NTUST, CSIE
Algorithms (CS3001301), Spring 2022

Homework 4

Due date: May 18
Total: 10 pts

Problem 4.1. In the code of LOOKUP-CHAIN (checking the slides or p.388 from the textbook), if the first two lines of code, namely,

```
1    if m[i, j] < ∞
2        return m[i, j]
```

are removed, what will happen? (2pts)

這兩行是要判斷：若以前曾經有算過，就不要再重複算，刪掉的話可能比較沒效率

Problem 4.2. Consider a variant of the matrix-chain multiplication problem in which the goal is to parenthesize the sequence of matrices so as to maximize, rather than minimize, the number of scalar multiplications. Does this problem exhibit optimal substructure? (2pts)

Yes, this problem does exhibit optimal substructure.

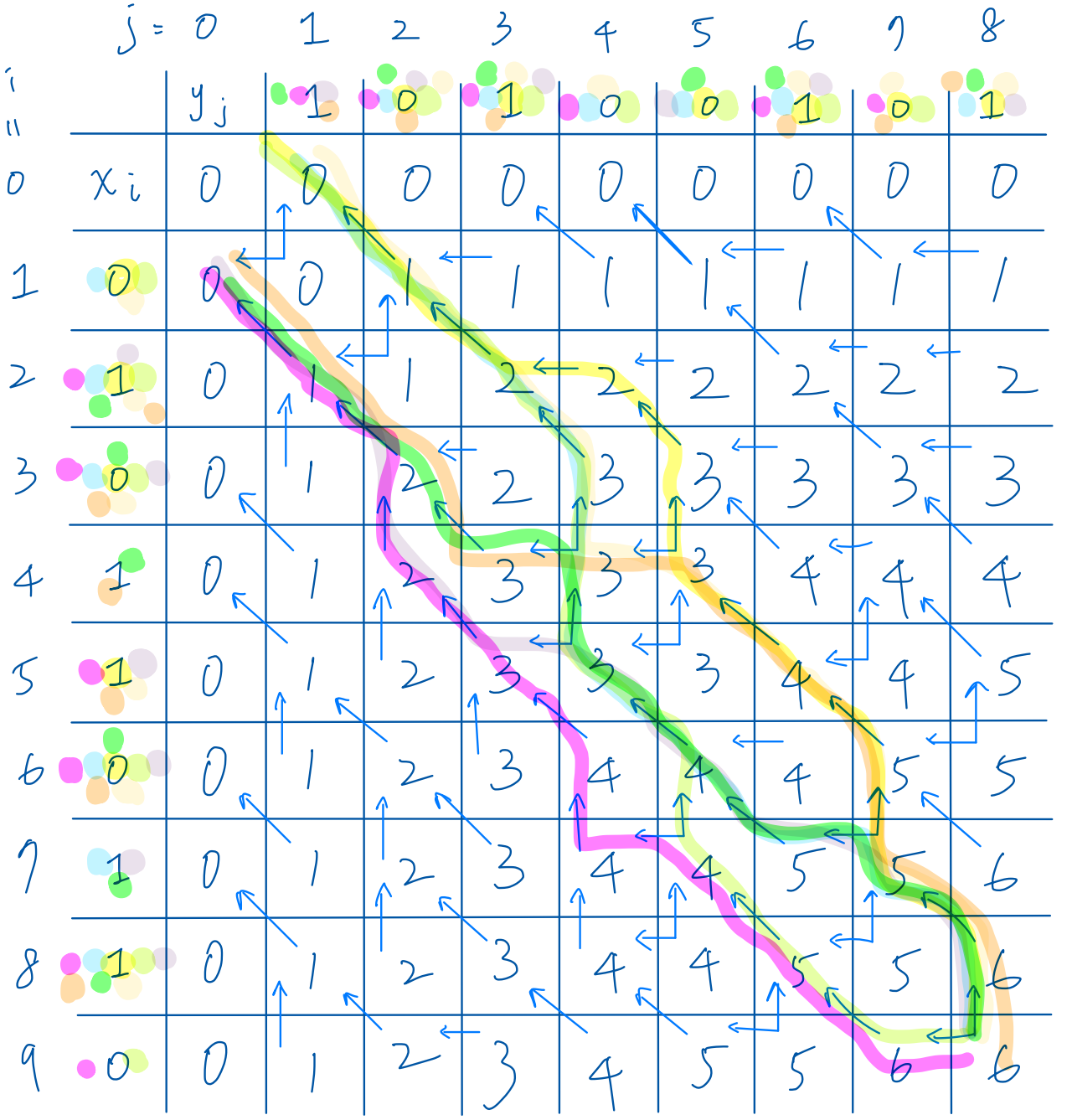
假設 sub-chain 可以找到更多的 multiplications，那其他的 sub-chain 也可以找到更多 multiplications，這會使總體的 multiplications 增加，此假設是不合理的。

所以說一個大問題的最佳解也是拆分後小問題的最佳解。

Problem 4.3. Determine an LCS of $\langle 1, 0, 1, 0, 0, 1, 0, 1\rangle$ and $\langle 0, 1, 0, 1, 1, 0, 1, 1, 0\rangle$. Moreover, as discussed in our class, generalize the recursive formula used in the textbook Eq. 15.9 (p.393) to the following one:

$$c[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ \max(c[i-1,j-1] + (x_i == y_j), c[i,j-1], c[i-1,j]) & \text{if } i, j > 0, \end{cases}$$

($x_i == y_j$ is 1 if $x_i = y_j$ or 0 if not), so that <u>all</u> LCS's of two sequences can be found. Use the recursive formula to find all the LCS's. (2pts)



0 1 0 1 0 1
0 1 0 0 1 0
0 1 0 0 1 1
1 0 1 0 1 0
1 0 1 0 1 1

0 1 0 1 0 1
1 0 1 0 1 1
1 0 1 1 0 1

LCS: 0 1 0 1 0 1
0 1 0 0 1 0
0 1 0 0 1 1
1 0 1 0 1 0
1 0 1 0 1 1
1 0 1 1 0 1

Problem 4.4. Give an $O(n^2)$-time algorithm to find the longest monotonically increasing subsequence of a sequence of n numbers. Discuss two cases separately: (1) simple increasing subsequences (2) strictly increasing subsequences. (2pts)

int A = [$a_1, a_2, \cdots, a_n$] , B = [ ]

(1) simple increasing subsequence

```
LongestIncreasingSubsequence ( A, B)
        sort A and store the result in B
        return LongestIncreasingSubsequence(A, B)
```

時間複雜度 : $\Theta(n \log n) + \Theta(n^2) = \Theta(n^2)$

(2) strictly increasing subsequences

int A = [$a_1, a_2 \ldots a_n$] , B[n] = {0}

```
LIS ( A, B, max)
i = n
j = max
while j > 0
    if B[i] = j
        LIS[j] = A[i]
        j = j-1
    i = i-1
```

Example A = [1, 8, 4, 14, 2, 10, 6]

i = 1 ⟶ A[1] = 1, S = {1}, B[1] = 1
i = 2 ⟶ A[2] = 8, S = {1, 8}, B[2] = 2
i = 3 ⟶ A[3] = 4, S = {1, 4}, B[3] = 2
i = 4 ⟶ A[4] = 14, S = {1, 4, 14}, B[4] = 3
i = 5 ⟶ A[5] = 2, S = {1, 2, 14}, B[5] = 3
i = 6 ⟶ A[6] = 10, S = {1, 2, 10}, B[6] = 3
i = 7 ⟶ A[7] = 6, S = {1, 2, 6}, B[7] = 3

S = {1, 2, 6} ↓ LIS

Problem 4.5. Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time. (2pts)

Activity i	Start time	Finish time
1	1	6
2	3	4
3	5	6

用 greedy 解 : {1}

最佳解 : {2, 3}