Introduction to Dynamic Programming

Winter 2023

1 Recurrences and Subproblem Reuse

A lot of computational problems can be solved by finding the solutions to complicated recurrences (see: Fibonacci numbers, Shortest paths in DAGs, Bellman-Ford, Longest Common Subsequences). Unfortunately, for all of these problems, the naive recursive algorithms run in exponential time. However, some of the time this can be solved if there is *subproblem reuse*, that is if the recursive algorithm is slow due to needing to solve for the same recursive steps over and over again.

2 Tabulation vs. Memoization

One standard way to solve this is *tabulation*. That is to make a table of all of the sub-problem answers and fill it in one step at a time (this is what we've been doing in lecture).

Another technique is *memoization*. That is to keep track of a hash table of all of the recursive subcalls that have already been evaluated and when evaluating a new recursive subcall to first see if the answer is already in the hash table. Then the algorithm can be implemented as a naive recursion without exponential blowup.

[[Work some simple examples with both techniques]]

These different ways of looking at the problem have some advantages and disadvantages. Tabulation is probably more efficient in most cases as it puts everything into a single clean array. However, tabulation requires that you know ahead of time which subproblems are being solved in what order. Memoization can be better if, for example, some subproblems turn out to be unnecessary to evaluate.

3 Exam Review

If there is additional time, it might be good to go over Exam 2.