

Problem Set 8

Due Date November / 2nd / 2021
Name Michael Ghattas
Student ID 109200649
Collaborators Me, Myself & I

Contents

1	Instructions	1
2	Honor Code (Make Sure to Virtually Sign)	2
3	Standard 21- Dynamic Programming: Identify the Precise Subproblems	3
3.1	Problem 2	3
3.2	Problem 3	4
4	Standard 22- Dynamic Programming: Write Down Recurrences	5
4.1	Problem 4	5
4.2	Problem 5	6
5	Standard 23- Dynamic Programming: Using Recurrences to Solve	7
5.1	Problem 6	7
5.2	Problem 7	8

1 Instructions

- The solutions **must be typed**, using proper mathematical notation. We cannot accept hand-written solutions. Here's a short intro to L^AT_EX.
- You should submit your work through the **class Canvas page** only. Please submit one PDF file, compiled using this L^AT_EX 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 2). Failure to do so will result in your assignment not being graded.

2 Honor Code (Make Sure to Virtually Sign)

Problem 1. • 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.

I agree to the above, Michael Ghattas.

□

3 Standard 21- 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.

3.1 Problem 2

Problem 2. Consider the Stair Climbing problem, defined as follows.

- **Instance:** Suppose we have n stairs, labeled s_1, \dots, s_n . Each stair s_k has a number $a_k \geq 1$ associated to it. At stair s_k , we may jump forward i stairs, where $i \in \{1, 2, \dots, a_k\}$. You start on s_1 .
- **Solution:** The number of ways 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:

If we are trying to get from $s_i \rightarrow s_j$ we need to consider where we are s_i , the number of steps we are allowed to jump forward based on the $i \in \{1, 2, \dots, a_k\}$ value associated with s_i , and where we need to go s_j .

□

3.2 Problem 3

Problem 3. 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:

We would consider the amount the player has and the amount the player is willing to transfer, but most importantly the number of the round we are on, depending on whether the $(n - 1)$ round is odd or even.

□

4 Standard 22- Dynamic Programming: Write Down Recurrences

4.1 Problem 4

Problem 4. Suppose we have an m -letter alphabet $\Sigma = \{0, 1, \dots, m-1\}$. 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:

- **Case 1:** If $\omega_1 = 0$, then we have $(m-1)$ elements to select from for ω_2 . So $\omega = \omega_1\omega_2\tau$, where $\tau \in W_{n-2}$. There are $f_{n-2} = |W_{n-2}|$ ways to select τ .
- **Case 2:** If $\omega_1 \neq 0$, then we have $(m-1)$ elements to select from for ω_1 . So $\omega = \omega_1\tau$, where $\tau \in W_{n-1}$. There are $f_{n-1} = |W_{n-1}|$ ways to select τ .

With the initial conditions $f_0 = 0$, $f_1 = (m)$, then $f_n = (m-1) \cdot f_{n-1} + (m-1) \cdot f_{n-2}$ □

4.2 Problem 5

Problem 5. 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:

- **Case 1:** If the last element is x , then there are still f_{n-1} possibilities to get yyy .
- **Case 2:** If the last two elements are xy , then there are still f_{n-2} possibilities to get yyy .
- **Case 3:** If the last three elements are xyy , then there are still f_{n-3} possibilities to get yyy .
- **Case 4:** If the last three elements are yyy , then we have reached our goal and the remaining 2^{n-3} elements can be either x or y .

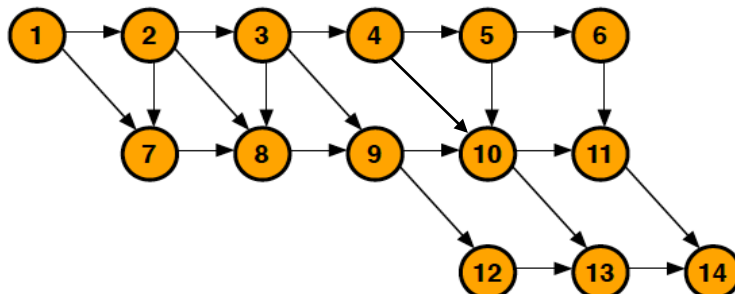
$$f_n = \begin{cases} 0 & : n < 3 \\ f_{n-1} + f_{n-2} + f_{n-3} + 2^{n-3} & : n \geq 3 \end{cases}$$

□

5 Standard 23- Dynamic Programming: Using Recurrences to Solve

5.1 Problem 6

Problem 6. 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	Number of Paths	path(Vertex)
14	1	Same vertex
13	1	$p(13) = p(14)$
12	1	$p(12) = p(13)$
11	1	$p(11) = p(14)$
10	2	$p(10) = p(11) + p(13)$
9	3	$p(9) = p(10) + p(12)$
8	3	$p(8) = p(9)$
7	3	$p(7) = p(8)$
6	1	$p(6) = p(11)$
5	3	$p(5) = p(6) + p(10)$
4	5	$p(4) = p(5) + p(10)$
3	11	$p(3) = p(4) + p(8) + p(9)$
2	17	$p(2) = p(3) + p(7) + p(8)$
1	20	$p(1) = p(2) + p(7)$

□

5.2 Problem 7

Problem 7. Suppose we wish to multiply an $n \times m$ and $m \times k$ matrix. This operation takes nmk multiplications. Consider the Matrix Product problem, which defined as follows.

- **Instance:** Matrices A_1, \dots, A_n with integer entries, where A_i has dimensions $n_i \times m_i$. Note that for $1 \leq i < n$, we necessarily have that $m_i = n_{i+1}$.
- **Solution:** Compute $A_1 A_2 \cdots A_n$, using the minimum number of integer multiplications.

We note that matrix multiplication is associative; that is, $A(BC) = (AB)C$. So we may parenthesize $A_1 A_2 \cdots A_n$ however we wish, without changing the expression. Certain parenthesizations require fewer multiplications than others. For instance, suppose A is a 10×30 matrix, B is a 30×5 matrix, and C is a 5×60 matrix. Observe that $A(BC)$ requires $(10 \cdot 30 \cdot 60) + (30 \cdot 60 \cdot 5) = 27000$ multiplications, while $(AB)C$ requires $(30 \cdot 5 \cdot 10) + (10 \cdot 5 \cdot 60) = 4500$ multiplications.

For $1 \leq i \leq j \leq n$, denote $m[i, j]$ as the minimum number of multiplications required to evaluate $A_i \cdots A_j$. Note that $m[i, j]$ is given by the following recurrence:

$$m[i, j] = \begin{cases} 0 & : i = j, \\ \min_{i \leq k < j} (m[i, k] + m[k+1, j] + n_i m_k m_j) & : i < j. \end{cases}$$

Here, we have that:

- $m[i, k]$ is the minimum number of multiplications to compute $M_1 := A_i \cdots A_k$. Note that M_1 is an $n_i \times m_k$ matrix.
- $m[k+1, j]$ is the minimum number of multiplications to compute $M_2 := A_{k+1} \cdots A_j$. Note that M_2 is an $m_k \times m_j$ matrix (here, $m_k = n_{k+1}$).
- $n_i m_k m_j$ is the number of multiplications required to compute $M_1 \cdot M_2$.

Your job is as follows. Suppose we are given matrices of the following dimension:

- A_1 : 2×8 .
- A_2 : 8×9 .
- A_3 : 9×10 .
- A_4 : 10×20 .
- A_5 : 20×6 .

Design and fill in a 2D lookup table to compute the minimum number of integer multiplications required to compute $A_1 \cdots A_5$. Include the optimal back pointers. [**Note:** You may hand-draw and embed your lookup table, provided it is legible and we do not have to rotate our screens to read it. All other accompanying work **must** be typed.]

Answer:

$$M[1, 5]_{\min} = \begin{cases} [M(1, 1) = 0] + [M(2, 5) = 2172] + [(2 \cdot 8 \cdot 6) = 96] = 2268 \\ [M(1, 2) = 144] + [M(3, 5) = 1740] + [(2 \cdot 9 \cdot 6) = 108] = 1992 \\ [M(1, 3) = 324] + [M(4, 5) = 1200] + [(2 \cdot 10 \cdot 6) = 120] = 1644 \\ [M(1, 4) = 724] + [M(5, 5) = 0] + [(2 \cdot 20 \cdot 6) = 240] = \mathbf{964} \end{cases}$$

$$M[2, 5]_{min} = \begin{cases} [M(2, 2) = 0] + [M(3, 5) = 1740] + [(2 \cdot 8 \cdot 6) = 432] = \textcolor{red}{2172} \\ [M(2, 3) = 720] + [M(4, 5) = 1200] + [(2 \cdot 9 \cdot 6) = 480] = 2400 \\ [M(2, 4) = 2320] + [M(5, 5) = 0] + [(2 \cdot 10 \cdot 6) = 960] = 3280 \end{cases}$$

$$M[1, 2]_{min} = \left\{ [M(1, 1) = 0] + [M(2, 2) = 0] + [(2 \cdot 8 \cdot 9) = 144] = \textcolor{red}{144} \right.$$

$$M[3, 5]_{min} = \begin{cases} [M(3, 3) = 0] + [M(4, 5) = 1200] + [(9 \cdot 10 \cdot 6) = 540] = \textcolor{red}{1740} \\ [M(3, 4) = 1800] + [M(5, 5) = 0] + [(9 \cdot 20 \cdot 6) = 1080] = 2880 \end{cases}$$

$$M[1, 3]_{min} = \begin{cases} [M(1, 1) = 0] + [M(2, 3) = 720] + [(2 \cdot 8 \cdot 10) = 160] = 880 \\ [M(1, 2) = 144] + [M(3, 3) = 0] + [(2 \cdot 9 \cdot 10) = 180] = \textcolor{red}{324} \end{cases}$$

$$M[4, 5]_{min} = \left\{ [M(4, 4) = 0] + [M(5, 5) = 0] + [(10 \cdot 20 \cdot 6) = 1200] = \textcolor{red}{1200} \right.$$

$$M[1, 4]_{min} = \begin{cases} [M(1, 1) = 0] + [M(2, 4) = 2320] + [(2 \cdot 8 \cdot 20) = 320] = 2640 \\ [M(1, 2) = 144] + [M(3, 4) = 1800] + [(2 \cdot 9 \cdot 20) = 360] = 2304 \\ [M(1, 3) = 324] + [M(4, 4) = 0] + [(2 \cdot 10 \cdot 20) = 400] = \textcolor{red}{724} \end{cases}$$

$$M[2, 3]_{min} = \left\{ [M(2, 2) = 0] + [M(3, 3) = 0] + [(8 \cdot 9 \cdot 10) = 720] = \textcolor{red}{720} \right.$$

$$M[2, 4]_{min} = \begin{cases} [M(2, 2) = 0] + [M(3, 4) = 1800] + [(8 \cdot 9 \cdot 20) = 1440] = 3240 \\ [M(2, 3) = 720] + [M(4, 4) = 0] + [(8 \cdot 10 \cdot 20) = 1600] = \textcolor{red}{2320} \end{cases}$$

$$M[3, 4]_{min} = \left\{ [M(3, 3) = 0] + [M(4, 4) = 0] + [(9 \cdot 10 \cdot 20) = 1800] = \textcolor{red}{1800} \right.$$

$M(A_i, A_j)$	A_1	A_2	A_3	A_4	A_5
A_1	0	144	324	724	964
A_2	144	0	720	2320	2172
A_3	324	720	0	1800	1740
A_4	724	2320	1800	0	1200
A_5	964	2172	1740	1200	0

□