A0244126M

Task1 : Min Partition

(a) i. We reduce MIN - PARTITION $\leq_p$ MAX - PARTITION

(b) Define s_i as (s_i, s_i) for $i \in \{1, 2, \dots, N\}$

$$W = \frac{\sum_{1}^{N} s_i}{2}$$

Then S_1 is the subset that maximizes $\sum_{i \in S} s_i$ such that $\sum_{i \in S} s_i \leq W$

S_2 is $S - S_1$

(c) iii. Dynamic Programming (DP) Algorithm

(d) Time complexity : $O(\underbrace{n}_{\text{number of elements}} \underbrace{s}_{\text{sum of all elements}})$

It is <u>not</u> a polynomial solution

(e) iii, We can solve A2 using randomized algo

Task2 : Recombination

(a) v. The correct reduction is N - LCS(F, O)

(b) ii. Dynamic Programming (DP) Algorithm

(c) Time complexity : $O(M \times N^2)$

There are M viruses, each virus requires $O(N^2)$ time to find LCS

This time complexity cannot passes B2 constraints

(d) Yes, we can do so.

⊛ According to A, we know that we can reduce RECOMBINATION $\leq_p$ LCS

⊛ Now, reduce LCS $\leq_p$ LIS

- Create a map to map each element in F to their indexes
  Ex. F: 3, 2, 4, 1, 5

  ↳ F': {3:1, 2:2, 4:3, 1:4, 5:5} (1-index)

- For each C, overwrite the element with the indexes from map F'
  Ex: O: 1, 2, 4, 3, 5

  ↳ O': 4, 2, 3, 1, 5

- Now, instead of finding LCS(F, O), we can find LIS(O'), which equals to LCS(F, O)

(e) Patient Solitaire algorith

① Suppose we have a sequence S from 1 to N with arbitary order

② We create a table N × N
The rule we impose on the matrix: Each column top down must be in descending order

③ The strategy is to go through every element in S, and start inserting into the table
- Starting from left most column, if it is smaller than the lowest element, insert it into that column

- If not, check the following columns

- If cannot fit into any column that has some elements, insert it into a new column

④ After inserting all elements into the table, we pick the lowest element in the right most coluonn and start back tracking.

- For each subsequence, find the element that is smaller the previous element by binary search

↳ Time complexity: $O(M \times N \log \log N)$

suppose there are k columns, then each columns on average has $\log_k N$ elements, then binary seach take $\log_2(\log_k N) = O(\log \log N)$