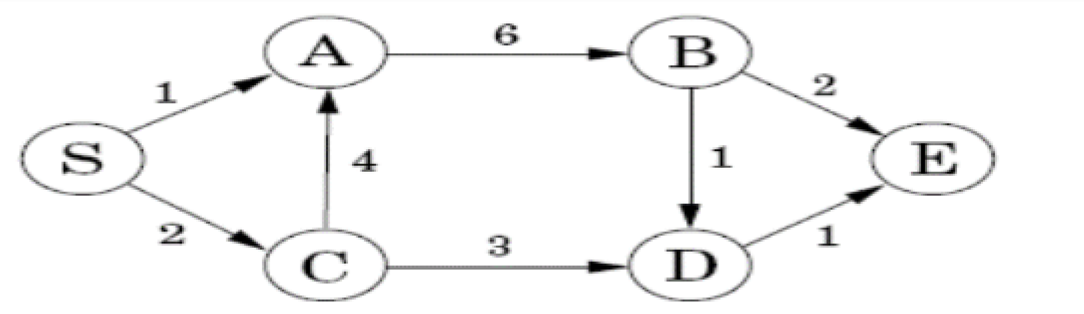
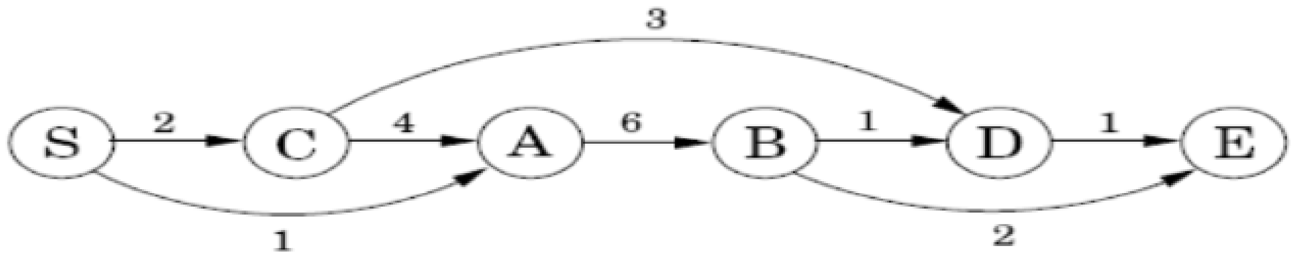
WEEK 5, LECTURE 9 ON 18 SEPTEMBER 2021 CS1.301.M21 ALGORITHM ANALYSIS AND DESIGN

DYNAMIC PROGRAMMING

SHORTEST PATH IN DAGS



Every DAG will have a topological sorting order as follows:



It can be shown that every DAG has a topological sorting order since a DAG always has a source. If the source is removed then the remaining nodes form a DAG themselves with another different source node, and this continues inductively.

Problem: For all non-source nodes calculate the shortest distance from the source node to that node.

Solution

```
intialize dist(all nodes) to \inf
dist(s) = 0
for all nodes v in V-{s}:       //non-source nodes
    dist(v) = minimum of {dist(u) + l(u,v)} for all u's that are in-neighbors of v
```

LONGEST PATHS IN DAGS

Instead of book-keeping the minimum of the values `{dist(u) + l(u,v)}`

But to find the longest paths in non-DAGs this isn't possible. The possibility of a cycle means that the answer for a longest path would be meaningless.

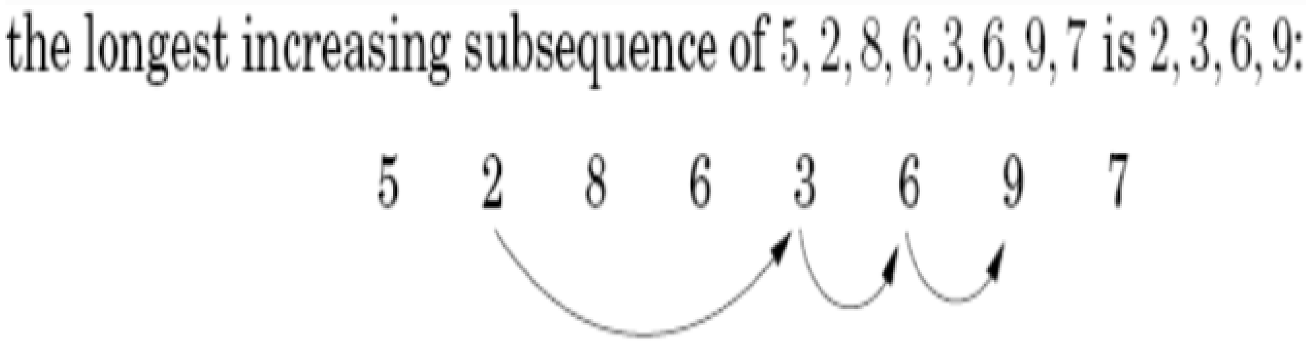
LONGEST INCREASING SUBSEQUENCE

Problem

The input is a sequence of of numbers $a_1, a_2, \ldots, a_n$

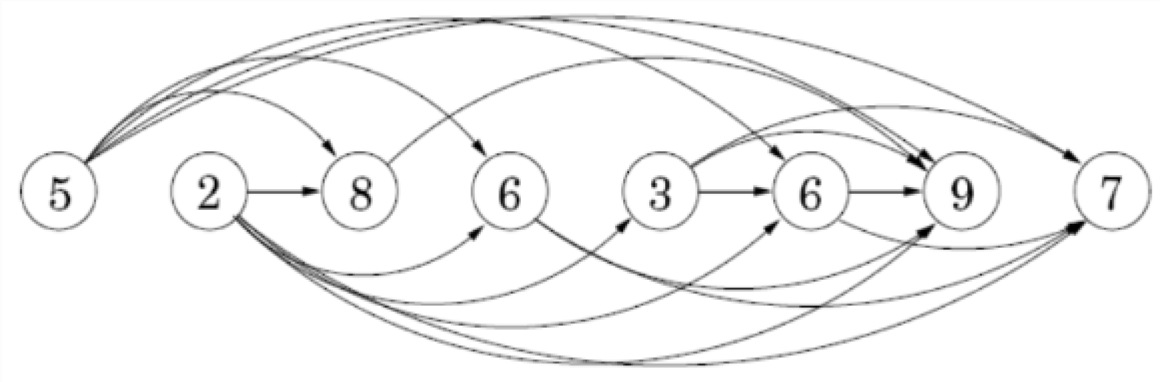
A subsequence is $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ such that $1 \leq i_1 \leq i_2 \leq \ldots \leq i_k$. An increasing subsequence is one where the a values are increasing.

For example,



Answer

Create a DAG of all permissible transitions. If a node i exists for each a_i then an edge (i,j) exists if $i < j$ and $a_i < a_j$ i.e. if a_i can come before a_j in an increasing subsequence.



$L(j)$ is the length of the longest path ending at a node j. In other word, it is the LIS ending at j.

Therefore the LIS of the given sequence is the maximal value of $L(j) + 1$ for all j

> In general, the dynamic programming paradigm is used when there are subproblems that depend on the answers of other subproblems.

All in all the algorithm is:

```
for all j:
    L(j) = 1+ max{L(i):fot all (i,j)}
return max L(j)
```