Dynamic Programming: Bottum-up Problem Solving

Remington Greko, Tyler Gutowski, and Spencer Hirsch

February 13, 2023

Bottom-up Problem Solving

Richard Bellman described this problem solving technique in the 1950's and called it Dynamic Programming to impress his sponsors. Dynamic Programming is an important problem solving paradigm. Read about Bellman on Wikipedia and summarize the importance of his contributions.

Describe how dynamic programming works: Solve and memoriz- ing solutions to small problems and use these solutions as building blocks to construct a solution to a larger problem from the bottom- up.

Example uses of Dynamic Programming

Give at least three examples problems that use the dynamic program- ming paradigm to solve problems. (You should be able to search finding examples and summarize these in your group's words.)

Example One: (Spencer)

Coin Change Problem:

One problem that can utilize dynamic programming to optimize the solution is the coin change problem. I chose this problem because it reminds me of a problem that we had in CSE 1002 and is something that I often think about because I typically handle cash at work.

The problem goes as follows:

Given an unlimited supply of coins as well as the denominations of coins as input. Find all possible ways the desired change, also given as input, can be returned. There are always different solutions to returning change, however, we typically default to the most convient, least number of coins.

This problem can he solved using recursion, which would most likely be the go-to solution for most programmers. However, using dynamic programming you can optimize this solution bringing down the time,

space and memory complexity of the problem. This being said, using dynamic programming would be the best way to solve this particular problem.

Example Two: ()

Example Three: ()

Good approximations to hard problems

There are Hard Problems: Problems that do not (seem) to have effi- cient solutions. You will explore some of these later in future quests.

There are dynamic programming algorithms that provide good ap- proximate solutions to these hard problems. My memory tells me the 0–1 Knapsack Problem is an example of a hard problem with a good dynamic programming approximation. Report on this example and at least 2 additional hard problems with good approximate solutions.

0-1 Knapsack Problem: (Spencer)

The problem goes as follows:

The 0-1 Knapsack problem is a programming problem where as input you are given, an array of items, [1, 2, ..., n - 1, n], and their respective weights given as an array, where each index in the array corresponds to it's respective item. You are also given a maximum capacity of the the sack that is going to be used to hold the items. You must find the maximum number of items you can fit into the knapsack without exceeding the capacity of the knapsack. However, the quantity of the items that can be put into the bag is either zero or one of each specific item.

The Solution:

Using the iterative dynamic programming approach you would define a 2d array, where the rows corresponds to the index of the items and the weights are defined on the columns. For every weight you can either choose to ignore it or use it when constructing the 2d array. Thus, you can calculate the maximum weight that can be held in the knapsack based on the items and their corresponding weights.

By iterating through the 2d array and constructing all possibilites, comparing the maximum for each to the absolute maximum of the knapsack will give you the best solution to the given problem. The use of dynamic programming gives the most optimal space and time complexity as opposed to the brute force approach that could also be used for this problem.

Name	**Section**
Remington Greko	
Tyler Gutowski	
Spencer Hirsch	One Example use of Dynamic Programming & 0-1 Knapsack Problem algorithm example