CMSC 351 - Introduction to Algorithms
Spring 2012
Lecture 18

Instructor: MohammadTaghi Hajiaghayi
Scribe: Rajesh Chitnis

1 Introduction

In this lecture we will look at DAGs and Topological Sorting.

2 Topological Sorting for DAGs

DAGs (directed acyclic graphs) are directed graphs with no directed cycles. Suppose there is a set of tasks that need to be performed one at a time. Some tasks depends on other tasks and they cannot be started until the other tasks are completed. All the dependencies are known and we want to arrange a schedule for performing the tasks which is consistent with the dependencies. We can associate a directed graph whose vertices are the tasks and whose edges are the dependencies. The graph must be acyclic otherwise the tasks can never be completed. This problem is known as topological sorting. More formally, the problem is as follows: Given a DAG with n vertices, label the vertices from 1 to n such that if v is labeled k, then all vertices that can be reached from v by a directed path are labeled with labels $> k$. First we have a small observation: A DAG always contains a vertex of indegree 0. The proof is similar to the one which shows that every tree has a leaf. Otherwise we can traverse the graph forever which contradicts the fact that we do not have a cycle. Note that the same argument gives the existence of a vertex of outdegree 0 as well. Thus the algorithm is as follows: First we find a vertex of indegree 0. Once we find it, we label it with 1 and remove its adjacent edges and label the rest of the graph (which is still acyclic) by induction.

Time Complexity: Initializing indegrees with DFS takes $O(|V|+|E|)$ time. Finding a vertex of indegree 0 takes constant time using a queue. We consider each edge (v, w) exactly once we bring v out of the queue. Thus we update the Indegree variable at most $|E|$ times and hence the total running time is $O(|V| + |E|)$.

Algorithm 1 TOPOLOGICAL-SORTING

Input: A DAG G.

Output: A topological sort of G.

```
1: Initialize v.Indegree for all vertices by DFS;
2: G-label:=0;
3: for $i = 1$ to $n$ do
4:   if $v_i$.Indegree=0 then
5:     Put $v_i$ in the queue;
6:     while Queue is not empty do
7:       Remove a vertex $v$ from the queue;
8:       G-label:=G-label+1; v.label:=G-label;
9:       for all edges (v,w) do
10:        w.Indegree:=w.Indegree-1;
11:        if w.Indegree =0 then
12:          Put w in the queue.
```

References

[1] Udi Manber, *Introduction to Algorithms - A Creative Approach*