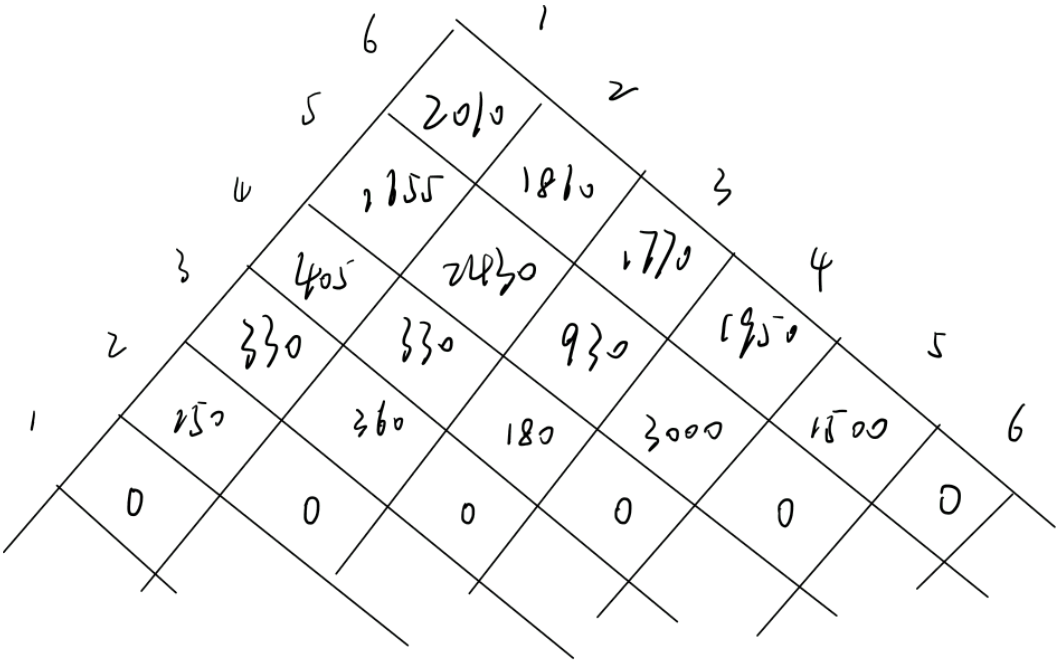
Algorithm Assignment 4

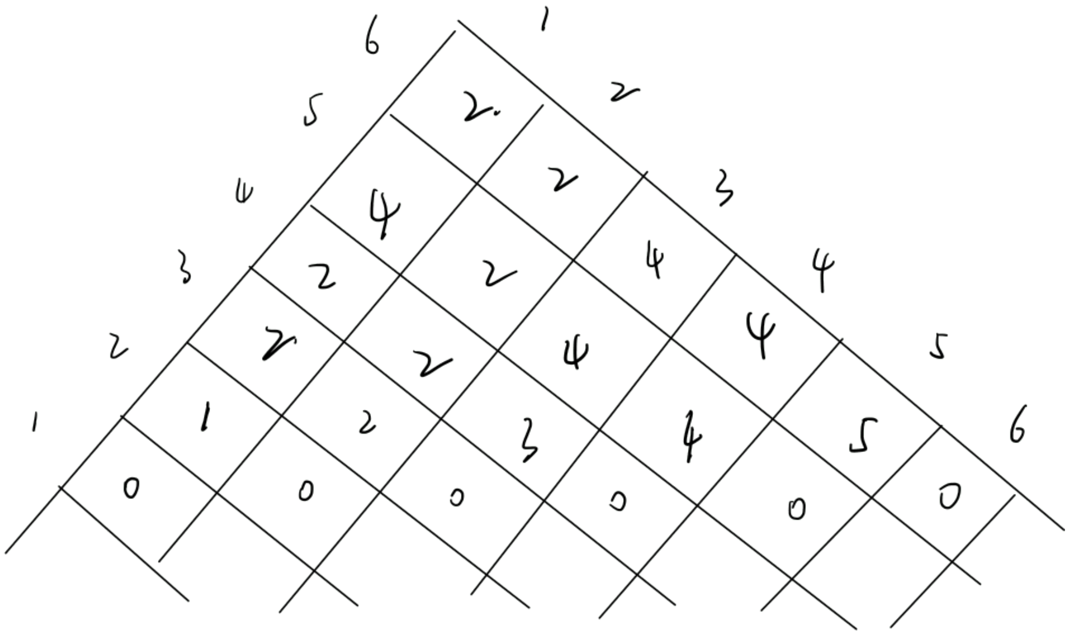
> *10185101210 陈俊潼*

15-2.1

Let
$A_1 = (5 \times 10), A_2 = (10 \times 3), A_3 = (3 \times 12), A_4 = (12 \times 6), A_5 = (5 \times 50), A_6 = (50 \times 6)$



m



s

From the s table, the optimum solution is:

$$(A_1A_2)(A_3A_4)(A_5A_6)$$

15.4-5

Let `Arr` be the input array, `L[n]` denote the longest increasing subsequence length in the subarray [0..n]. So using the dynamic programming, the formula is as follows:

$$\begin{cases} L(i) = 1 + \max(L(j)), \text{where } 0 < \text{j} < \text{i and arr[j]} < \text{arr[i]} \\ L(i) = 1, \text{ if no such j exists.} \end{cases}$$

So by traversing the array, the total complexity is $1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2} = O(n^2)$

15.4-6

Let `A` be the array, `B[n]` be the dp array where `B[n]` is the value of the **smallest element** for sub sequences of A of length n. So `B[0] = 0` is the initial state. Then for each A[n+1], use binary search to find B[k] such that B[k] <= A[n+1] < = B[k+1].

- If A[n+1] is larger than all elements in B, extend array B and add A[n+1] to the end.
- Else, since A[n+1] can combine with all elements before B[k] to generate a new sub sequence, and this will be a smaller result, so update the value of B[k+1] with A[n+1].

Traversing A one by one, and finally, the length of array B is the largest monophonic increasing sub sequence.

Since each traverse uses binary search which gives a complexity of $\log n$, the total complexity for this algorithm is $O(n \log n)$.