Week 12: Dynamic Programming

Day 20 (M 4/6): More DP and Floyd-Warshall Algorithm

- **Video (25 min)**: Watch the following video on dynamic programming.
  `https://youtu.be/Txax02l82xA`
- **Exercise (15 min)**: Argue the correctness of the equations given for the all-pairs shortest path DP using the principle of optimality.

- **Video (5 min):** Watch the following video talking about how to calculate DP values.
  `https://youtu.be/ifK3PlggYu4`
- **Exercise (10 min):** Write pseudocode below that computes the values functions $V[i, j, k]$ using a bottom-up approach. Your function should *not* have any recursive calls.

- What is the runtime of your algorithm above?

- **Post any questions about this section to the corresponding Canvas discussion thread prior to the live discussion.**

Day 21 (R 4/9): Dynamic Programming Applications

- **Video (10 min)**: Watch the following video about the dining hall problem X.
  `https://youtu.be/pGVdKjhWF6Q`

- **Class Preparation (15 min)**: Find the DP problem you've been assigned to listed in the worksheet group document. Be sure to familiarize yourself with the problem description below and think about the problem so that you can be ready to jump in with your team to formalize your DP and prepare a presentation. Use the time limit guideline for this exercise as a guide for how long you should be spending on preparation.

  Each group will have 5 minutes to give an impromptu presentation to the rest of the class. Your presentation should describe the problem, give a DP solution with intuition, argue correctness of the DP using the principle of optimality, and give the runtime. If you have time, an example is also nice.

  Shortest Paths with Fuel Given a weighted graph $G = (V, E)$, a start node s, and a destination node t, our goal is to find the shortest path from s to t such that we don't run out of gas (only some nodes have gas stations). You may assume that your tank holds U amount of gas and that the amount of gas used on an edge $(i, j) \in E$ is given by $u \cdot w(i, j)$.

  Production Planning Suppose that we want to schedule production of a product over the next N months to meet predicted demand d_j for each month $i = 1, 2, \ldots, N$. In each month j, if we produce $x_j > 0$ units that is a cost of $f_j + c_j x_j$, where f_j is a fixed setup cost and c_j is an additional cost per unit. If we choose not to produce anything in month j, then the cost is zero. You may assume that anything produced in month j is instantly available so it can be used to meet demand in that month. Further, we are allowed to store inventory between months with a holding cost of h per unit per month stored. Our goal is to find the cheapest production plan to meet demand.

  Sequence Alignment Sequence alignment is used in genetics as a way to measure the similarity between DNA or RNA sequences. Suppose that we are given two strings $s1$ and $s2$, an alignment between two strings inserts blank characters into one or both strings such that the two strings are of equal length. The scoring of the alignment is the number of matching indices of the resulting strings. For example, if $s1$ = 'cake' and $s2$ = 'cae' then inserting one blank into $s2$ yields the alignment 'cake' and 'ca-e' which has a score of 3. Our goal is to find an alignment that maximizes the resulting matches using at most k blank characters.

  Longest Common Subsequence Suppose that we are given two strings $s1$ and $s2$, our goal is to find the longest common subsequence between them. A subsequence does not have to be continuous. For example, if $s1$ = 'unicorn' and $s2$ = 'bnycr' then the longest common subsequence is 'ncr'. This problem stems from the field of bioinformatics.

  Weighted Activity Scheduling Suppose that you are trying to plan your first Saturday after social distancing and you have a list of n possible activities to do where each activity $i = 1, 2, \ldots n$ has a start time s_i, an end time e_i, and a happiness score h_i. The goal is to find the the optimal subset of activities to maximize your total happiness score such that none of the activities overlap. This problem stems from job scheduling on machines.

Number of Ways to Make Change Suppose that you have a set number of twenty dollar bills, ten dollar bills, five dollar bills, one dollar bills, quarters, dimes, and nickels available. The goal is to count all the ways to make x dollars using the change available. returns the total possible number of ways to give exact change for Note that the order of change doesn't matter: 2 nickels and 1 dime is equivalent to 1 dime and 2 nickels. For example, if x=0.50 and we have two quarters, one dime, and 10 nickels available, there are five possibilities.

- **Post any questions about this section to the corresponding Canvas discussion thread prior to the live discussion.**

Extra Resources for the Week

- `https://www.topcoder.com/community/competitive-programming/tutorials/dynamic-programmi`
- `https://medium.com/basecs/less-repetition-more-dynamic-programming-43d29830a630`
- `https://ecal.berkeley.edu/files/ce191/CH05-DynamicProgramming.pdf`