CSCI 3104 Fall 2022
Instructors: Prof. Grochow and Chandra Kanth Nagesh

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. Our precise subproblem will consist of when we are at stair s_k how many ways it will take to get from the next step of s_{k+i} ,where $i \in \{1, 2, \ldots, a_k\}$, to the step of s_n, the last step.

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. □

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. □

Problem 22(b)

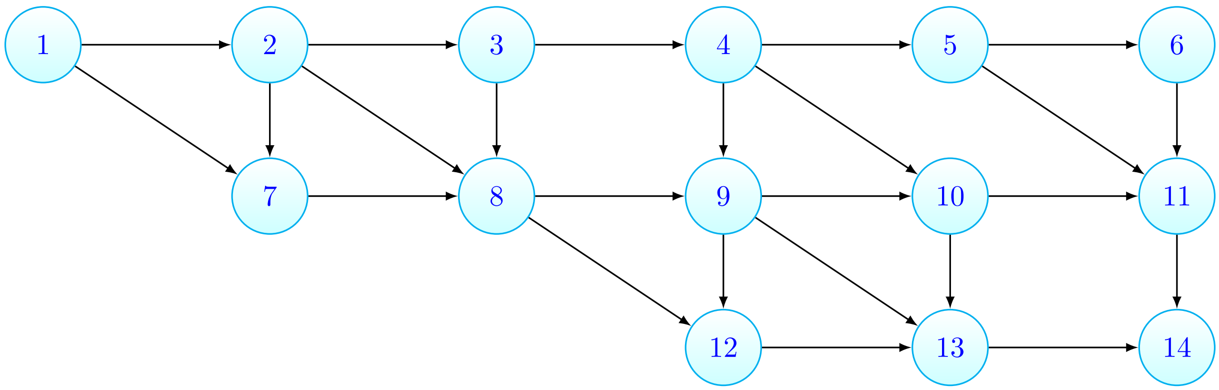
Suppose we have the alphabet $\Sigma = \{x, y\}$. For $n \geq 0$, let W_n be the set of strings $\omega \in \{x, y\}^n$ where ω contains yyy as a substring. Let $f_n := |W_n|$. Write down an explicit recurrence for f_n, including the base cases. Clearly justify each recursive term.

Answer. $\square$

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**.



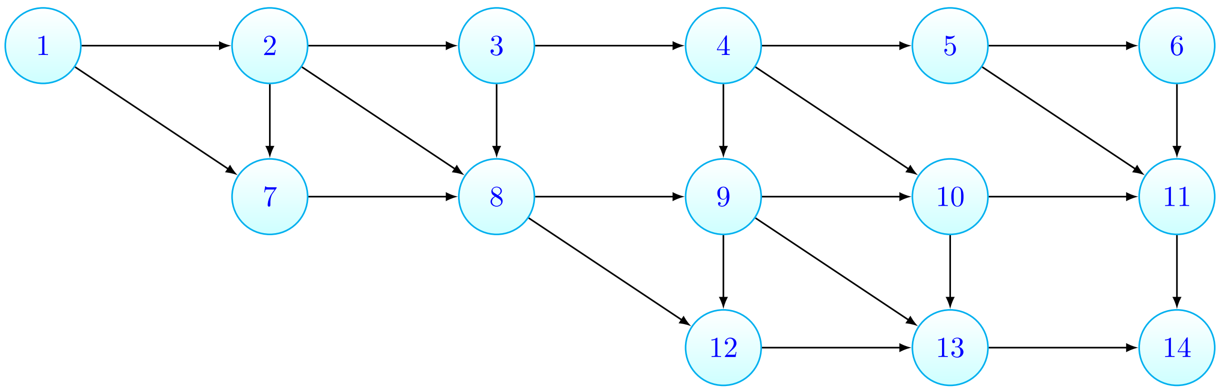
Answer. □

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. □