Problem Set 8

Due Date .... November 1, 2022
Name .... **Tyler Huynh**
Student ID **109603994**
Collaborators .... **N/A**

Contents

Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Useful links and references on LaTeXcan be found here on Canvas.
- 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.
- You **must** virtually sign the Honor Code (see Section Honor Code). Failure to do so will result in your assignment not being graded.

Honor Code (Make Sure to Virtually Sign the Honor Pledge)

Problem HC. On my honor, my submission reflects the following:

- My submission is in my own words and reflects my understanding of the material.
- Any collaborations and external sources have been clearly cited in this document.
- I have not posted to external services including, but not limited to Chegg, Reddit, StackExchange, etc.
- I have neither copied nor provided others solutions they can copy.

In the specified region below, clearly indicate that you have upheld the Honor Code. Then type your name.

Honor Pledge. I, **Tyler Huynh** on my honor pledge that my submission is a reflection of my own understanding of the material, any and all collaborations/sources have been properly cited, I have not posted any material to external sources, and I have not copied other solutions as my own. □

21 Standard 21 – Dynamic Programming: Identify precise subproblems

Problem 21. 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.

Problem 21(a)

a. 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. **Referenced Levet Notes**
I will begin by providing an example:

We are currently on stair s_k where we can jump forward i stairs. When we are at stair s_k our precise subproblems will consist of how many ways there are to get to s_n from the next step of s_{k+i} ,where $i \in \{1, 2, \ldots, a_k\}$.

Below I will show an arbitrary example:
Suppose we are at stair s_k, $T(k)$ represents the number of ways to reach s_n from s_k, such that:

When we recurse on $T(k)$ we need to consider the case of $T(k+i)$ for all $i \in \{1, 2, \ldots, a_k\}$, in order to determine how many steps it will take us to get to s_n from s_{k+i}. This will represent our precise subproblem for this problem

From this we can see that when we are at the base case of s_n there will only be one way to get to s_n from s_n.

From the above we can see that our precise subproblem for this problem would be how many ways it will take us to reach s_n from s_{k+i} for all $i \in \{1, 2, \ldots, a_k\}$

Our base case for this problem would be s_n where we know that there only exists one way to get to s_n from s_n

Our recursive structure for $T(k)$ would be to find $T(k+i)$ for all $i \in \{1, 2, \ldots, a_k\}$. □

Problem 21(b)

b. 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. **Referenced Levet Notes:**
To begin I will provide an example:

Based off of this question we can see that the base cases for the question would be:

If either player were to not send any money to each other than the game would end.
Or, if we have reached the maximum amount of rounds of n.

For some arbitrary round i the current player could send no money to the other player, such that it causes the game to end, otherwise if the player sends an amount of money that $\neq 0$, than the game would continue, this would represent our **recursive structure.**

We then have a smaller instance of our trust game that is of size $(n - 1)$ rounds. This smaller instance is our **precise subproblem.**

From this eventually we will reach a game of size 1, where only 1 round happened, which would cause the game to terminate.

□

22 Standard 22 – Dynamic Programming: Write Down Recurrences

Problem 22(a)

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. **Referenced Levet notes**
For this problem I will first start by identifying the base cases for the question such that,

$n = 0$, this will occur when the string is of size 0 where there only exists an empty string.
$n = 1$, this will occur when the string is of size 1 with no possibility that the substring of 00 will appear, where m will represent the total size of W_n
$n = 2$, this will occur when the string is of size 2 with the possibility that the substring of 00 will appear or not appear, where m^2 will represent the total size of W_n

I will now identify my explicit recurrence for f_n:

When $n \geq 2$ than there exists two possible cases:

- **Case 1:**
  Consider if the substring that we are given to begin with a 0. Where $w_0 = 0$, if the substring were to begin with a 0 than the character after it must be in the set of $\Sigma^1 = \{1, \ldots, m-1\}$. Thus we know that the second character in the substring must have $m-1$ options. This is to ensure that we do not have the substring of 00 within our string.

  For the rest of the characters in the string, it can be any string that does not contain 00 of size $n-2$ this will be the same as W_{n-2}, we know the size of W_{n-2} will be equal to the size of f_{n-2}, our recursive structure for this case will be:

$$f_n = (m-1)(f_{n-2})$$

- **Case 2:**
  Consider if the substring that we are given were to begin with a character that is not 0. There are $m-1$ characters that are not 0. For the rest of our string it can be any string of size $n-1$ where the substring of 00 does not exist where that will be W_{n-1}. Thus, the size of W_{n-1} is equal to the size of f_{n-1}.

  From this the recurrence structure for this case will be:

$$f_n = (m-1)(f_{n-1})$$

From the above two cases we can then compile our explicit recurrence for f_n considering the two cases, such that:

$$f_n = (m-1)(f_{n-2}) + (m-1)(f_{n-1})$$

Then our final explicit recurrence and including base cases:

$$f_n = \begin{cases} 1 & : n = 0, \\ m & : n = 1, \\ m^2 - 1 & : n = 2, \\ (m-1)(f_{n-2}) + (m-1)(f_{n-1}) & : n > 2. \end{cases}$$

$\square$

Problem 22(b)

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. **Referenced Recitation 10 solutions and Levet notes:**
I will first begin by identifying the base cases for the problem such that:

The first base case will be when n = 0. In this case there are no strings that contain the substring yyy because they are not large enough.
The second base case will be when n = 1. In this case there are no strings that contain the substring yyy because they are not large enough.
The third base case will be when n=2. In this case there are no strings that contain the substring yyy because they are not large enough.

Let W_n be the set of string of length n which do contain the substring yyy, such that we have the following cases:

- **Case 1:**
  We have that $w_0 \neq y$, from this we can see that if the first index in the substring were to not equal y then it must be x, which is just one option.

  For the rest of the characters in the string, it can be any string that does contain yyy of size $n-1$. This will be the same as W_{n-1}, we know the size of W_{n-1} will be equal to the size of f_{n-1}, our recursive structure for this case will be:

  $$f_n = 1 * f_{n-1}$$

- **Case 2 (Subcases):**

  - **2a:**
    We have that when $w_0 = y$ and $w_1 \neq y$, from this we can see that the second index in the string is not equal y.
    For the rest of the characters in the string, it can be any string that does contain the substring yyy of size $n-2$. This will be the same as W_{n-2}, we know the size of W_{n-2} will be equal to the size of f_{n-2}, our recursive structure for this case will be:

    $$f_n = 1 * 1 * f_{n-2}$$

  - **2b(More subcases):**
    We have that when $w_0 = y$ and $w_1 = y$, from this we can see that the second index in the string is equal to y.

    * **2bi:**
      We have that when $w_0 = y$ and $w_1 = y$ and $w_2 \neq y$, from this we can see that the third index in the string is not equal to y. Such that the remaining characters within the string, must be a string of size $n-3$, that does contain the substring of yyy.

      This will be the same as W_{n-3}, we know the size of W_{n-3} will be equal to the size of f_{n-3}, our recursive structure for this case will be:

      $$f_n = 1 * 1 * f_{n-3}$$

* **2b*ii*:**
  We have that when $w_0 = y$ and $w_1 = y$ and $w_2 = y$, from this we can see that the third index in the string is equal to y. Such that the remaining characters in the string, can be any combination of x, y, from within the set of strings $\{x, y\}^{n-3}$.

  From this we know that the size of the string must be of size $n - 3$, where our entire string now contains the substring of yyy, thus we do not care about the remaining characters within the string that contains the substring of yyy.

  The total combinations of strings of size $n - 3$ is 2^{n-3}.

  $$f_n = 1 * 1 * 1 * 2^{n-3}$$

From the above cases for this recurrence we have that our final explicit recurrence for f_n, including the base cases will be:

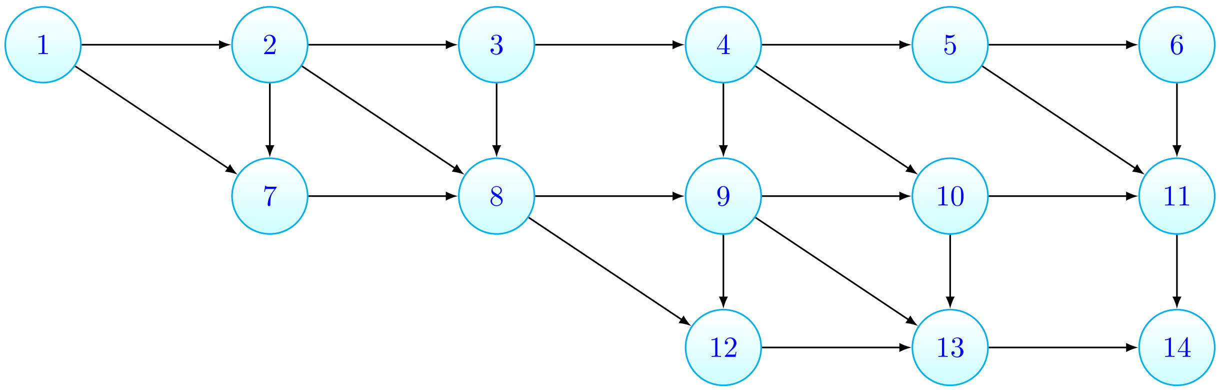
$$f_n = \begin{cases} 0 & : n = 0, \\ 0 & : n = 1, \\ 0 & : n = 2, \\ f_{n-1} + f_{n-2} + f_{n-3} + 2^{n-3} & : n > 2. \end{cases}$$

$\square$

23 Standard 23 – Dynamic Programming: Using Recurrences to Solve

Problem 23(a)

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. **Referenced https://sites.santafe.edu/ãaronc/courses/3104/csci3104_S2018_L6.pdf**
I will first begin by creating the lookup table for vertices $1-14$ as follows:

Vertices:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+X_{8,14}$	$X_{4,14}+X_{8,14}$	$X_{5,14}+X_{9,14}+X_{10,14}$	$X_{6,14}+X_{11,14}$	$X_{11,14}$	$X_{8,14}$	$X_{9,14}+X_{12,14}$	$X_{10,14}+X_{12,14}+X_{13,14}$	$X_{11,14}+X_{13,14}$	$X_{14,14}$	$X_{13,14}$	$X_{14,14}$	1
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+X_{8,14}$	$X_{4,14}+X_{8,14}$	$X_{5,14}+X_{9,14}+X_{10,14}$	$X_{6,14}+X_{11,14}$	$X_{11,14}$	$X_{8,14}$	$X_{9,14}+X_{12,14}$	$X_{10,14}+X_{12,14}+X_{13,14}$	$X_{11,14}+X_{13,14}$	1	$X_{13,14}$	1	1
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+X_{8,14}$	$X_{4,14}+X_{8,14}$	$X_{5,14}+X_{9,14}+X_{10,14}$	$X_{6,14}+1$	1	$X_{8,14}$	$X_{9,14}+X_{12,14}$	$X_{10,14}+X_{12,14}+1$	2	1	1	1	1
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+X_{8,14}$	$X_{4,14}+X_{8,14}$	$X_{5,14}+X_{9,14}+2$	2	1	$X_{8,14}$	$X_{9,14}+1$	4	2	1	1	1	1
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+X_{8,14}$	$X_{4,14}+X_{8,14}$	8	2	1	$X_{8,14}$	5	4	2	1	1	1	1
$X_{j,14}$	$X_{2,14}+X_{7,14}$	$X_{3,14}+X_{7,14}+5$	13	8	2	1	5	5	4	2	1	1	1	1
$X_{j,14}$	$X_{2,14}+5$	23	13	8	2	1	5	5	4	2	1	1	1	1
$X_{j,14}$	28	23	13	8	2	1	5	5	4	2	1	1	1	1

From the above the last line of our lookup table for the vertices from $1-14$ will be:

$$
\begin{gathered}
1 = 28 \\
2 = 23 \\
3 = 13 \\
4 = 8 \\
5 = 2 \\
6 = 1 \\
7 = 5 \\
8 = 5 \\
9 = 4 \\
10 = 2 \\
11 = 1 \\
12 = 1 \\
13 = 1 \\
14 = 1
\end{gathered}
$$

I will now show my work for vertices $9-14$, such that:
For vertex 14 I know that since by the definition of the problem itself that a single vertex is considered to have a path of length 0.

For vertex 13 the work we have is:

$$
\begin{aligned}
X_{13,14} &= X_{14,14} \\
X_{13,14} &= 1
\end{aligned}
$$

For vertex 12 the work we have is:

$$
\begin{aligned}
X_{12,14} &= X_{13,14} \\
X_{12,14} &= 1
\end{aligned}
$$

For vertex 11 the work we have is:

$$
\begin{aligned}
X_{11,14} &= X_{14,14} \\
X_{11,14} &= 1
\end{aligned}
$$

For vertex 10 the work we have is:

$$
\begin{aligned}
X_{10,14} &= X_{11,14} + X_{13,14} \\
&= X_{11,14} + 1 \\
&= 1 + 1 \\
X_{10,14} &= 2
\end{aligned}
$$

For vertex 9 the work we have is:

$$
\begin{aligned}
X_{9,14} &= X_{10,14} + X_{12,14} + X_{13,14} \\
&= X_{10,14} + X_{12,14} + 1 \\
&= X_{10,14} + 1 + 1 \\
&= 2 + 1 + 1 \\
X_{9,14} &= 4
\end{aligned}
$$

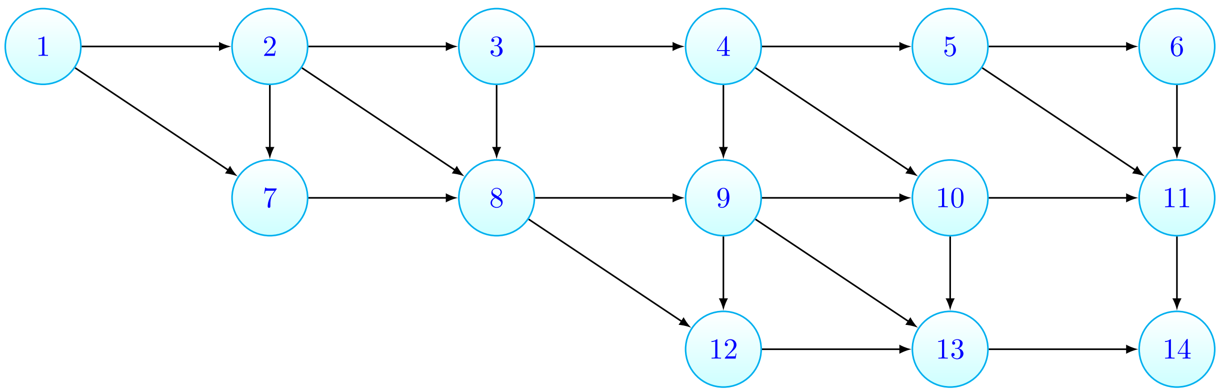
From the above I have found the total number of paths from from each node of j to 14, $j \geq 1$. $\square$

23.1 Problem 23(b)

Consider the following directed acyclic graph (the same as in the previous problem). Use dynamic programming to fill in a **one-dimensional** lookup table that computes the length of the longest path from each node j to 14, for $j \geq 1$. You may use the recurrence

$$L[j] = \begin{cases} 0 & j = 14 \\ 1 + \max\{L[k] : (j,k) \in E(G)\} & j < 14 \end{cases}.$$

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. **Referenced https://sites.santafe.edu/ aaronc/courses/3104/csci3104_S2018_L6.pdf**

I will begin by first creating the lookup table to finds the longest path of each node from j to 14, as follows:

Vertices:	1	2	3	4	5	6	7
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],L[8])$	$1+max(L[4],L[8])$	$1+max(L[5],L[9],L[10])$	$1+max(L[6],L[11])$	$1+max(L[11])$	$1+max(L[8])$
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],L[8])$	$1+max(L[4],L[8])$	$1+max(L[5],L[9],L[10])$	$1+max(L[6],L[11])$	$1+max(L[11])$	$1+max(L[8])$
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],L[8])$	$1+max(L[4],L[8])$	$1+max(L[5],L[9],L[10])$	$1+max(L[6],1)$	2	$1+max(L[8])$
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],L[8])$	$1+max(L[4],L[8])$	$1+max(L[5],L[9],2)$	3	2	$1+max(L[8])$
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],L[8])$	$1+max(L[4],L[8])$	4	3	2	$1+max(L[8])$
$L[J]$	$1+max(L[2],L[7])$	$1+max(L[3],L[7],4)$	5	4	3	2	5
$L[J]$	$1+max(L[2],5)$	6	5	4	3	2	5
$L[J]$	7	6	5	4	3	2	5

Vertices:	8	9	10	11	12	13	14
$L[J]$	$1+max(L[9],L[12])$	$1+max(L[10],L[12],L[13])$	$1+max(L[11],L[13])$	$1+max(L[14])$	$1+max(L[13])$	$1+max(L[14])$	0
$L[J]$	$1+max(L[9],L[12])$	$1+max(L[10],L[12],L[13])$	$1+max(L[11],L[13])$	1	$1+max(L[13])$	1	0
$L[J]$	$1+max(L[9],L[12])$	$1+max(L[10],L[12],1)$	2	1	2	1	0
$L[J]$	$1+max(L[9],1)$	3	2	1	2	1	0
$L[J]$	4	3	2	1	2	1	0
$L[J]$	4	3	2	1	2	1	0
$L[J]$	4	3	2	1	2	1	0
$L[J]$	4	3	2	1	2	1	0

From the above the last line of our lookup table for the vertices from $1-14$ will be:

$$\begin{aligned} 1 &= 7 \\ 2 &= 6 \\ 3 &= 5 \\ 4 &= 4 \\ 5 &= 3 \\ 6 &= 2 \\ 7 &= 5 \\ 8 &= 4 \\ 9 &= 3 \\ 10 &= 2 \\ 11 &= 1 \\ 12 &= 2 \\ 13 &= 1 \\ 14 &= 0 \end{aligned}$$

I From the above I have created the one-dimensional lookup table that computes the longest path from each node j to 14, I will now show my work for vertices $9-14$, such that:
For vertex 14, by the definition of our recurrence $L[j]$, we know that if j were to equal 14, than the path would be

0.
For vertex 13 the work we have is:

$$\begin{aligned}
L[13] &= 1 + max(L[14]) \\
&= 1 + max(0) \\
&= 1 + 0 \\
L[13] &= 1
\end{aligned}$$

For vertex 12 the work we have is:

$$\begin{aligned}
L[12] &= 1 + max(L[13]) \\
&= 1 + max(1) \\
&= 1 + 1 \\
L[12] &= 2
\end{aligned}$$

For vertex 11 the work we have is:

$$\begin{aligned}
L[11] &= 1 + max(L[14]) \\
&= 1 + max(0) \\
&= 1 + 0 \\
L[11] &= 1
\end{aligned}$$

For vertex 10 the work we have is:

$$\begin{aligned}
L[10] &= 1 + max(L[11], L[13]) \\
&= 1 + max(1, 1) \\
&= 1 + 1 \\
L[10] &= 2
\end{aligned}$$

For vertex 9 the work we have is:

$$\begin{aligned}
L[9] &= 1 + max(L[10], L[12], L[13]) \\
&= 1 + max(2, 1, 1) \\
&= 1 + 2 \\
L[9] &= 3
\end{aligned}$$

From the above I have found the longest path from from each node of j to 14, $j \geq 1$. $\square$