Lecture 21 Longest Path in DAG, Longest Inc Subseq

Lecturer: Shayan Oveis Gharan Scribe:

1 Longest Path in a DAG

Given a DAG G consider the topological order where the vertices are labelled $1, 2, \dots, n$ such that for any directed edge $i \to j$ we have $i < j$.

Now for $1 \leq j \leq n$, define $OPT(j)$:=length (i.e., the number of edges) of the longest path ending at j.

Base Case: For any vertex j with $indeg(j) = 0$ we have $OPT(j) = 0$ Because no path ends at j. So, $OPT(1) = 0$ as well.

IH: Suppose we have computed $OPT(i)$ for all $i < j$ for some $j \geq 2$.

IS: We want tot find $OPT(j)$. We guess that the last node prior to j in the longest path ending at j is i. Then, we have

- $i \to j$ must be an incoming edge of j.
- $i < j$ by the topological sorting
- Since $i < j$, by IH, $OPT(i)$ is already computed.

So, the longest path ending at j must be the longest path ending at i together with the edge $i \to j$. This means that

$$OPT(j) = OPT(i) + 1.$$

Now, we need to consider all possibilities for the guessed vertex i. All we need to do is to check over all in-comming edges of j and take the one with the largest OPT.

$$OPT(j) = \max_{i:i \to j} OPT(i) + 1.$$

This completes the proof of induction. The algorithm simply runs in time $O(|V| + |E|$ assuming that for every vertex we have stored all of its incoming edges in an adjacency list; note that the time process j is simply the indegree of j.

Once we compute $OPT(j)$ for all j we can simply output, $\max_{1 \leq j \leq n} OPT(j)$; this is because the longest path in G must end at one of the vertices $1, 2, \dots, n$.

2 Longest Increasing subseqence

Say $x_1, \dots, x_n$ is the input sequence. Define $OPT(j)$ = the length (number of integers) of the longest increasing subsequence that ends at j.

Base Case: Obviously $OPT(1) = 1$; similarly for any j where $x_j < x_i$ for all $i < j$ we have $OPT(j) = 1$¿

IH: Suppose we have computed $OPT(i)$ for all $i < j$ for some $j \geq 2$.

IS: We need to find $OPT(j)$. Similar to the previous problem, we guess i is the number right before j in the longest increasing subsequence that ends at j. Then, we must have

- $i < j$ and $x_i < x_j$ by definition of increasing subsequence.
- Since $i < j$, $OPT(i)$ is already computed by IH
- The longest increasing subsequence ending at j is simply the one ending at i together with the number j.

So, we get $OPT(j) = OPT(i) + 1$¿ Now, considering all possibilities for i we get

$$OPT(j) = \max_{i:i<j, x_i<x_j} 1 + OPT(i).$$

This completes the proof.

The algorithm we just explained runs in $O(n^2)$. Because it takes $O(j)$ operations to find $OPT(j)$; taking the sum $1+\cdots+n$, it runs in $O(n^2)$. The final output of the algorithm is $\max_{1\le j\le n} OPT(j)$.

Lastly, this problem can be solved by reducing it to the Longest path in a DAG problem; all we need to do is to construct a DAG from the given sequence of numbers. We put a vertex i for the number x_i. We add the directed edge $i \to j$ iff $i < j$ and $x_i < x_j$. Now, it can be seen that any increasing subsequence with l numbers correspond to a path in the DAG with $l - 1$ edges. Similarly, any path in the DAG with l edges correspond to an increasing subsequence with $l + 1$ numbers. So, we can just find the longest path in this DAG return the output plus one.