# CSCI 270 Lecture 10: Greedy Algorithms

## All-Pairs Shortest Paths

We want to find the shortest path between all pairs of points (we'll return $n(n-1)$ different answers, one for each pair).

How could we do this, using existing algorithms?

We'll instead calculate this more directly, using dynamic programming, in the hopes that we are able to improve the runtime in some way.

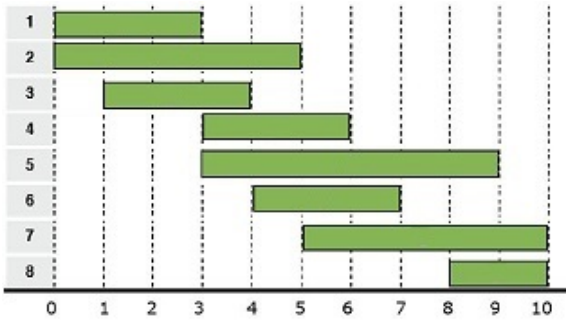
Let $ASP[i, x, z]$ = the length of shortest path from $x$ to $z$ using no more than $i$ edges.

$ASP[i, x, z] = \min_{(x,y)\in E}(c_{(x,y)} + ASP[i-1, y, z])$

- What should my base cases be?
- What order should I fill in the array?
- Where are the answer(s) stored?
- What is the runtime of the algorithm?
- Did we net any improvement in the running time?

## Unweighted Interval Scheduling

We are given $n$ tasks, each with a start time $s_i$, and finish time $f_i$. Find the largest subset of tasks such that none of them overlap.




- What is the size of the largest possible subset?
- Propose a greedy criteria. How should we decide which interval to include next?
- Can you find counter-examples to any of these criteria?
- Once you've narrowed down your list of candidate algorithms to one, what should you do?

Greedy algorithms are easy to design, easy to write, and very efficient. The tradeoff is that they also tend to be unclear. **Why** does the algorithm work? How can you convince someone that the algorithm is correct?

Greedy algorithms often require a proof of correctness. It is not clear at all that our proposed algorithm for Interval Scheduling is actually correct.

Proving the correctness of an entire algorithm is rather overwhelming, so it is useful to break it down. Just as Greedy Algorithms tackle each bite-size decision sequentially, we will prove the correctness of each of these decisions sequentially, via an inductive proof.

**Inductive Hypothesis:** There is an optimal solution which includes the first $k$ intervals from our solution.

The base case is almost always trivial. Specifically, our base case is $k = 0$.

Now we need to show there is an optimal solution which includes the first $k + 1$ intervals from our solution.

What we know:

- There is some interval $i$ which is the $k + 1$st choice in our solution.
- There is at least one (possibly multiple) optimal solutions which include the first $k$ choices we did.
- We need to show at least one of those optimal solutions also include $i$.

Take an arbitrary optimal solution $OPT$ which includes the first $k$ intervals from our solution. Presumably $OPT$ does not include interval $i$ (otherwise we're done). However, of all the intervals it does include (other than the first $k$), there must be one with smallest finish time. We'll call this interval $j$.

- Who do $f_i$ and $f_j$ compare?
- How should I transform $OPT$?
- What is the runtime of our algorithm?

Extra Problems:

- Chapter 4: exercises 3, 5, 6, 7, 9, 13, 15, 24
- Prove the correctness of the algorithms you found, using exchange arguments.