Pre-Announcements

Jessica from Del-Sigma-Pi (business fraternityish) has an event to announce about machine learning.

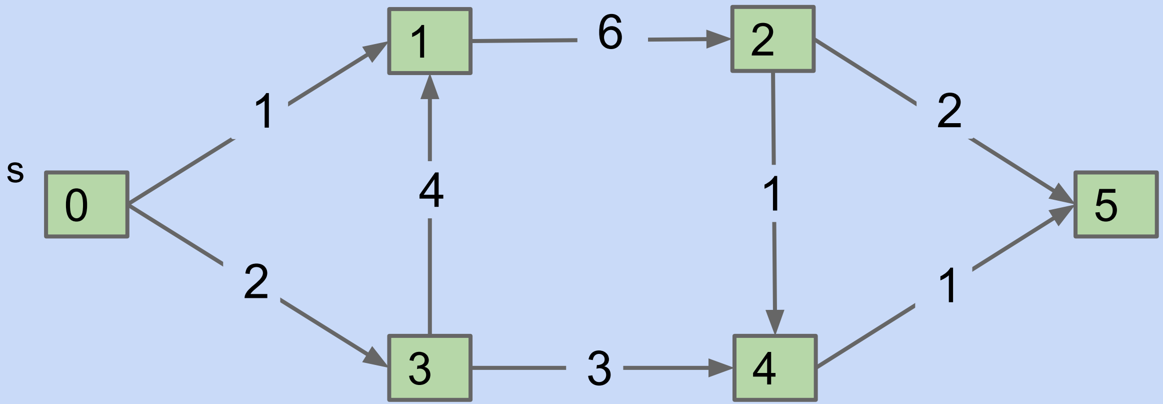
- Speakers from Amazon, Google, Facebook, etc.
- Free food after from food places.
- Raffle prizes of various items.
- Monday 6:30 PM in Anderson Auditorium in HAAS
- “Machine Learning: Workforce of Tomorrow)

Announcements

Project 3 coming imminently (tonight or tomorrow morning)

Graph Warmup Problem 1

What is the shortest paths tree for the graph below, using s as the source?
In what order will Dijkstra's algorithm visit the vertices?

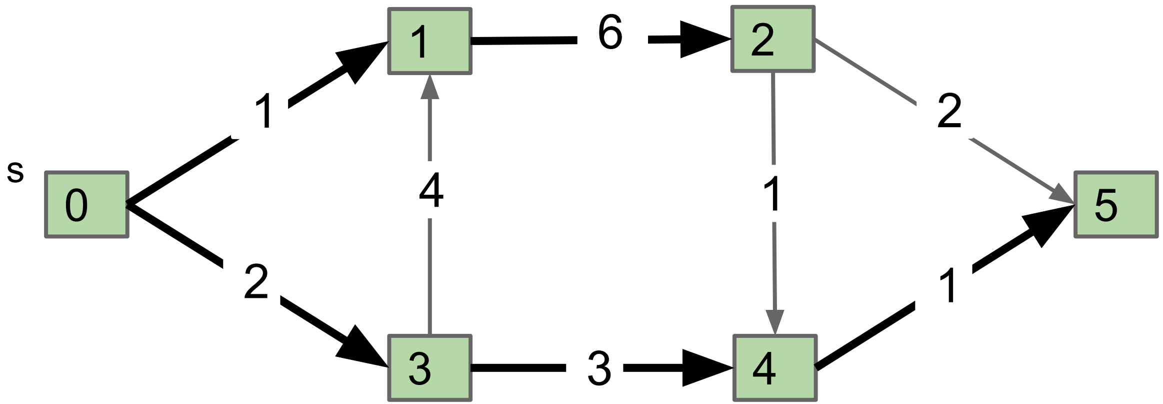


Graph from Algorithms, by Vazirani/Papadimitriou

Graph Warmup Problem 1

What is the shortest paths tree for the graph below, using s as the source?
In what order will Dijkstra's algorithm visit the vertices?

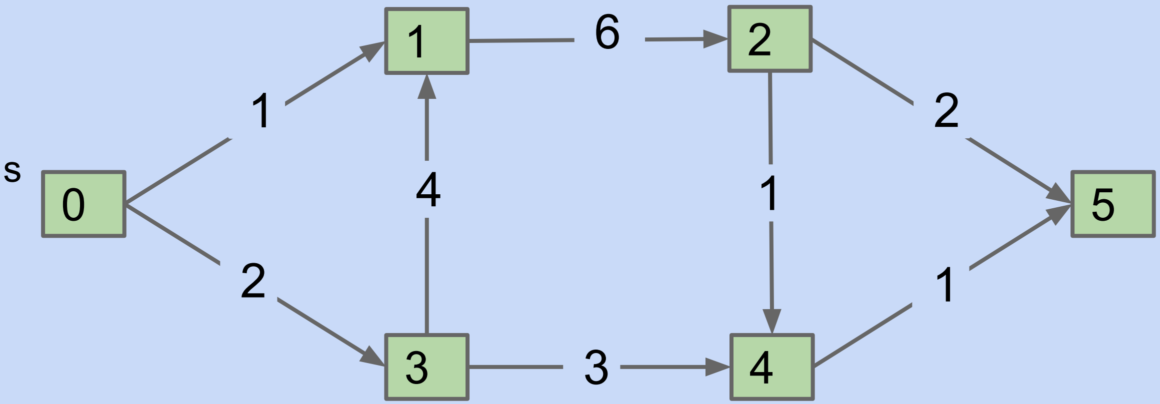
- 0, 1, 3, 4, 5, 2



Graph Warmup Problem 2

Give a topological ordering for the graph below (a.k.a. topological sort).

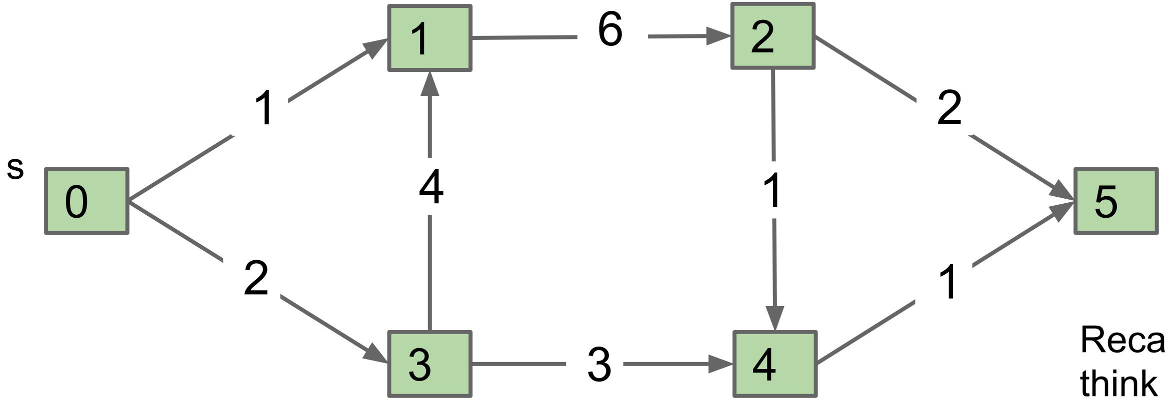
- (If you want the same answer as me, use the algorithm from lecture/section)



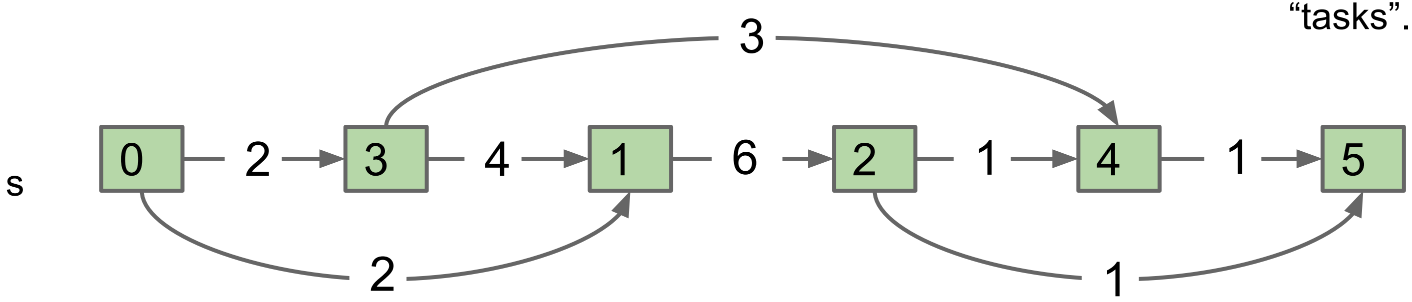
Graph Warmup Problem 2

Give a topological ordering for the graph below (a.k.a. topological sort)

- 0, 3, 1, 2, 4, 5



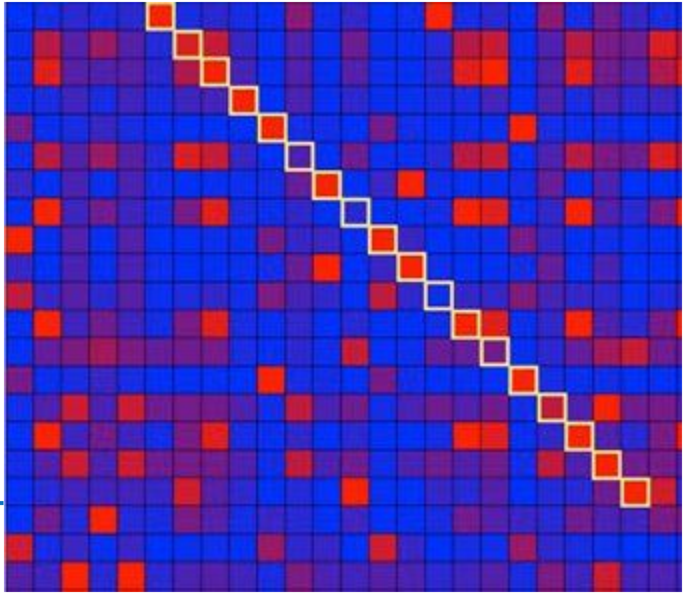
Recall that we can think of topological sort as an ordering of "tasks".



CS61B

Lecture 31: Dynamic Programming

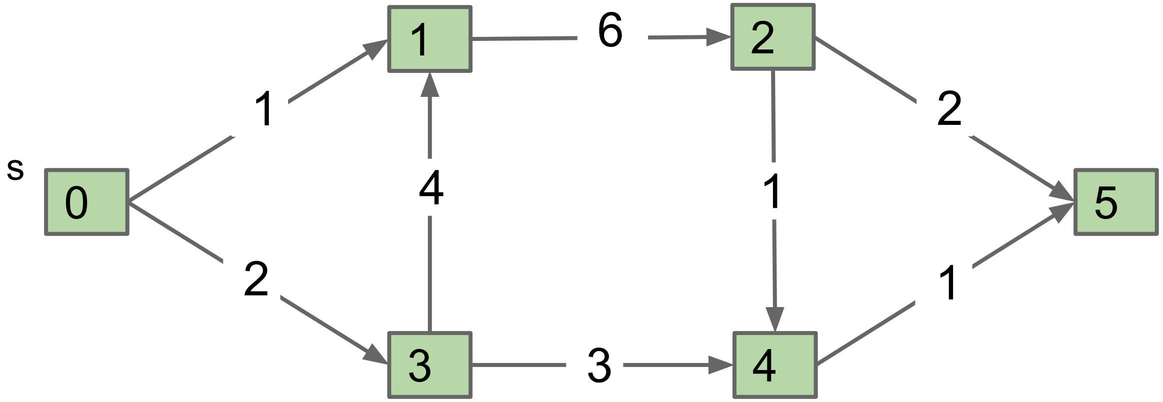
- Shortest Paths in a DAG
- Dynamic Programming
- Longest Increasing Subsequence (LIS)
- LIS Using Dynamic Programming

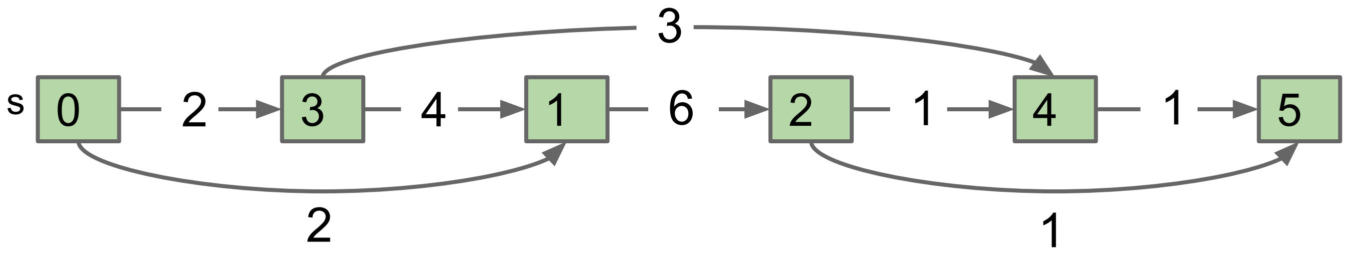


Directed Acyclic Graphs

One special type of graph is the directed acyclic graph (DAG).

- Any such graph can be topological sorted (sometimes called linearization).

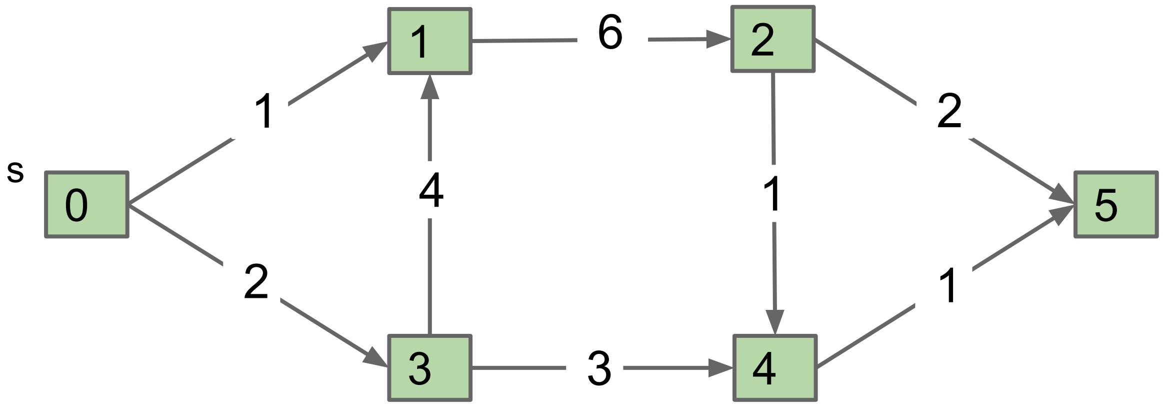




SPTs on Directed Acyclic Graphs

Can use Dijkstra's algorithm to find the SPT (we did this in warm-up problem).

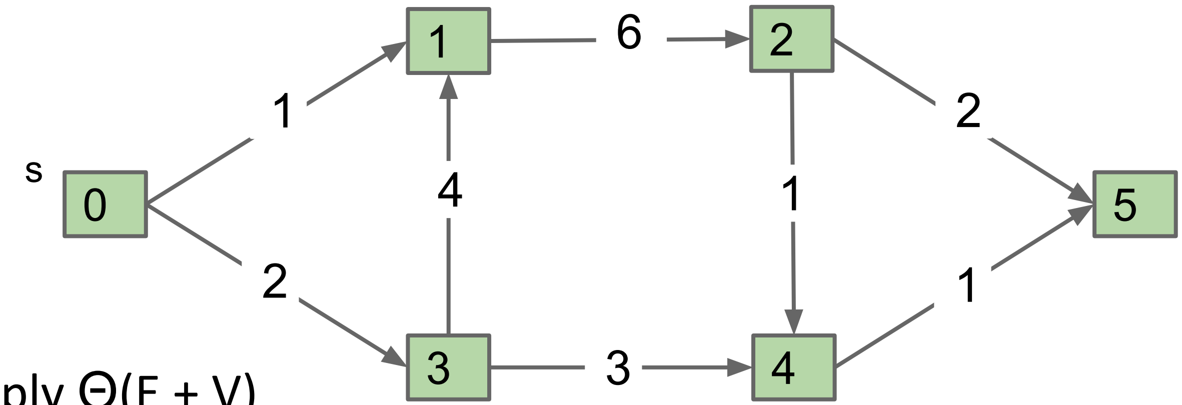
- Simpler approach: Just visit vertices in topological order, relaxing all edges from a vertex when it is visited. DAGSPT Algorithm: Demo



SPTs on Directed Acyclic Graphs

Can use Dijkstra's algorithm to find the SPT (we did this in warm-up problem).

- Simpler approach: Just visit vertices in topological order, relaxing all edges from a vertex when it is visited. DAGSPT Algorithm: [Demo]



Runtime is simply $\Theta(E + V)$

- Step 1: Have to do a topological sort, which is $\Theta(E + V)$ time.
- Step 2: Initialize some arrays of size V (in total, takes $\Theta(V)$ time).
- Step 3: Each edge gets relaxed exactly once (each taking constant time), so $\Theta(E)$

Dynamic Programming

The DAG SPT Algorithm

The DAG SPT algorithm can be thought of as solving increasingly large subproblems:

- Distance from source to source is very easy, and is just zero.
- We then tackle distances to vertices that are a bit farther to the right.
- We repeat this until we get all the way to the end of the graph.

Problems grow larger and larger.

- By “large” we informally mean depending on more and more of the earlier subproblems.

This approach of solving increasingly large subproblems is sometimes called **dynamic programming**.

Dynamic Programming

Dynamic programming is a terrible name for a simple and powerful idea for solving "big problems":

- Identify a collection of subproblems.
- Solve subproblems one by one, working from smallest to largest.
- Use the answers to the smaller problems to help solve the larger ones.

Identification of the "right" subproblems is often quite tricky.

- Largely beyond scope of CS61B.
- You'll study this in much more detail in CS170.

Why is the name so bad? It was by design! (Link)

Longest Increasing Subsequence

Example: Longest Increasing Subsequence

Given a sequence of numbers, find the longest increasing subsequence (LIS).

Example:

- Given the sequence 6, 2, 8, 4, 5, 7.
- The LIS is 2, 4, 5, 7.

Related problem: Find the length of the longest increasing subsequence (LLIS).

- For the problem above, the LLIS is 4.

Test Your Understanding: LIS / LLIS, yellkey.com/miss

Given the sequence 5, 2, 8, 6, 3, 6, 9, 7, what is the longest increasing subsequence and the LLIS?

A. LIS: 5, 2, 8, 6, 3, 6, 9, 7 LLIS: 8
B. LIS: 3, 6, 9 LLIS: 3
C. LIS: 2, 3, 6, 9 LLIS: 4
D. LIS: 2, 3, 5, 6, 6, 7, 8, 9 LLIS: 8
E. Something else

Test Your Understanding: LIS / LLIS, yellkey.com/miss

Given the sequence 5, 2, 8, 6, 3, 6, 9, 7, what is the longest increasing subsequence and the LLIS?

A. LIS: 5, 2, 8, 6, 3, 6, 9, 7 LLIS: 8
B. LIS: 3, 6, 9 LLIS: 3
C. LIS: 2, 3, 6, 9 LLIS: 4
D. LIS: 2, 3, 5, 6, 6, 7, 8, 9 LLIS: 8
E. Something else

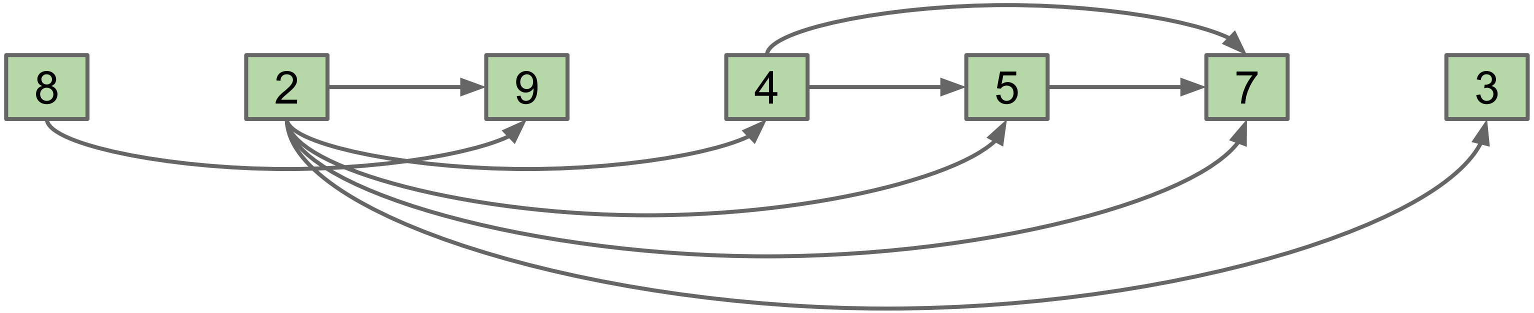
The Longest Increasing Subsequence and DAG Connection

Given a sequence of numbers, find the longest increasing subsequence (LIS).

Example:

- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The LIS is 2, 4, 5, 7.
- The LLIS is 4.

Observation: Can conceptualize this sequence as a DAG.



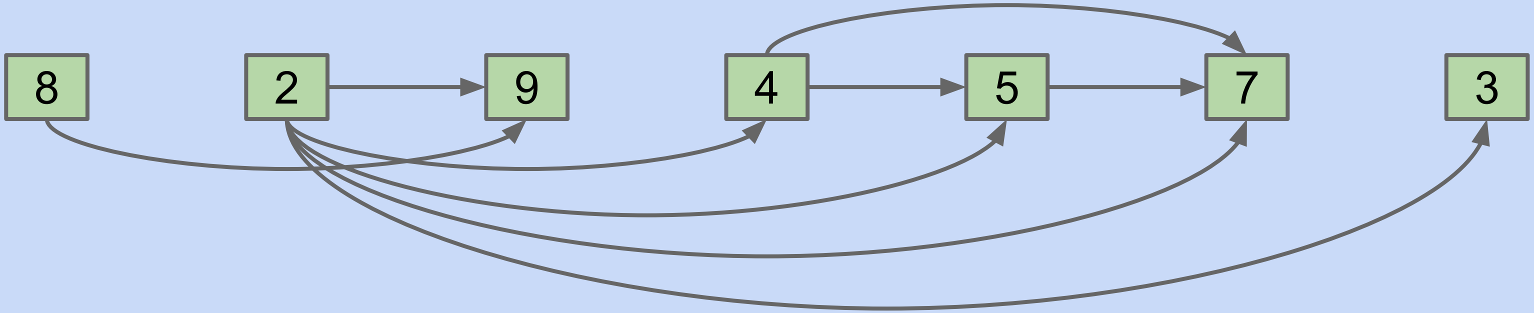
The Longest Increasing Subsequence and DAG Connection

Goal: Describe the **LLIS** problem in terms of a DAG problem.

- In other words, what property of the DAG are we looking for? Is it shortest paths? Something else?

Example:

- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The LIS is 2, 4, 5, 7.
- The LLIS is 4.



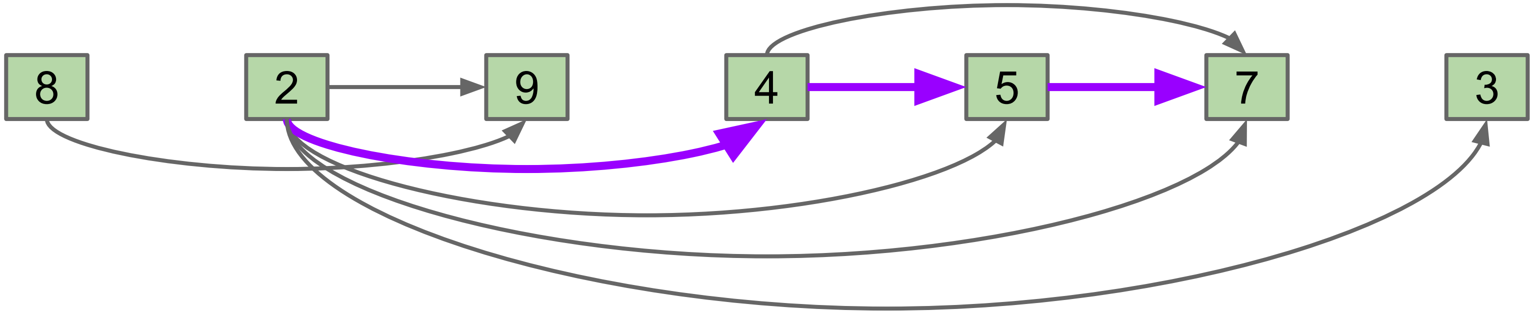
The Longest Increasing Subsequence and DAG Connection

Goal: Describe the LLIS problem in terms of a DAG problem.

- What is the length of the longest path from any vertex to any vertex in the entire DAG? (well… plus one)

Example:

- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The length of the longest path in the DAG is 3, and therefore the LLIS is 4.

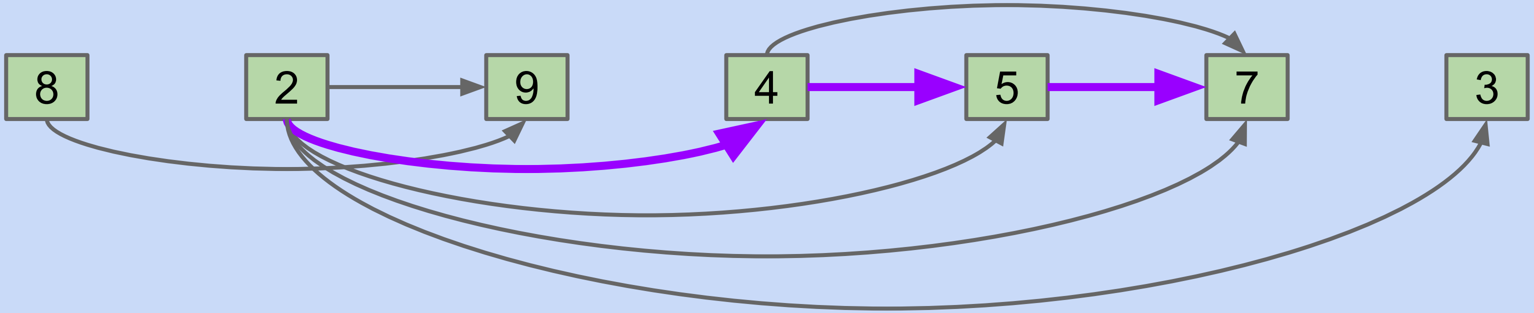


The Longest Increasing Subsequence and DAG Connection

Very Challenging Problem: Design an algorithm for finding the length of the longest path in a DAG.

Example:

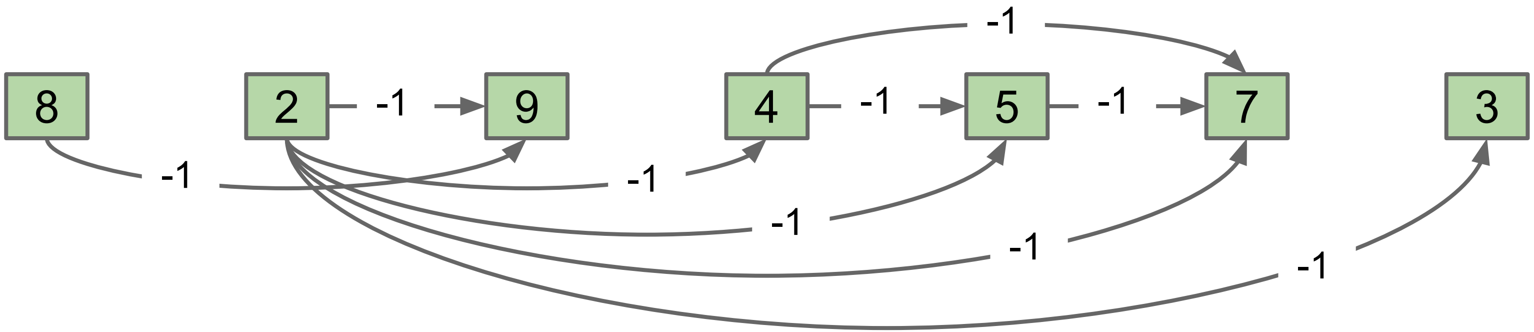
- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The length of the longest path in the DAG is 3, and therefore the LLIS is 4.



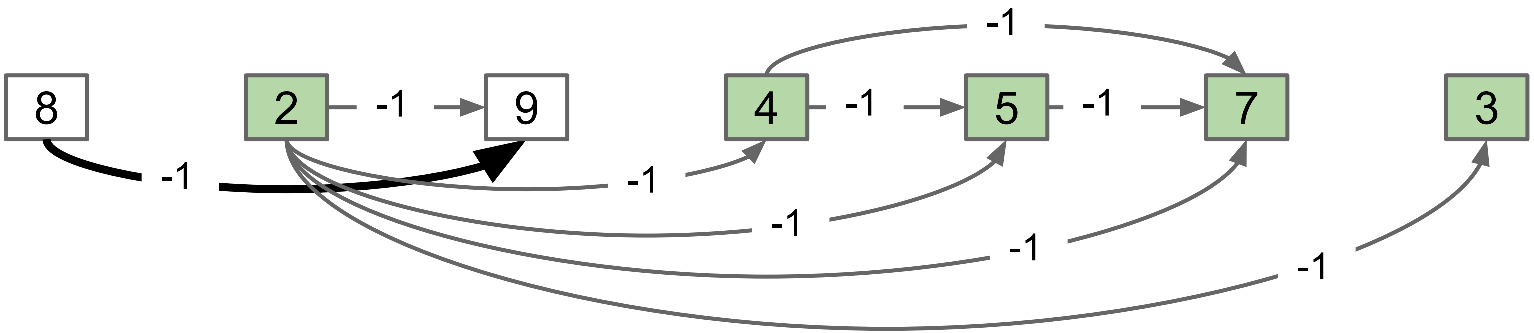
Using a Negative Weight Graph to Find Longest Paths

How do we find the length of the longest path from any vertex in the DAG?

- Create copy with edge weights set to -1.
- For each vertex v:
    - Run DAGSPT algorithm from v. ← Recall, this was a dynamic programming algorithm.
    - Let LPLS[v] = abs(min(distTo)), i.e. length of the longest path starting at v.
- Return max(LPLS), i.e. longest of the longest paths lengths.



DAGSPT With s=8



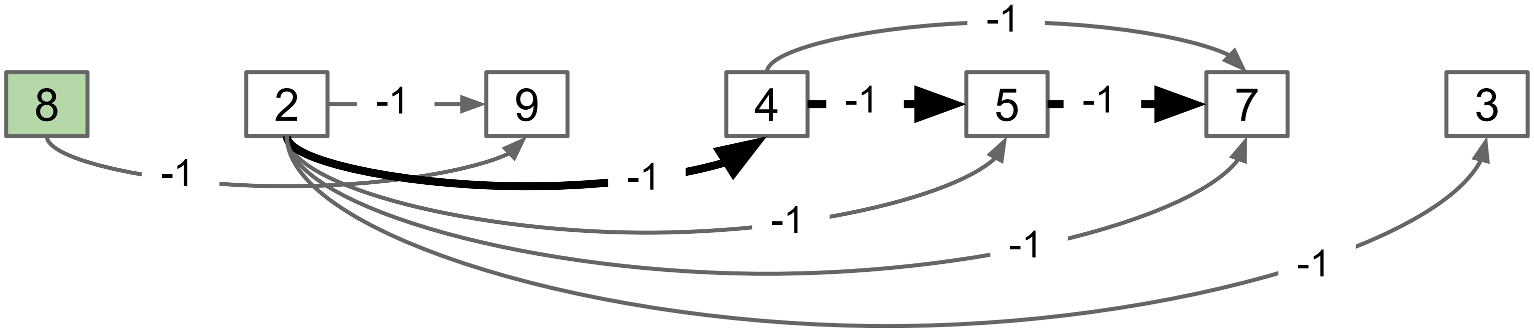
#	distTo	edgeTo
8	0	-
2	∞	-
9	**-1**	**8**
4	∞	-
5	∞	-
7	∞	-
3	∞	-

abs(min(distTo)) is 1.

- Longest path from 8 is of length 1.

LPLS: [1]

DAGSPT With s=2



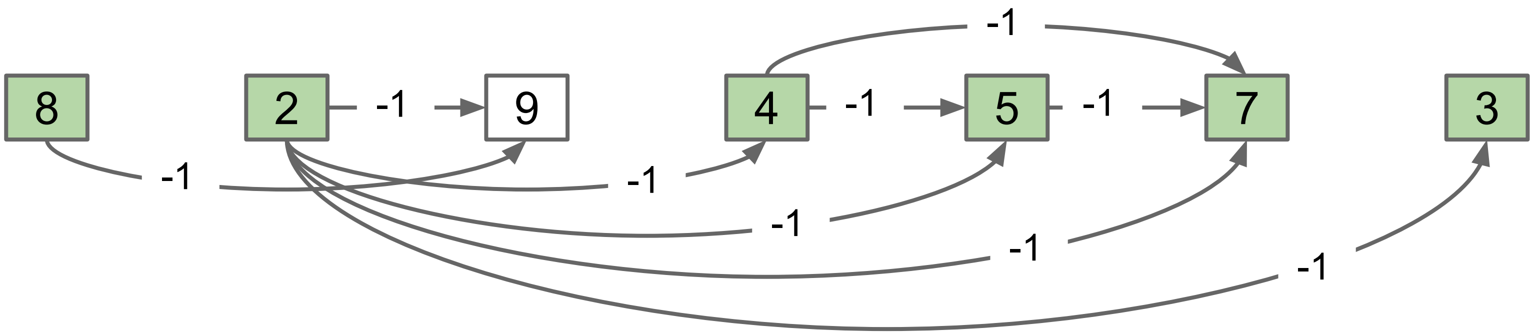
#	distTo	edgeTo
8	∞	-
2	0	-
9	-1	2
4	-1	2
5	-2	4
7	**-3**	**5**
3	-1	1

abs(min(distTo)) is 3.

- Longest path from 2 is of length 3.

LPLS: [1, 3]

DAGSPT With s=9



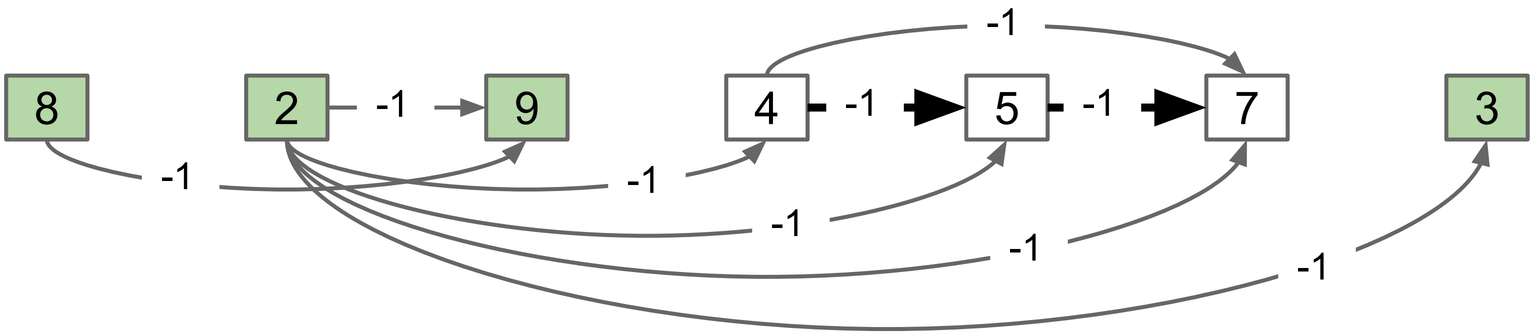
#	distTo	edgeTo
8	∞	-
2	∞	-
9	0	-
4	∞	-
5	∞	-
7	∞	-
3	∞	-

abs(min(distTo)) is 0.

- Longest path from 9 is of length 0.

LPLS: [1, 3, 0]

DAGSPT With s=4



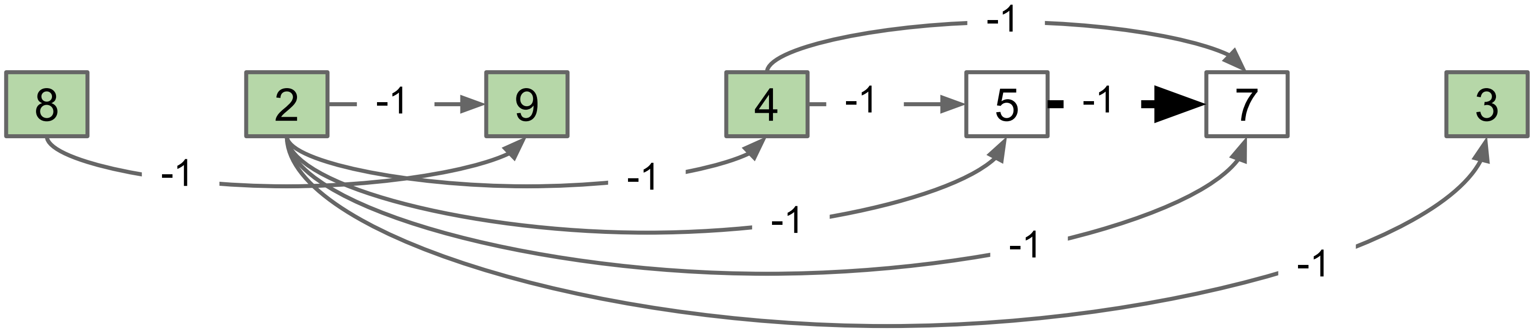
#	distTo	edgeTo
8	∞	-
2	∞	-
9	∞	-
4	0	-
5	-1	4
7	**-2**	**5**
3	∞	-

abs(min(distTo)) is 2.

- Longest path from 4 is of length 2.

LPLS: [1, 3, 0, 2]

DAGSPT With s=5



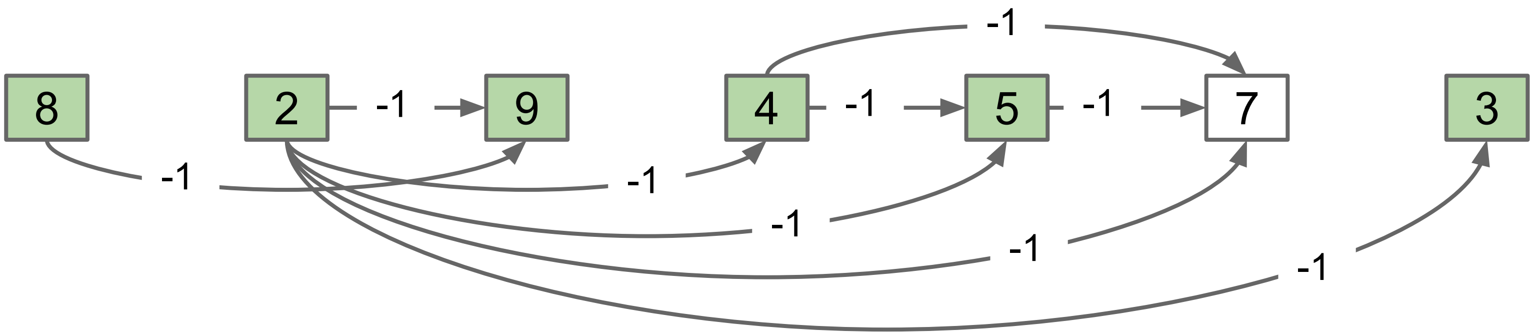
#	distTo	edgeTo
8	∞	-
2	∞	-
9	∞	-
4	∞	-
5	0	-
7	**-1**	**5**
3	∞	-

abs(min(distTo)) is 1.

- Longest path from 5 is of length 1.

LPLS: [1, 3, 0, 2, 1]

DAGSPT With s=7



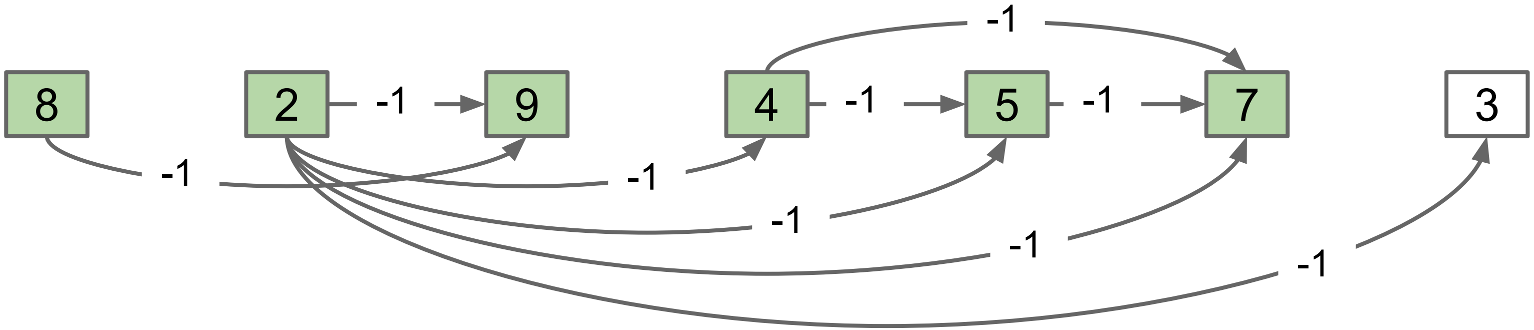
#	distTo	edgeTo
8	∞	-
2	∞	-
9	∞	-
4	∞	-
5	∞	-
7	**0**	**-**
3	∞	-

abs(min(distTo)) is 0.

- Longest path from 7 is of length 0.

LPLS: [1, 3, 0, 2, 1, 0]

DAGSPT With s=7



#	distTo	edgeTo
8	∞	-
2	∞	-
9	∞	-
4	∞	-
5	∞	-
7	∞	-
3	0	-

abs(min(distTo)) is 0.

- Longest path from 3 is of length 0.

LPLS: [1, 3, 0, 2, 1, 0, 0]

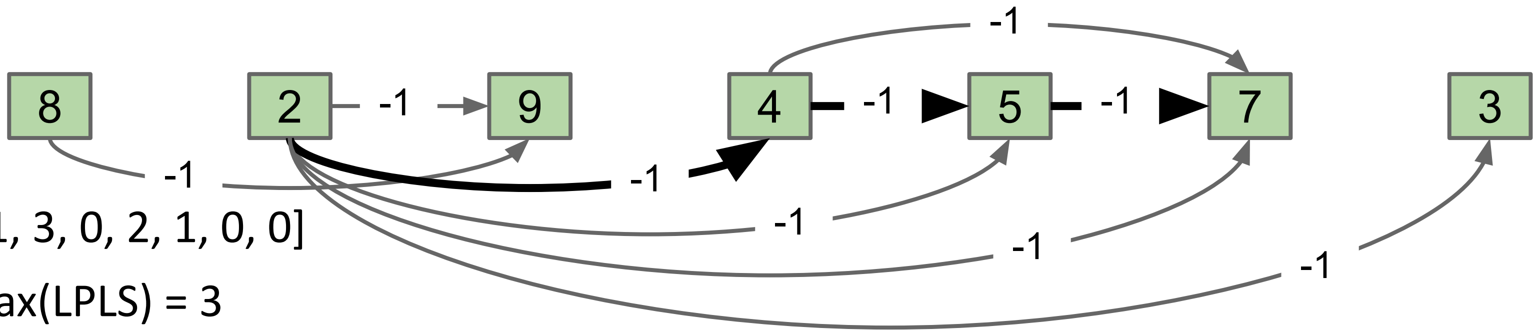
Using a Negative Weight Graph to Find Longest Paths

How do we find the length of the longest path from any vertex in the DAG?

- Create copy with edge weights set to -1.
- For each vertex v:
  - Run DAGSPT algorithm from v. ← Recall, this was a dynamic programming algorithm.
  - Let LPLS[v] = abs(min(distTo)), i.e. length of the longest path starting at v.
- Return max(LPLS), i.e. longest of the longest paths lengths.

See extra slides!

Solution to original LLIS problem is max(LPLS) + 1 = 4. Runtime is $O(N^3)$.



LPLS: [1, 3, 0, 2, 1, 0, 0]

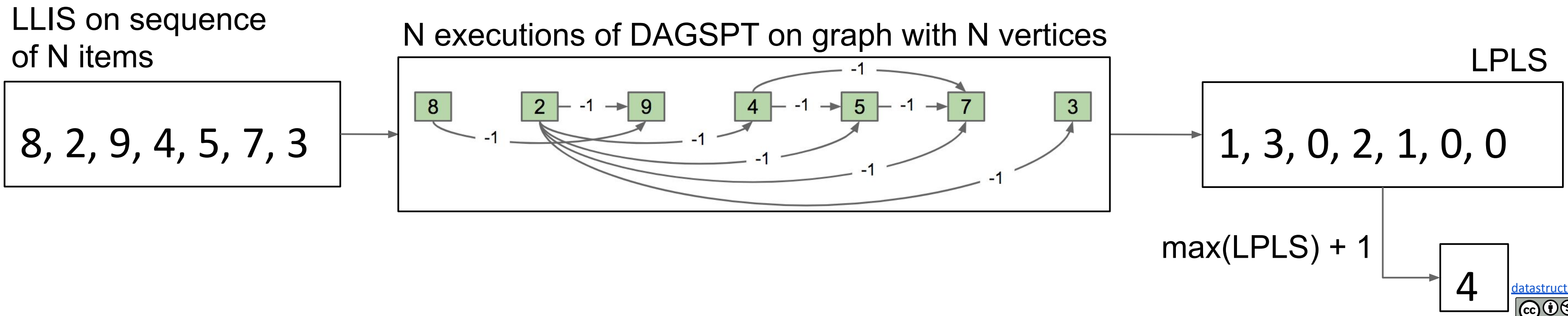
- max(LPLS) = 3

LLIS Summary

Given a sequence of integers, we transformed our problem into a problem on DAGs.

- This process is sometimes called **reduction**.
- We “reduced” LLIS to N solutions of the longest paths problem on DAGs.

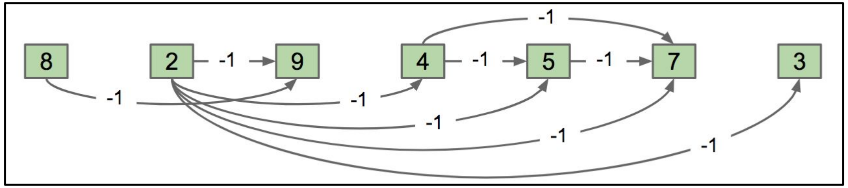
Solutions to longest path problems were found by executing a dynamic programming algorithm called DAGSPT.



LIS Using Dynamic Programming

Optimizing our LLIS Solution

LLIS problem: Given a sequence of numbers, find the length of the longest increasing subsequence.



Example:

- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The LIS is 2, 4, 5, 7, so the LLIS is 4.

The process we described felt redundant.

- Example: DAGSPT for s=4 is a subproblem of DAGSPT for s=2, but we ran both in their entirety.

Many possible ways to optimize, but we'll use dynamic programming today.

Example of an LLIS Subproblem

Longest Increasing Subsequence Problem

- Given the sequence 8, 2, 9, 4, 5, 7, 3.
- The LIS is 2, 4, 5, 7, so the LLIS is 4.

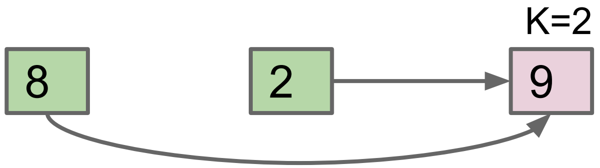
Dynamic programming:

- Identify a collection of subproblems.
- Solve subproblems one by one, working from smallest to largest.
- Use the answers to the smaller problems to help solve the larger ones.

Finding the “right” subproblems is often very hard.

Example subproblem for LLIS: Length of the Longest Subsequence Ending At (LLSEA)

- What is the LLIS **ending** at index #2?
  - 2 (either 8, 9 or 2, 9)



Examples of the LLSEA Problem

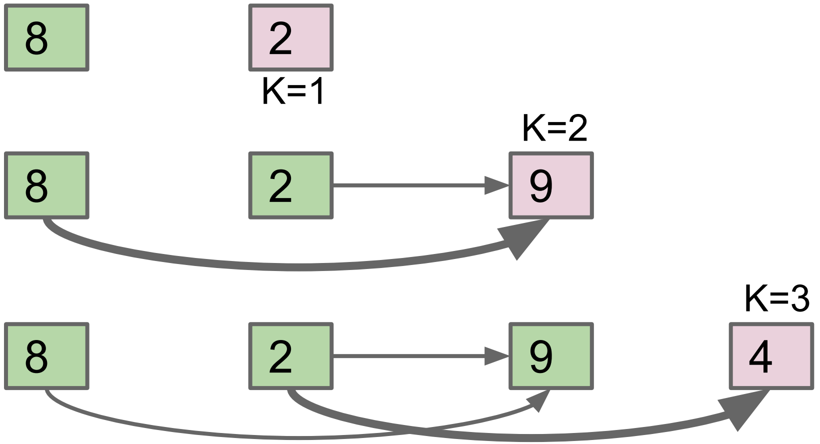
LLSEA Problem:

- Given the sequence 8, 2, **9**, 4, 5, 7, 3, the LLSEA(2) = 2.

Note: This 2 refers to item #2 in the sequence, which has value 9.

Definition: Let Q(K) be the LLIS **ending** at index K, a.k.a. the LLSEA(K). Examples:

- Q(1) = 1
- Q(2) = 2
- Q(3) = 2



Q values are solutions to the LLSEA problem.

Example of the LLSEA Problem, yellkey.com/miss

LLSEA Problem:

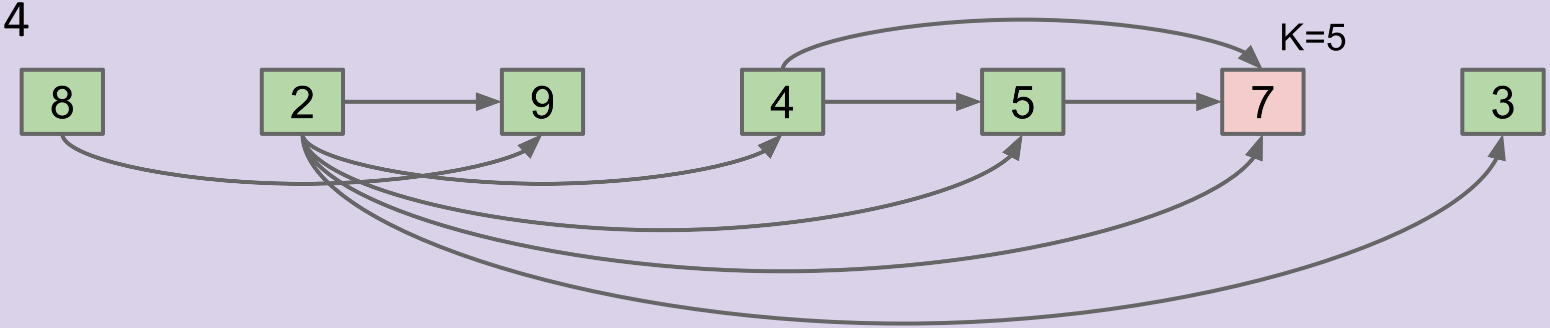
- Given the sequence 8, 2, **9**, 4, 5, 7, 3, the LLSEA(2) = 2.

Note: This 2 refers to item #2 in the sequence, which has value 9.

What is Q(5)?

A. 0
B. 1
C. 2
D. 3
E. 4

Recall: Q(K) is the solution to LLSEA(K), i.e. the length of the longest increasing subsequence ending at digit #K.



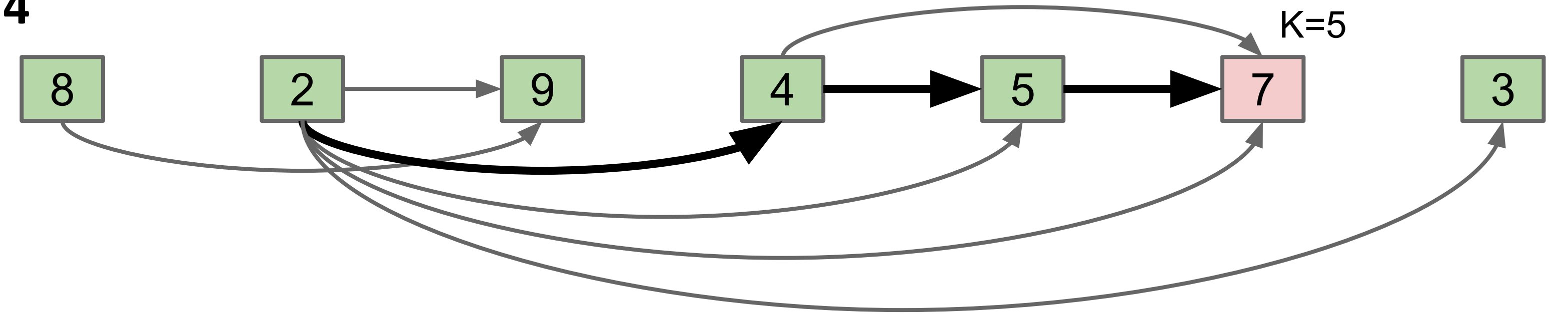
LLIS Subproblems, yellkey.com/miss

LLSEA Problem:

- Given the sequence 8, 2, **9**, 4, 5, 7, 3, the LLSEA(2) = 2.

What is Q(5)?

A. 0
B. 1
C. 2
D. 3
E. 4

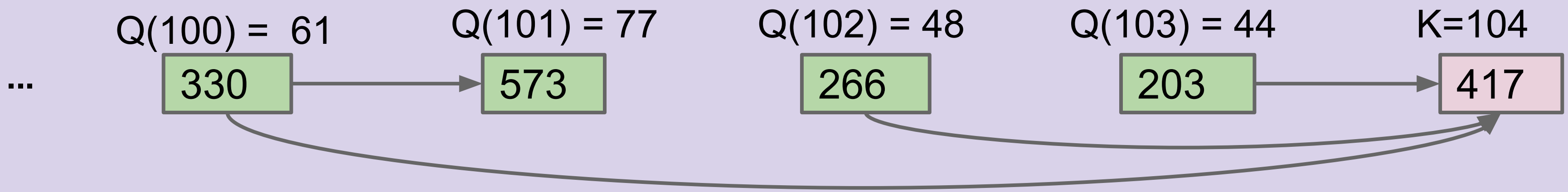


LLSEA Problem and Memoization, yellkey.com/computer

Nice Fact: We can use results for small Q to compute results for large Q.

Example: Suppose we know Q(0) through Q(103). What is Q(104) for the graph below? Assume all values to the left of 330 are less than 200.

A. 62
B. 78
C. 49
D. 45
E. I don't even...



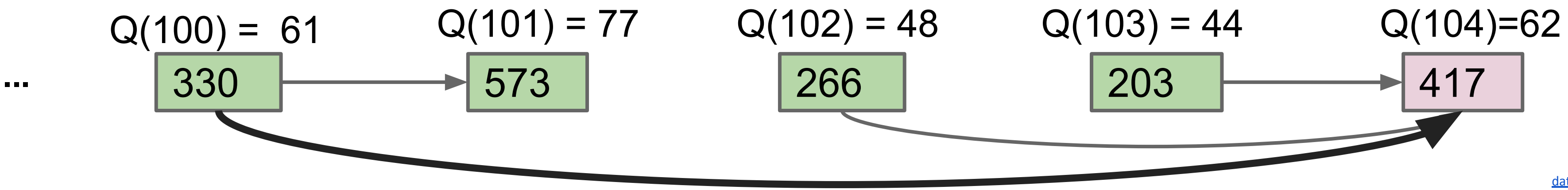
LLSEA Problem and Memoization

Example: Suppose we know Q(0) through Q(103). What is Q(104) for the graph below? Assume all values to the left of 330 are less than 200.

A. **62**: **Because of all the increasing subsequences that have an edge pointing to node #104, the longest one was of length 61 (ending at node #100, which contained value 330).**

Can think of the Q values as <u>**memoized**</u> answers to shorter subproblems.

- The LLIS ending at 417 is the max of 1 + max(*LLIS ending at 266*, *LLIS ending at 330*, …). And we've memoized the answers to those subproblems.



Using LLSEA to solve LLIS

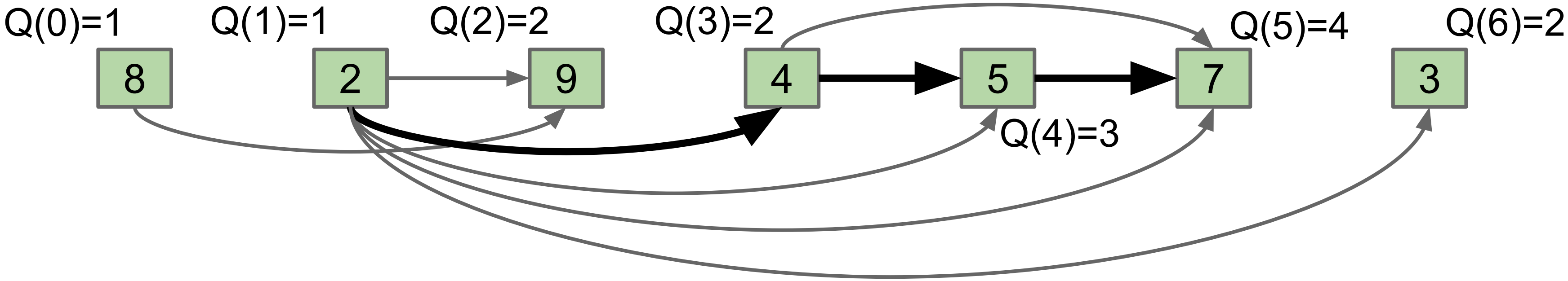
Nice Fact #2: We can use our Q values to solve LLIS of the entire sequence.

Q(K) is the length of the longest subsequence ending at K.

- Thus, length of the longest subsequence is just the maximum of all Q.

Example:

- Q = [1, 1, 2, 2, 3, 4, 2] (solutions to LLSEA)
- LLIS = max(Q) = 4



Dynamic Programming Solution for LLIS

Initialization:

- Create Q, an array of length N. Set Q[0] = 1, and Q[K] = negative infinity.
- Create a DAG with N vertices. Draw an edge between vertices if left vertex is less than right vertex.

For each K = 1, …, N:

- Then for each L = 0, …, K - 1:
  - If there exists an edge from L to K:
    - If Q[L] + 1 > Q[K], set Q[K] = Q[L] + 1

(runtimes are worst case)

Initialization Runtime: Creating Q array and DAG is $\Theta(N)$ and $\Theta(N^2)$, respectively.

Execution Runtime: Nested for loop with constant time work, so $\Theta(N^2)$

DAGLess Dynamic Programming Solution for LLIS

Initialization:

- Create Q, an array of length N. Set Q[0] = 1, and Q[K] = negative infinity.

For each K = 1, …, N:

- Then for each L = 0, …, K - 1:
  - If item L is less than item K:
    - If Q[L] + 1 > Q[K], set Q[K] = Q[L] + 1

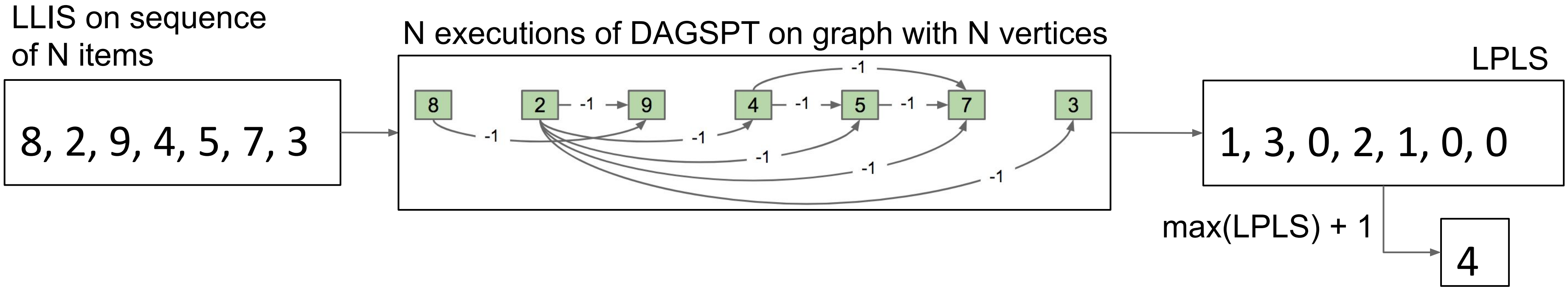
(runtimes are worst case)

Initialization Runtime: Creating array is $\Theta(N)$.

Execution Runtime: Nested for loop with constant time work, so $\Theta(N^2)$

Summary of LLIS Solutions

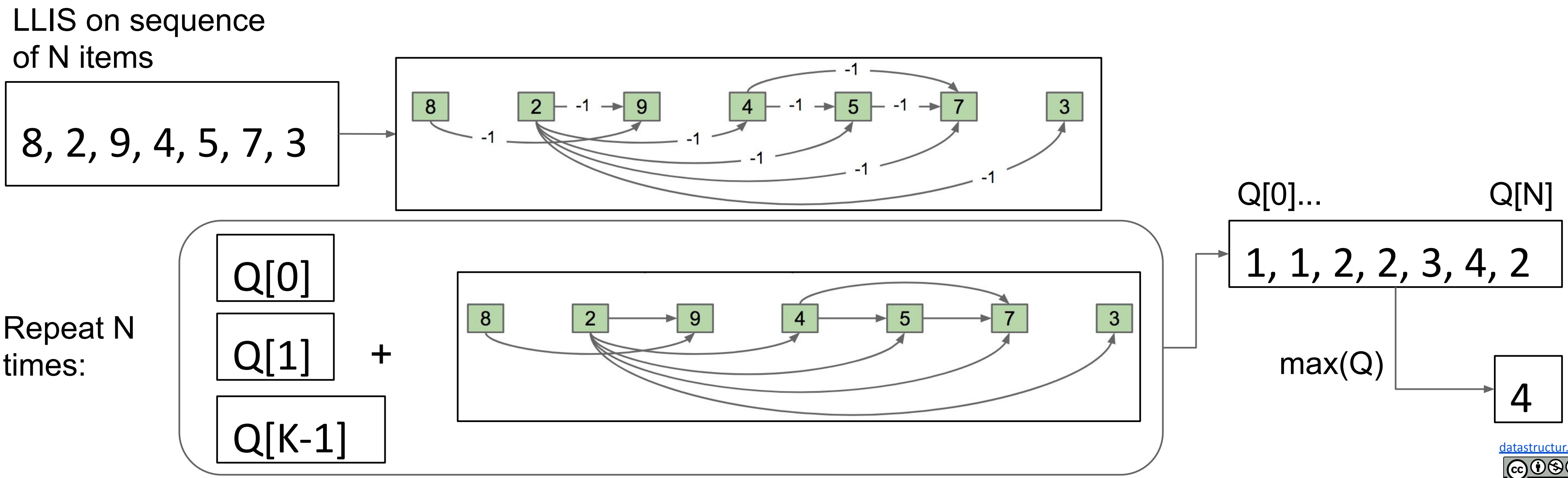
Approach 1: Reduce LLIS to N executions of DAGSPT. Runtime was $\Theta(N^3)$



Summary of LLIS Solutions

Approach 2A: Reduce LLIS to N executions of LLSEA, where we use memoization to make solving large instances of LLSEA easy. Runtime is $\Theta(N^2)$.

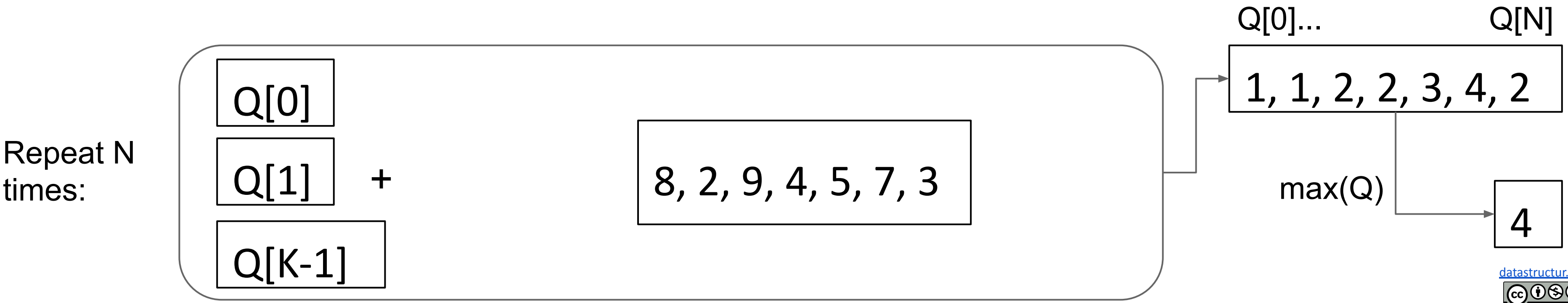
- Can use DAG + solutions to Q[0] through Q[K-1] to compute Q[K] in constant time.
- LLIS solution is max of all Q values.



Summary of LLIS Solutions

Approach 2B: Reduce LLIS to N executions of LLSEA, where we use memoization to make solving large instances of LLSEA easy. Runtime is $\Theta(N^2)$.

- Can use solutions to Q[0] through Q[K-1] to compute Q[K] in constant time.
- LLIS solution is max of all Q values.



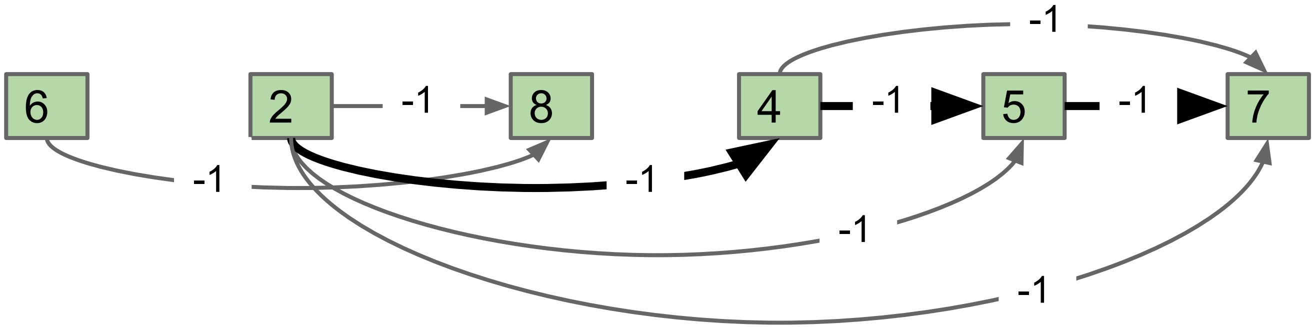
Runtime for Approach 1 (Extra)

The Longest Increasing Subsequence and DAG Connection

How do we find the longest path from any vertex in the DAG?

- Create copy with -1 edge weights.
- For each vertex:
  - Run DAGSPT algorithm. ← Recall, this was a dynamic programming algorithm.
  - Find minimum of distTo array.
- Return abs(minimum of minimums) + 1

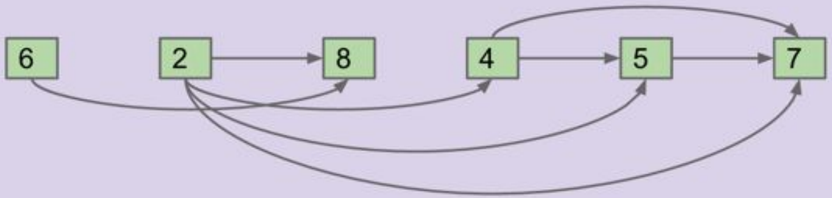
Can show that the runtime of this algorithm is $\Theta(N^3)$. Let's do it...



The Longest Increasing Subsequence and DAG Connection

Longest Path Algorithm:

- Create copy with -1 edge weights.
- For each vertex:
  - Run DAGSPT algorithm.
  - Find max of distTo array.
- Return max of maxes.



What is the runtime of our longest path algorithm?

A. $\Theta(E + V)$
B. $\Theta(EV + V^2)$
C. $\Theta(E^3)$
D. $\Theta(E^2 + EV^2)$

Hint: DAGSPT takes $\Theta(E + V)$ time.

The Longest Increasing Subsequence and DAG Connection

Longest Path Algorithm:

- Create copy with -1 edge weights.
- For each vertex:
  - Run DAGSPT algorithm.
  - Find max of distTo array.
- Return max of maxes.

Operation	# Times	Runtime	Total Time
Copy Graph	1	$\Theta(E + V)$	$\Theta(E + V)$
Run DagSPT	V	$\Theta(E + V)$	$\Theta(EV + V^2)$
Find Max	1	$\Theta(V)$	$\Theta(V^2)$

What is the runtime of our longest path algorithm?

B. $\Theta(EV + V^2)$

Longest Increasing Subsequence Problem

Given the sequence 6, 2, 8, 4, 5, 7. To find the LISS:

- Create a DAG with edge weights -1.
- For each vertex:
  - Run the DAGSPT algorithm.
  - Find the max and record it someplace.
- Return the max of maxes.

Given a sequence of numbers of length N, how to we build the DAG?

- What is the runtime of your algorithm in terms of N?
- How many vertices are there in terms of N?
- In the worst (i.e. largest) case, how many edges are there in terms of N?

Longest Increasing Subsequence Problem

Given the sequence 6, 2, 8, 4, 5, 7. To find the LISS:

- Create a DAG with edge weights -1.

Approach: For each pair of numbers x and y where x comes before y.

- If x < y, then add an edge. Otherwise don't.
- Runtime: $\Theta(N^2)$
- Number of vertices: $\Theta(N)$
- Number of edges: $O(N^2)$

Longest Increasing Subsequence Problem

Given a sequence like 6, 2, 8, 4, 5, 7. To find the LLIS:

- Create a DAG with edge weights -1.
- For each vertex:
  - Run the DAGSPT algorithm
  - Find the max and record it someplace.
- Return the max of maxes.

$O(N^2)$

$O(N^3)$ because it's $\Theta(EV + V^2)$ where E is $O(N^2)$ and V is $\Theta(N)$)

$\Theta(N)$ since edgeTo has length N

Runtime: $O(N^3)$

Operation	# Times	Runtime	Total Time
Create Graph	1	$O(N^2)$	$O(N^2)$
Run DagSPT	N	$O(N^2)$	$O(N^3)$
Find Max	1	$\Theta(N)$	$\Theta(N)$

Citations

Rosalind Dynamic Programming Picture:

- http://rosalind.info/static/img/topics/pictures/dynamic-programming.jpg?v=1384701438