1 (a). HW 6 Recurive Formula for Finding the Maximum Pages

The recursive formula is defined as:

$$d(i,j) = \begin{cases} 0, & \text{if } i = 0, j = 0, \\ 0, & \text{if } d(i-2, j-1), \\ d(j, i-1), & \text{if } j = 1, i \bmod 2 = 0, \\ \min(m, p) + d(j, i-1), & \text{if } j = 1, i \bmod 2 = 1, \\ \max \begin{pmatrix} d(i-1, j) + \min(p_i, m_j), \\ d(i-1, j-1) + \min(p_i, m_j) \end{pmatrix}, & \text{otherwise.} \end{cases}$$

Variable Definitions

- $d(i,j)$: Maximum number of pages reviewed up to day i with j days off taken.
- p_i: Number of pages received on day i.
- m_j: Maximum number of pages the editor can review on the j-th consecutive day of work since the last day off.
- j: Number of days since break.
- i: Current day being considered (1-based index).
- n: Set number of days over a given schedule.

Explanation

We are trying to find the maximum pages the editor can review after a set of n days. If we were to visualize the dynamic programming table, when $j = 1$, every other day should be a break because the editor can only work on $j = 1$ if they had taken a break the previous day. Otherwise, it should check the day $d(i-1, j-1)$ and add the new day's pages worked on. The worked-on pages per day will always be the minimum of m and p, as the value can never exceed m, due to it being the limit. We then search to find the maximum value taken and use that schedule.

2. Extra Credit

We are using the following values $X = (A, B, C, N, O, D, A, B)$ and $Y = (A, C, D, O, C, D, B, C)$ with the algorithm for finding the longest common subsequence.

If i and j are 0, the value stored in the matrix is equal to 0. When we compare two values and they are equal, we point to the value located at $[i-1, j-1]$ and add 1.

If $c[i-1, j] \geq c[i, j-1]$, we use the value stored at $c[i-1, j]$; otherwise, we use the value at $c[i, j-1]$. With these conditions, we get the following dynamic table (without arrows, tho still assume they are there):

0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1
0 1 1 2 2 2 2 2 2
0 1 1 2 2 2 3 3 3
0 1 1 2 2 3 3 3 3
0 1 1 2 2 3 3 3 3
0 1 1 2 2 3 4 4 4
0 1 2 2 2 3 4 4 5
0 1 2 3 3 3 4 4 5

If we following the arrows that would normally get there, keeping a list of the i and j values, and printing the values in the X and Y arrays when they are equal to each other, we get:

(8,8), (8,7), (7, 6), (6, 6), (5, 5), (5, 4), (4, 3), (4, 2), (3, 2), (2, 1), (1, 1)

which equals:

(B, C), (B, B), (A, D), (D, D), (O, C), (O, O), (N, D), (N, C), (C, C), (B, A), (A, A)
Only keep the values that match:
B, D, O, C, A
Then flip, because we started at the end:
A, C, O, D, B