

CS 325 Summer 2020

Homework 3

Problem 1 (4 points)

What does dynamic programming have in common with divide-and-conquer? What is a principal difference between them?

Problem 2 (6 points)

Shortest path counting: A chess rook can move horizontally or vertically to any square in the same row or the same column of a chessboard. Find the number of shortest paths by which a rook can move from one corner of a chessboard to the diagonally opposite corner. The length of a path is measured by the number of squares it passes through, including the first and the last squares. Solve the problem:

- (a) by a dynamic programming algorithm.
- (b) by using elementary combinatorics.

Problem 3 (6 points)

Maximum square submatrix: Given an $m \times n$ Boolean matrix B , find its largest square submatrix whose elements are all zeros. Design a dynamic programming algorithm and indicate its time efficiency. (The algorithm may be useful for, say, finding the largest free square area on a computer screen or for selecting a construction site.)

Problem 4 (14 points)

Consider the following instance of the knapsack problem with capacity $W = 6$

Item	Weight	Value
1	3	\$25
2	2	\$20
3	1	\$15
4	4	\$40
5	5	\$50

- (a) Apply the bottom-up dynamic programming algorithm to that instance.
- (b) How many different optimal subsets does the instance of part (a) have?
- (c) In general, how can we use the table generated by the dynamic programming algorithm to tell whether there is more than one optimal subset for the knapsack problem's instance?
- (d) Implement the bottom-up dynamic programming algorithm for the knapsack problem. The program should read inputs from a file called "data.txt", and the output will be written to screen, indicating the optimal subset(s).

data.txt (example)

3 25

2 20

1 15

4 40

5 50

The capacity should be input by the user at runtime.

- (e) For the bottom-up dynamic programming algorithm, prove that its time efficiency is in $\Theta(nW)$, its space efficiency is in $\Theta(nW)$ and the time needed to find the composition of an optimal subset from a filled dynamic programming table is in $O(n)$.

EXTRA CREDIT (3 points)

Implement an algorithm that finds the composition of an optimal subset from the table generated by the bottom-up dynamic programming algorithm for the knapsack problem.

Problems 1, 2, 3, & 4: Written HW will only be submitted in Canvas. You must submit a pdf version.

Problems 4d and Extra Credit: Submit the code and a README file in a zipped folder to TEACH. We will only test execution with an input file named data.txt.