CSCE 629 - 602 Analysis of Algorithms

October 26, 2016

Homework VII **Rishabh Yadav**
Prof. Anxiao (Andrew) Jiang UIN: 425009824

1A. 26.2-11 Solution

Idea

Given an un-directed graph $G(V, E)$, where V is the number of vertices and E is the number of edges in the graph. We can define a flow network $G_{uv}(V', E')$ which is nothing but a directed version of the given graph $G(V, E)$ with *source(s)* $= u$, and *sink* $= t$ with, $V' = V, E' = 2E$. All edge capacities set to 1, since we want the number of edges of G crossing a cut equals the capacity of the cut in G_{uv}.) Let f_{uv} denote maximum flow in G_{uv}.

Now, for each G_{uv} we will find the maximum flow f_{uv}. The edge connectivity, k of the Graph G would be minimum of these f_{uv}.

Pseudo code

def EDGE-CONNECTIVITY(G)

1. $k = \infty // k \rightarrow$ Edge Connectivity
2. select any vertex $u \epsilon V$
3. for each vertex $v \epsilon V - \{u\}$
   (a) make the flow network G_{uv}[Same vertices and twice the edges, and capacity $C(u, v) = 1$]
   (b) find the maximum flow f_{uv} on G_{uv}
   (c) $k = min(k, |f_{uv}|)$
4. return k

Proof of correctness

Our claim is that, edge connectivity, $k = min(|f_{uv}|), \forall v \epsilon V - \{u\}$

We prove this in 2 parts. One by showing $k \leq min(|f_{uv}|), \forall v \epsilon V - \{u\}$ and. Two $k \geq min(|f_{uv}|), \forall v \epsilon V - \{u\}$ Combining which we can conclude the desired result.

Case 1: $k \leq min(|f_{uv}|), \forall v \epsilon V - \{u\}$

Let $m = min(|f_{uv}|), \forall v \epsilon V - \{u\}$. Suppose we remove $m - 1$ edges from G. For any vertex, from the max-flow min-cut theorem, u and v are still connected. Since the max flow f_{uv} is at least m, hence any cut separating u from v has capacity at least m, which means at least m edges cross any such cut. Therefore one edge is crossing the cut when we remove $m-1$ edges. Thus every node is connected to u, which implies that the graph is still connected. So at least $m - 1$ edges must be removed to disconnect the graph. Hence $k \leq min(|f_{uv}|), \forall v \epsilon V - \{u\}$

Case 2: $k \geq min(|f_{uv}|), \forall v \epsilon V - \{u\}$

Consider a vertex with the minimum f_{uv}. From max-flow min-cut theorem, we can say that there is a cut of capacity f_{uv} separating u and v. Also as all edge capacities are 1, exactly f_{uv} edges cross this cut. If these edges are removed, there is no path from u to v, our graph gets disconnected.Hence $k \geq min(|f_{uv}|), \forall v \epsilon V - \{u\}$

Combining the above two we can say $k < min(|f_{uv}|), \forall v \epsilon V - \{u\}$.

Time Complexity

Time complexity to find the maximum flow network is $O(VE^2)$, where V is the number of vertices in the graph and E is the number of edges. And we do this for all vertices, Hence Time complexity is:

$O(V^2E^2)$

2A. 26.3-3 Solution

The graph given to us is a bipartite graph. So by definition G has no edges between vertices in L and no edges between vertices in R, by extension neither does the flow network G' and hence neither does the residual network, G'_f

Also, any augmenting path is a simple path of the form $s \rightarrow .. \rightarrow t$ in the residual network G'_f. By definition we can say that a simple path from $s$to t, involves s to L and R to t.

Now, any augmenting path when running the FORD-FULKERSON method, will be a simple path which will start with an edge $s \rightarrow L$ and will end with an edge $R \rightarrow t$. However, this can contain several edges which alternate between L and R between these edges. The path would thus be of the form:

$s \rightarrow L \rightarrow R \rightarrow ...... \rightarrow L \rightarrow R \rightarrow t$

going back and forth between L and R at most as many times as it can do so without using a vertex twice. The maximum number of edges which we can have is $2 * min(|L|, |R|) - 1$. Plus the edges $s \rightarrow L$ and $R \rightarrow t$. Hence the length of augmenting path during execution of FORD-FULKERSON method is thus bounded above by $2 * min(|L|, |R|) + 1$.

3A. 26-2 Solution

Idea

From the hint, we will construct a graph $G'(V', E')$ from G(V,E) in the following manner. From the DAG $G(V, E)$ where $V = (1, 2, ..., n)$, we construct a bipartite graph, $G'(V', E')$ such that $V' = (V \cup V^*)$, where $V^* = \{u^*, v^*.....\}$ for $V = \{u, v.....\}$ and edge $(u_i, v'_j) \epsilon\ E'$ if edge $(u_i, v_j) \epsilon\ E$.

Now we run the maximum flow algorithm on this graph G' using a dummy super-source and super-sink node to find the maximal matching. By retracing the edges in the maximal matching, we will obtain the minimum path cover.

Pseudo code

def MINIMUM-PATH-COVER(G)

1. Construct $G'(V', E')$ from the given graph $G(V, E)$
2. add a dummy super-source s and a super-sink t node in G'.
3. find the maximum flow, f_{st} in G', which gives us the maximum matching of G'. Let M be the edges in the maximal matching of G'.
4. $paths = \Phi$
5. While *paths* does not have every vertex ϵV:
   (a) get a vertex $u \epsilon V$ such that $u \notin paths$ and $u \notin M$.
   (b) $currentPath = [u]$
   (c) while u has an edge $(u, v') \epsilon\ M$
      i. append v to *currentPath*
      ii. $u = v$
      iii. append *currentPath* to *paths*
6. return *paths*

Proof of Correctness

There are $|V'| = n$ edges in the graph and $|M|$ matching. So we can say that we will have at most $n - |M|$ paths. Also we run the algorithm for finding the path for un-matched vertices ,i.e, the vertices which are not present in any matching M. Number of such vertices is $n - |M|$.If G has a path cover, $paths = P$ with k paths, we will show that G' will have a matching with $n - k$ edges. For every edge (u, v) which is in P, we can add the edge (u, v') to G', since every vertex has at most one outgoing edge and one incoming edge, u in G will have at most one out going edge and v' in G' at most one incoming edge. which is nothing but matching in G'. If the path cover has e edges, every vertex in the graph is either a starting point of a path or is pointed to by a unique edge, thus, $n = e + k \Rightarrow k = n - e$.

G' with a maximum matching of $|M|$, we find a path cover of $n - |M|$ for G. Say that the minimum path cover of G is $k < |M|$. From definition of G' and the argument above we can say that there are paths which combine to form longer paths i.e. there would be more edges in the minimum path cover. Since, more edges are possible, it would mean that we can have a matching of $(n - k) > |M|$ in G'. However, since M is the maximum matching, we have a contradiction. Therefore, it is not possible to have a minimum cover, less than $n - |M|$.

Time Complexity

Maximum Matching algorithm runs in $O(VE)$. To identify the paths is run at most on all the vertices, hence $O(V)$, therefore

$\textit{Time complexity} = O(VE)$

Part b

NO. The above proposed algorithm does not work for a cyclic graph. Because paths identified in the matching may actually be part of a cycle in the original graph, giving wrong results. Let us for example take the simplest cyclic graph with two vertex and two edges, forming a cycle. Running the algorithm to create a bipartite G' and thereafter finding the maximum flow, will return 2. If we use this result to find the minimum path cover, $n - |M| = 0$, with $n = 2, |M| = 2$ which is not the right solution.

References

Introduction to Algorithms by T. Cormen, C. Leiserson, R. Rivest, C. Stein