CSCI 3104 Spring 2022
Instructors: Profs. Chen and Layer

Problem Set 7

Due Date .... March 15
Name .... **Alex Ojemann**
Student ID **109722375**
Collaborators ....

Contents

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to LaTeX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this LaTeX template.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please submit this document with no fewer pages than the blank template (or Gradescope has issues with it).
- You are welcome and encouraged to collaborate with your classmates, as well as consult outside resources. You must **cite your sources in this document. Copying from any source is an Honor Code violation. Furthermore, all submissions must be in your own words and reflect your understanding of the material.** If there is any confusion about this policy, it is your responsibility to clarify before the due date.
- Posting to **any** service including, but not limited to Chegg, Reddit, StackExchange, etc., for help on an assignment is a violation of the Honor Code.

2 Standard 19 - Dynamic Programming: Identify the Precise Subproblems

The goal of this standard is to practice identifying the recursive structure. To be clear, you are **not** being asked for a precise mathematical recurrence. Rather, you are being asked to clearly and precisely identify the cases to consider. Identifying the cases can sometimes provide enough information to design a dynamic programming solution.

2.1 Problem 1

Problem 1. Consider the Stair Climbing problem, defined as follows.

- Instance: Suppose we have n stairs, labeled $s_1, \ldots, s_n$. Associated with each stair s_k is a number $a_k \geq 1$. At stair s_k, we may jump forward i stairs, where $i \in \{1, 2, \ldots, a_k\}$. You start on s_1.
- Solution: The number of ways to to reach s_n from s_1.

Your job is to clearly identify the recursive structure. That is, suppose we are solving the subproblem at stair s_k. What precise sub-problems do we need to consider?

Answer. Since we are able to jump i steps forward at each iteration where $i \in \{1, 2, \ldots, a_k\}$, when we are moving recursively backwards from s_k the subproblems are the number of ways to get to s_i for each i. The base case is if k is only 1 more than where you're starting because in that case there's only one way to get to s_k from where you're starting. $\square$

2.2 Problem 2

Problem 2. Fix $n \in \mathbb{N}$. The *Trust Game* on n rounds is a two-player dynamic game. Here, Player I starts with \$100. The game proceeds as follows.

- **Round 1:** Player I takes a fraction of the \$100 (which could be nothing) to give to Player II. The money Player I gives to Player II is multiplied by 1.5 before Player II receives it. Player I keeps the remainder. (So for example, if Player I gives \$20 to Player II, then Player II receives \$30 and Player I is left with \$80).
- **Round 2:** Player II can choose a fraction of the money they received to offer to Player I. The money offered to Player I increases by a multiple of 1.5 before Player I receives it. Player II keeps the remainder.

More generally, at round i, the Player at the current round (Player I if i is odd, and Player II if i is even) takes a fraction of the money in the current pile to send to the other Player and keeps the rest. That money increases by a factor of 1.5 before the other player receives it. The game terminates if the current player does not send any money to the other player, or if round n is reached. At round n, the money in the pile is split evenly between the two players.

Each individual player wishes to maximize the total amount of money they receive.

Your job is to clearly identify the recursive structure. That is, at round i, what precise sub-problems does the current player need to consider? [**Hint:** Do we have a smaller instance of the Trust Game after each round?]

Answer. The base cases are if i=n in which the total money would be split between the two players and if one of the players has no money and it's their turn. The recursive case that each player needs to consider at each round is each version of the game in which the player gives d dollars to the other player (and d is multiplied by 1.5 before the other player receives it) for each d in $\{1, 2, \ldots, t\}$ where t is the number of dollars in the player's hand. □

3 Standard 20- Dynamic Programming: Write Down Recurrences

3.1 Problem 3

Problem 3. Suppose we have an m-letter alphabet $\Sigma = \{0, 1, \ldots, m-1\}$. Let W_n be the set of strings $\omega \in \Sigma^n$ such that ω does not have 00 as a substring. Let $f_n := |W_n|$. Write down an explicit recurrence for f_n, including the base cases. Clearly justify each recursive term.

Answer. We know that for an n letter string not to have 00 as a substring it must have a non 0 letter in either spot of any given two letter substring.

Case 1: The non 0 letter occurs in the first position of the two letter substring.

Case 2: The non 0 letter occurs in the second position of the two letter substring.

So, we have that if $n > 2$ of the current string, $f_n = f_{n-1} + f_{n-2}$, if $n = 2$ then $f_n = m^2 - 1$ because we could have any 2 of m different letters except for 00, and if $n = 1$ then $f_n = m$ because we could have any 1 of m different letters. □

3.2 Problem 4

Problem 4. Suppose we have the alphabet $\Sigma = \{x, y\}$. For $n \geq 0$, let W_n be the set of strings $\omega \in \{x, y\}^n$ where ω contains yyy as a substring. Let $f_n := |W_n|$. Write down an explicit recurrence for f_n, including the base cases. Clearly justify each recursive term.

Answer. We know that for an n letter string not to have yyy as a substring it must have an x in one of the three spots of any given three letter substring.

Case 1: The x occurs in the first position of the three letter substring.

Case 2: The x occurs in the second position of the three letter substring.

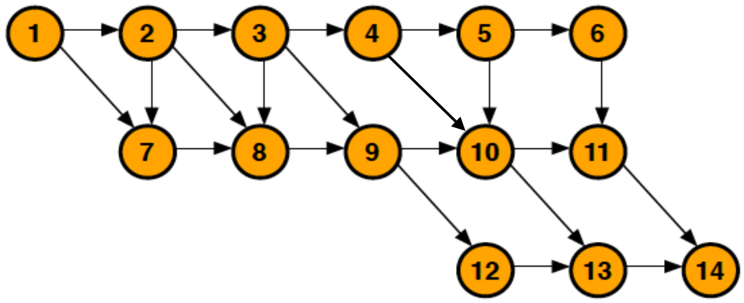
Case 3: The x occurs in the third position of the three letter substring.

The total number of potential strings of n length that could be made using this alphabet is 2^n so to find the number of strings of length n that have yyy as a substring, we subtract the number of strings of n length that don't have yyy as a substring from 2^n. So, we have that if $n > 3$ of the current string, $f_n = 2^n - (f_{n-1} + f_{n-2} + f_{n-3})$, if $n = 1$ then $f_n = 2$ because we could have x or y, if $n = 2$ then $f_n = 4$ because we could have xx, xy, yx, or yy, and if $n = 3$ then $f_n = 7$ because we could have xxx, xxy, xyx, xyy, yxx, yxy, or yyx. $\square$

4 Standard 21- Dynamic Programming: Using Recurrences to Solve

4.1 Problem 5

Problem 5. Given the following directed acyclic graph. Use dynamic programming to fill in a **one-dimensional** lookup table that counts number of paths from each node j to 14, for $j \geq 1$. Note that a single vertex is considered a path of length 0. **Fill in the lookup table for all vertices 1-14; and in addition, clearly show work for vertices 9-14**.



Answer.

Vertex:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Number of Paths:	1	1	1	1	1	1	2	4	5	7	8	5	12	20

Let P(x) represent the number of paths to vertex x.

Vertex 9: P(9) = P(3) + P(8) = P(2) + P(3) + P(2) + P(7) = 1 + P(2) + 1 + P(2) + 1 = 1 + 1 + 1 + 1 + 1 = 5

Vertex 10: P(10) = P(9) + P(4) + P(5) = P(3) + P(8) + P(3) + P(4) = P(2) + P(3) + P(2) + P(7) + P(2) + P(3) = 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) = 1 + 1 + 1 + 1 + 1 + 1 + 1 = 7

Vertex 11: P(11) = P(10) + P(6) = P(9) + P(4) + P(5) + P(5) = P(3) + P(8) + P(3) + P(4) + P(4) = P(2) + P(3) + P(2) + P(7) + P(2) + P(3) + P(3) = 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) + P(2) = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 8

Vertex 12: P(12) = P(9) = P(3) + P(8) = P(2) + P(3) + P(2) + P(7) = 1 + P(2) + 1 + P(2) + 1 = 1 + 1 + 1 + 1 + 1 = 5

Vertex 13: P(13) = P(12) + P(10) = P(9) + P(9) + P(4) + P(5) = P(3) + P(8) + P(3) + P(8) + P(3) + P(4) = P(2) + P(3) + P(2) + P(7) + P(2) + P(3) + P(2) + P(7) + P(2) + P(3) = 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 12

Vertex 14: P(14) = P(11) + P(13) = P(10) + P(6) + P(12) + P(10) = P(9) + P(4) + P(5) + P(5) + P(9) + P(9) + P(4) + P(5) = P(3) + P(8) + P(3) + P(4) + P(4) + P(3) + P(8) + P(3) + P(8) + P(3) + P(4) = P(2) + P(3) + P(2) + P(7) + P(2) + P(3) + P(3) + P(2) + P(3) + P(2) + P(7) + P(2) + P(3) + P(2) + P(7) + P(2) + P(3) = 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) + P(2) + 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) + 1 + P(2) + 1 + 1 + P(2) = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 20 □

4.2 Problem 6

Problem 6. Consider the following input for the Knapsack problem with a capacity $W = 12$:

item i	1	2	3	4	5	6
value v_i	2	7	18	23	29	35
weight w_i	1	2	5	6	7	9

Fill in a lookup table, similar to the example on page 12 of course notes for week 9 (see Week 9 under "Modules" of the course canvas). In addition, clearly explain how you obtain the maximum values/profits $OPT(6, w)$, $w = 7,\ 8,\ 9,\ 10,\ 11,\ 12$.

Answer.

Weight Limit ->	1	2	3	4	5	6	7	8	9	10	11	12
{}	0	0	0	0	0	0	0	0	0	0	0	0
{1}	2	2	2	2	2	2	2	2	2	2	2	2
{1,2}	2	7	9	9	9	9	9	9	9	9	9	9
{1,2,3}	2	7	9	9	18	20	25	27	27	27	27	27
{1,2,3,4}	2	7	9	9	18	23	25	30	32	32	41	43
{1,2,3,4,5}	2	7	9	9	18	23	29	31	36	38	41	48
{1,2,3,4,5,6}	2	7	9	9	18	23	29	31	36	38	42	48

w = 7: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {2,3} which has a value of 25. 25 > 9 so 25 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {2,3} or {1,4} which has a value of 25. 25 <= 25 so the optimal value doesn't change. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {5} which has a value of 29. 29 > 25 so 29 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {5} which has a value of 29. 29 <= 29 so the optimal value doesn't change. The final optimal solution is {5} which has a value of 29.

w = 8: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {1,2,3} which has a value of 27. 27 > 9 so 27 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {2,4} which has a value of 30. 30 > 27 so 30 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {1,5} which has a value of 31. 31 > 30 so 31 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {1,5} which has a value of 31. 31 <= 31 so the optimal value doesn't change. The final optimal solution is {1,5} which has a value of 31.

w = 9: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {1,2,3} which has a value of 27. 27 > 9 so 27 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {1,2,4} which has a value of 32. 32 > 27 so 32 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {2,5} which has a value of 36. 36 > 32 so 36 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {2,5} which has a value of 36. 36 <= 36 so the optimal value doesn't change. The final optimal solution is {2,5} which has a value of 36.

w = 10: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {1,2,3} which has a value of 27. 27 > 9 so 27 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {1,2,4} which has a value of 32. 32 > 27 so 32 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {1,2,5} which has a value of 38. 38 > 32 so 38 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {1,2,5} which has a value of 38. 38 <= 38 so the optimal value doesn't change. The final optimal solution is {1,2,5} which has a value of 38.

w = 11: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {1,2,3} which has a value of 27. 27 > 9 so 27 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {3,4} which has a value of 41. 41 > 27 so 41 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {3,4} which has a value of 41. 41 <= 41 so the optimal value doesn't change. The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {2,6} which has a value of 42. 42 > 41 so 42 becomes the new optimal value. The final optimal solution is {2,6} which has a value of 42.

w = 12: The optimal solution if the set of items we can take from is {1} is {1} which has a value of 2. The optimal solution if the set of items we can take from is {1,2} is {1,2} which has a value of 9. 9 > 2 so 9 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3} is {1,2,3} which has a value of 27. 27 > 9 so 27 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4} is {1,3,4} which has a value of 43. 43 > 27 so 43 becomes the new optimal value. The optimal solution if the set of items we can take from is {1,2,3,4,5} is {2,3,4} which has a value of 48. 48 > 43 so 48 becomes the new optimal value The optimal solution if the set of items we can take from is {1,2,3,4,5,6} is {2,3,4} which has a value of 47. 48 <= 48 so the optimal value doesn't change. The final optimal solution is {2,3,4} which has a value of 48. □