A - Basketball Exercise

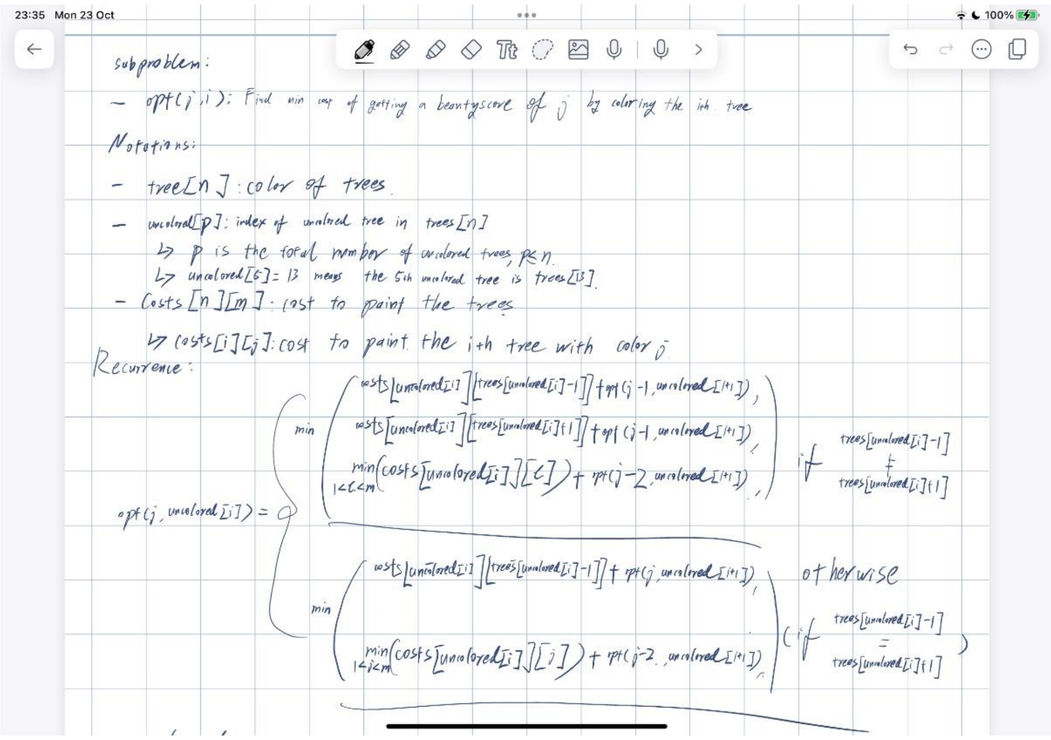
- This question is quite similar to knapsack problem, where at each team member, we can either take this person or skip this person. Hence we have the following algorithm:
  - Setup:
    - Let a1[i] be the height of the ith student in the 1st row
    - Let a2[i] be the height of the ith student in the 2nd row
  - Subproblem:
    - Let dp(i, j) be the max height up to ith student in the jth row where j = 0 or 1
  - Recurrence:
    - dp(i, 0) = max(dp(i-1, 1) + , dp(i-1, 0))
    - dp(i, 1) = max(dp(i-1, 0) + , dp(i-1, 1))
  - Base case: dp(0, j) = 0
  - Overall sol: max(dp(n, 0) dp(n, 1))

B - Colorful Bricks

- Define a sequence of contiguous bricks as a "block" of bricks. Since the bricks within every block have the same colour, the number of bricks that have different colour to the brick on its left is the number of blocks -1, since the first block has no block on its left.
- Now knowing the number of blocks we have, the problem is simply to calculate how many different ways are there to arrange k+1 blocks, then multiply by the number of ways of arrange the colours, which is m*(m-1)^k would be the final solution.
- However, multiplying the number at the end caused me lots of overflow issues, so instead at each dp step, I multiply the result by m-1. This would find the ways of colouring k+1 blocks with the first block being fixed, then I multiply the result by (m-1) to find the overall solution then take mod.
- Hence we have the following algorithm:
  - Setup:
  - Subproblem:
    - Let dp(i, j) be the number of ways to colour i bricks into j blocks with the first block being the fixed colour.
  - Recurrence:
    - $dp(i,j) = (\sum_{p=1}^{i-j+1} dp(i-p, j-1)) \times (m-1)\ \%998244353$
  - Base case:
    - dp(i, 1) = 1
  - Overall sol:
    - dp(n, k+1) * m % 998244353

C - Coloring Trees

- My initial approach was a 2d DP with 3 cases, where dp(i, j) is the minimum cost to colour the ith uncoloured tree with a beauty score of j. The cases are colouring a tree with a colour that is different from the colour to its left and right, different from left/right and same as right/left, and same as left and right. However, my tutor reminded me that this violates the natural ordering of dp as each solution would depend on the solution to its left and right, so I cannot proceed with this dp.



23:35 Mon 23 Oct

subproblem:

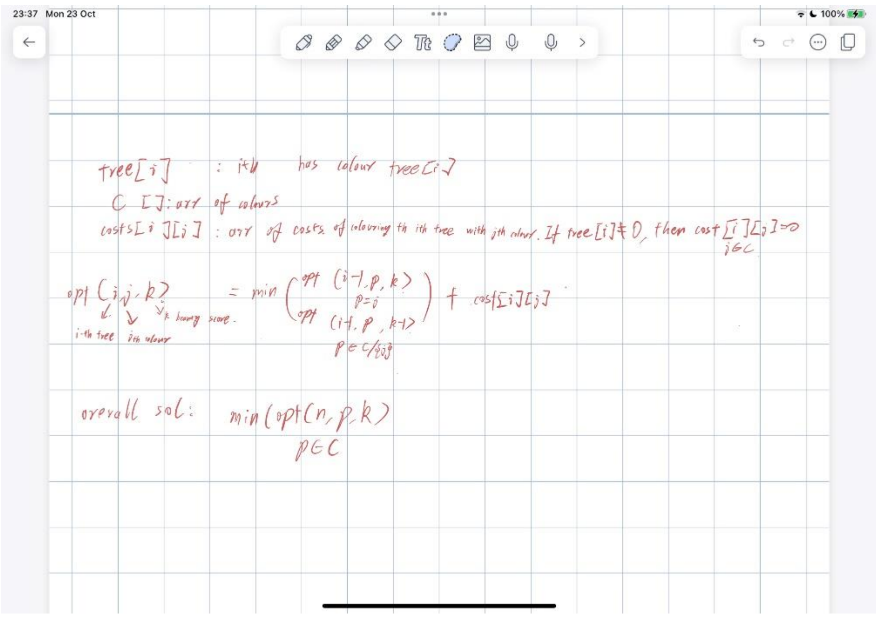
- opt(j, i): Find min cost of getting a beautyscore of j by coloring the ith tree

Notations:

- tree[n]: color of trees
- uncolored[p]: index of uncolored tree in trees[n]
  - ↳ p is the total number of uncolored trees, $p \le n$.
  - ↳ uncolored[5] = 13 means the 5th uncolored tree is trees[13].
- Costs[n][m]: cost to paint the trees
  - ↳ costs[i][j]: cost to paint the ith tree with color j

Recurrence:

$$\text{opt}(j, \text{uncolored}[i]) = \begin{cases} \min\begin{pmatrix} \text{costs}[\text{uncolored}[i]][\text{trees}[\text{uncolored}[i]-1]] + \text{opt}(j-1, \text{uncolored}[i+1]), \\ \text{costs}[\text{uncolored}[i]][\text{trees}[\text{uncolored}[i]+1]] + \text{opt}(j-1, \text{uncolored}[i+1]), \\ \min\limits_{1 \le c \le m}(\text{costs}[\text{uncolored}[i]][c]) + \text{opt}(j-2, \text{uncolored}[i+1]), \end{pmatrix} & \text{if } \text{trees}[\text{uncolored}[i]-1] \ne \text{trees}[\text{uncolored}[i]+1] \\ \min\begin{pmatrix} \text{costs}[\text{uncolored}[i]][\text{trees}[\text{uncolored}[i]-1]] + \text{opt}(j, \text{uncolored}[i+1]), \\ \min\limits_{1 \le j \le m}(\text{costs}[\text{uncolored}[i]][j]) + \text{opt}(j-2, \text{uncolored}[i+1]), \end{pmatrix} & \text{otherwise } (\text{if } \text{trees}[\text{uncolored}[i]-1] = \text{trees}[\text{uncolored}[i]+1]) \end{cases}$$

- To ensure natural ordering, I can only start from one direction. I was then thinking about finding all uncoloured trees then colour them one by one from left to right. However, the issue now is how can I know what the beauty score is after I colour a tree with a certain colour, I have no knowledge of what happened between a uncoloured tree and the next one, maybe all the trees in between have the same colour, or maybe they are all different, there's no way I can ensure that.
- Then the approach I came up with was to add the colour into my dp variable to make a 3D dp. This way I can iterate through all the trees and find the cost of colouring tree i with colour j with beauty score k. If tree i is already coloured with colour c, then there is no extra cost for colouring this tree with colour c, and the extra cost for colouring this tree with colour c is 0, with colours other than c is INF.
- Hence algorithm is as follow:
  - 

    23:37 Mon 23 Oct

    tree[i] : ith has colour tree[i]

    C[]: arr of colours

    costs[i][j] : arr of costs of colouring th ith tree with jth colour. If $\text{tree}[i] \ne 0$, then $\text{cost}[i][j]_{j \in C} = \infty$

    $$\underset{i\text{-th tree}}{\text{opt}}(i, \underset{j\text{th colour}}{j}, \underset{k \text{ beauty score}}{k}) = \min\begin{pmatrix} \text{opt}(i-1, p, k)_{p=j} \\ \text{opt}(i-1, p, k-1)_{p \in C/\{j\}} \end{pmatrix} + \text{cost}[i][j]$$

    overall sol: $\min\limits_{p \in C}(\text{opt}(n, p, k))$

D - Wi-Fi

- For this question, there are a few observations that I made:
  - The cost of building a router and connect directly to wifi are the same, so I would prefer put a router in a room if this room is not covered by wifi from another room.
  - If a room can be covered by multiple routers, then might as well build the router in the earliest possible room since it's cheapest
  - The cover range of a router is k, so all the rooms in the range [i-k. i+k] can be covered by its signal. In other words, this can be translated as:
    - If there is a router in the range [i-k, i+k], then there is no need to directly connect room i to wifi.
- My first thought when reading this question before I had any clue about the actual algorithm is to use a set to store the location of all the routers, so that I can quickly look