text{augment}(f, P) \)
5: update \( f \) to \( f' \)
6: update the residual graph \( G_f \)
7: end
Solution 1a: Ford Fulkerson algorithm

Step 1. Flow graph (on the left) and residual graph (on the right)
Step 2. Flow graph (on the left) and residual graph (on the right)
Solution 1a: Ford Fulkerson algorithm

Step 3. Flow graph (on the left) and residual graph (on the right)
Step 4. Flow graph (on the left) and residual graph (on the right)
Step 5. Flow graph (on the left) and residual graph (on the right)
Solution 1a: Ford Fulkerson algorithm

Step 6. Flow graph (on the left) and residual graph (on the right)
Solution 1a: Ford Fulkerson algorithm

Finally, the algorithm terminates because there are no more augmenting paths. Below is given the maximum flow we get from applying the Ford-Fulkerson algorithm on the given graph.
Solution 1b: Preflow-push algorithm

- **Initialization:** \( \text{height}(s) = 0 \) and \( h(v) = 0 \) for all other nodes
- Saturate all edges that go out of source node \( s \)
- Set the flow to 0 for all other nodes.
- Note that, given a node, along with its label (e.g., \( s, a, \ldots \)), the first number denotes the height and the second number the excess value.

![Network Diagram](image-url)
Choose node with maximum height. E.g., select node $a$ arbitrarily.

- Push operation is not possible, since $h(a) = h(v)$ for all $v \neq s$
- Therefore, relabel $a$, i.e., $h(a) := h(a) + 1 = 1$

Consider node $a$, since $h(a) = 1$.

- Push operation is possible, since $e = (a, t)$ has positive residual capacity
- Flow at this edge is increased by $\min\{e_f(a), c_e - f(e)\} = \{10, 5 - 0\} = 5$
Solution 1b: Preflow-push algorithm

Consider again node $a$.

- *Push* is possible, since $h(a) > h(c)$ and the edge $(a, c)$ has a positive residual capacity.

- Increase flow at edge $(a, c)$ by $\min\{e_f(a), c_e - f(e)\} = \{5, 3 - 0\} = 3$
Solution 1b: Preflow-push algorithm

Consider again node $a$ (all other nodes have height of 0 and the excess value of $a$ is still larger than 0).

- *Push* is not possible, since there are no more forward edges with a positive residual capacity.
- Hence, *relabel* $a$, i.e., $h(a) := h(a) + 1 = 2$
- Note the above steps will be repeated again and again until $h(a) = 7$

Consider again node $a$ (for the same reasons as above).

- *Push* is possible, because $s$ is a neighbor of $a$ such that $h(a) > h(s)$ and the residual capacity of edge $e = (a, s)$ is positive.
- Since we are now dealing with a *backward edge*, we must decrease the flow at this edge by $\min\{e_f(a), f(e)\} = \min\{2, 10\} = 2$
- Hence, the flow at edge $(s, a)$ equals 8.
Solution 1b: Preflow-push algorithm
Solution 1b: Preflow-push algorithm

Note that all remaining nodes with positive excess values have a height of 0. So, we arbitrarily pick node $b$.

- Push is not possible, since $h(b) = h(v)$ for all $v \neq s$
- Therefore, relabel $b$, i.e., $h(b) := h(b) + 1 = 1$

Now the algorithm will choose node $b$

- Push is possible, because $h(b) > h(d)$ and edge $e = (b, d)$ has positive residual capacity.
- Therefore, increase the flow at $e$ by $\min\{e_f(b), c_e - f(e)\} = \min\{8, 3 - 0\} = 3$
- The excess value of node $b$ will now become 5.

Choose node $b$ again

- Push is possible for similar reasons as above
- Increase flow at $e = (b, c)$ by $\min\{e_f(b), c_e - f(e)\} = \min\{5, 10\} = 5$
- The excess value of node $b$ will now become 0.
Solution 1b: Preflow-push algorithm
Solution 1b: Preflow-push algorithm

Between $c$, $d$ choose arbitrarily node $d$.

- *Push* is not possible, since $h(d) = h(v)$ for all $v \neq s$
- Therefore, **relabel** $d$, i.e., $h(d) := h(d) + 1 = 1$
- **Now** *Push* is possible, because $h(d) > h(t)$ and edge $e = (d, t)$ has positive residual capacity.
- Increase the flow at $e$ by $\min\{e_f(d), c_e - f(e)\} = \min\{5, 10\} = 5$
- The excess value of node $d$ will now become 0.
Select node $c$, as it is the only node with positive excess value.

- *Push* is not possible, since $h(c) = h(v)$ for all $v \neq s$
- Therefore, **relabel** $c$, i.e., $h(c) := h(c) + 1 = 1$
- **Now** *Push* is possible, because $h(c) > h(t)$ and edge $e = (c, t)$ has positive residual capacity.
- Increase the flow at $e$ by $\min\{e_f(c), c_e - f(e)\} = \min\{8, 8\} = 8$
- The excess value of node $c$ will now become 0.
Finally, the algorithm terminates because there are no nodes which have a positive excess value. The preflow turned into a maximum flow, the result of which is shown below.
Solution 1c: Minimum capacity $s - t$ cut

An $s - t$ cut is a partition $(A, B)$ of the vertex set such that $s \in A$ and $t \in B$. Note that “Ford-Fulkerson” also gives a min-cut algorithm

To get such minimum capacity cut we use the following steps

- Add source nodes to $A^*$. Hence $A^* = \{s\}$
- Extend $A^*$ by adding $a$ to it, because there exists the edge $(s, a)$ which has a positive residual capacity. Hence $A^* = \{s, a\}$
- Since now $A^*$ has only outgoing edges with $c_e - f(e) = 0$, it cannot be extended further. Solution found!!
- Finally, $A^* = \{s, a\}$ and $B^* = \{b, c, d, t\}$

Note that the capacity of a $s - t$ cut $(A, B)$ is defined as follows

$$c(A, B) = \sum_{e \text{ out of } A} c_e = 8 + 5 + 5 + 3 = 21$$
Exercise 2: Forward-Only Paths

Assume a simplified variant of the Ford-Fulkerson algorithm, such that when computing augmenting paths, it only considers forward edges of the residual graph and it does not consider backward edges at all.

The claim is that this algorithm always computes a solution that is within a constant factor of the optimal one. That is, there is an absolute constant $b > 1$ such that the forward-edge-only algorithm computes a flow of value at least $1/b$ times the value of an optimal flow.

If this claim is true, prove it. Otherwise show that the ratio of the maximum flow value and the flow computed by the forward-edge-only algorithm can be arbitrarily large.

Assume that the forward-edge-only implementation always takes an arbitrary (possibly worst-case) augmenting path of only forward edges as long as such an augmenting path exists. You can also assume that all edge capacities are positive integers.
Such simplified variant of the Ford-Fulkerson algorithm does not always compute a solution that is within a constant factor of the optimal one. Consider a construction like the following

Every node has an edge connecting it to every node either horizontally or vertically strictly adjacent to it (not diagonally).

Also, it has a source node $s$ connecting it to nodes $(i, 1)$ and a sink node connecting to nodes $(i, n)$ for all $i \in 1 \ldots n$.

All edge capacities are 1 and the maximum flow is $n$. 
Exercise 2: Forward-Only Paths

Assume if our first choice is a path like the one shown in red.

Now, we cannot choose any other augmenting path of forward-edges with positive residual capacity.

Hence, the resulting \( s - t \) flow has a value of 1. It is clear that the ratio \( \frac{n}{1} \) is not bounded by any absolute constant \( b \) such that \( b > 1 \).
We consider the Bipartite Matching Problem on a bipartite graph $G = (V, E)$. As usual, we assume that $V$ is partitioned into sets $X$ and $Y$, and each edge has one end in $X$ and the other in $Y$. If $M$ is a matching in $G$, we say that a node $y \in Y$ is covered by $M$ if $y$ is an end of one of the edges in $M$.

a) Consider the following problem. We are given $G$ and a matching $M$ in $G$. For a given number $k$, we want to decide if there is a matching $M'$ in $G$ so that

i) $M'$ has $k$ more edges than $M$ does, and
ii) every node $y \in Y$ that is covered by $M$ is also covered by $M'$.

Give a polynomial-time algorithm that takes an instance of Coverage Expansion and either returns a solution $M'$ or reports (correctly) that there is no solution.

b) Give an example of an instance of Coverage Expansion, specified by $G$, $M$, and $k$, so that the following situation happens.

*The instance has a solution; but in any solution $M'$, the edges of $M$ do not form a subset of the edges of $M'$.***
Solution 4a: Coverage Expansion

Idea: Transform “Coverage Expansion” into a “Maximum-Flow”

Given a bipartite graph $G = (V_1 \sqcup V_2, E)$, matching $M$ and an integer $k$

1. Add source and sink nodes, $s, t$ $\Rightarrow \mathcal{O}(1)$
2. Add directed edges from $s$ to each node in $V_1$ $\Rightarrow \mathcal{O}(|V_1|)$
3. Add directed edges from each node in $V_2$ to $t$ $\Rightarrow \mathcal{O}(|V_2|)$
4. Add edges from $V_1$ to $V_2$ $\Rightarrow \mathcal{O}(|E|)$
5. Set the capacity of each edge in matching $M$ to 0 $\Rightarrow \mathcal{O}(|M|)$
   Edges in $M$ cannot be taken in augmenting paths
6. Set the capacity of all the other edges to 1 $\Rightarrow \mathcal{O}(|E| - |M|)$
   Add capacities to the remaining edges to conclude the transformation

Therefore, the overall running time of this procedure is $\mathcal{O}(|E|)$
Algorithm: Coverage-Expansion-Solver($G, M, k$)
Input: bipartite graph $G$, matching $M$, and integer $k$
Output: a matching $M'$ if possible, otherwise false

1: if $k < 0$ then
2: return false $\implies \mathcal{O}(1)$
3: if $k == 0$ then
4: return $M$
5: $G' = \text{transform } G \text{ into a network flow } G'$ as specified above $\implies \mathcal{O}(|E|)$
6: $f = \text{Ford-Fulkerson}(G')$ $\implies \mathcal{O}(|E| \cdot n)$
7: if $|f| \geq |M| + k$ then
8: $M' = \text{pick } k \text{ disjoint edges, where each edge has the form } e = (v, u),$
   s.t., $v \in V_1, u \in V_2$ $\implies \mathcal{O}(|E|)$
9: $M' := M \cup M'$ $\implies \mathcal{O}(|M| + k) = \mathcal{O}(|E| + k) = \mathcal{O}(|E|)$
10: return $M'$
11: else
12: return false

Therefore, the overall running time of this algorithm is $\mathcal{O}(|E| \cdot n)$
Solution 4a: Coverage Expansion

\(G' = \text{transform } G \text{ into a network flow } G' \text{ as specified above}
\)
\(f = \text{Ford-Fulkerson}(G')\)

\[\begin{array}{l}
\text{if } |f| \geq |M| + k \text{ then} \\
\quad M' = \text{pick } k \text{ disjoint edges, where each edge has the form } e = (v, u), \text{ s.t., } v \in V_1, u \in V_2 \\
\quad M' := M \cup M' \\
\text{return } M'
\end{array}\]

We must show that \(|f| \geq |M| + k\) iff there are at least \(k\) disjoint edges between \(V_1\) and \(V_2\), which are also disjoint to the edges in \(M\).

\((\Rightarrow)\) Assume \(|f| \geq |M| + k\). Since all edges have capacity 1 (except those in \(M\)), then there must be at least \(|f|\) edge-disjoint paths from \(s\) to \(t\). This implies that there are at least \(k\) disjoint edges between \(V_1\) and \(V_2\).

\((\Leftarrow)\) Assume there are \(k\) disjoint edges between \(V_1\) and \(V_2\), which are also disjoint to the edges in \(M\). Now, we can take the edges from \(M\) and get a flow of \(|M| + k\) which satisfies the formula \(|f| \geq |M| + k\).
The instance has a solution; but in any solution $M'$, the edges of $M$ do not form a subset of the edges of $M'$.

Let the above be a bipartite graph whose nodes are partitioned in $A = \{v_1, v_2\}$ and $B = \{v_2, v_4\}$

Let $M = \{(v_3, v_2)\}$ be the matching indicated by the red line

Let $k = 1$ and let this be an instance of the coverage expansion problem

$|M| = 1 \implies |M'| = |M| + k = 2$

Let $M' = \{(v_1, v_2), (v_3, v_4)\}$

Note that $M'$ has more edges than $M$ and every node covered in $M$ is also covered in $M'$

Moreover, the edges in $M$ are not a subset of the edges in $M'$