Day 1 array questions

Monday, 10 July 2023 16:51

① Set Matrix Zero : LC-73

- The question wants us to set 0 in col and row elements if the a[i][j] is 0.
- We could do it with a copy matrix in which we would alter if we find zero at any place in original matrix
- Better :
  - We take 2 traversals –

    (i) top to down → when moving from top to down, we mark which rows and column we need to change to 0. We keep the first element of each row and column as our marker.

    (ii) while moving from bottom to up we can update the matrix based on the index we receive.

Note : Since first row and first column will have a[0][0] as their marker, which will overlap and cause problem, so we only store a[0][0] to store the state of row.
For column we can use a separate variable.

```
void setZeroes(vector<vector<int>> &matrix)
{
    int col0 = 1, rows = matrix.size(), cols = matrix[0].size();
    for (int i = 0; i < rows; i++)
    {
        if (matrix[i][0] == 0)
            col0 = 0;
        for (int j = 1; j < cols; j++)
            if (matrix[i][j] == 0)
                matrix[i][0] = matrix[0][j] = 0;
    }
    for (int i = rows - 1; i >= 0; i--)
    {
        for (int j = cols - 1; j >= 1; j--)
            if (matrix[i][0] == 0 || matrix[0][j] == 0)
                matrix[i][j] = 0;
        if (col0 == 0)
            matrix[i][0] = 0;
    }
}
```

Q2 Pascal's triangle :

1
1 1
1 2 1
1 3 3 1

- There could be recursive approach as well as an iterative.

Recurrence Relation

Let's start with the recurrence relation within the Pascal's Triangle.

First of all, we define a function $f(i,j)$ which returns the number in the Pascal's Triangle in the `i-th` row and `j-th` column.

We then can represent the recurrence relation with the following formula:

$$f(i,j) = f(i-1, j-1) + f(i-1, j)$$

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

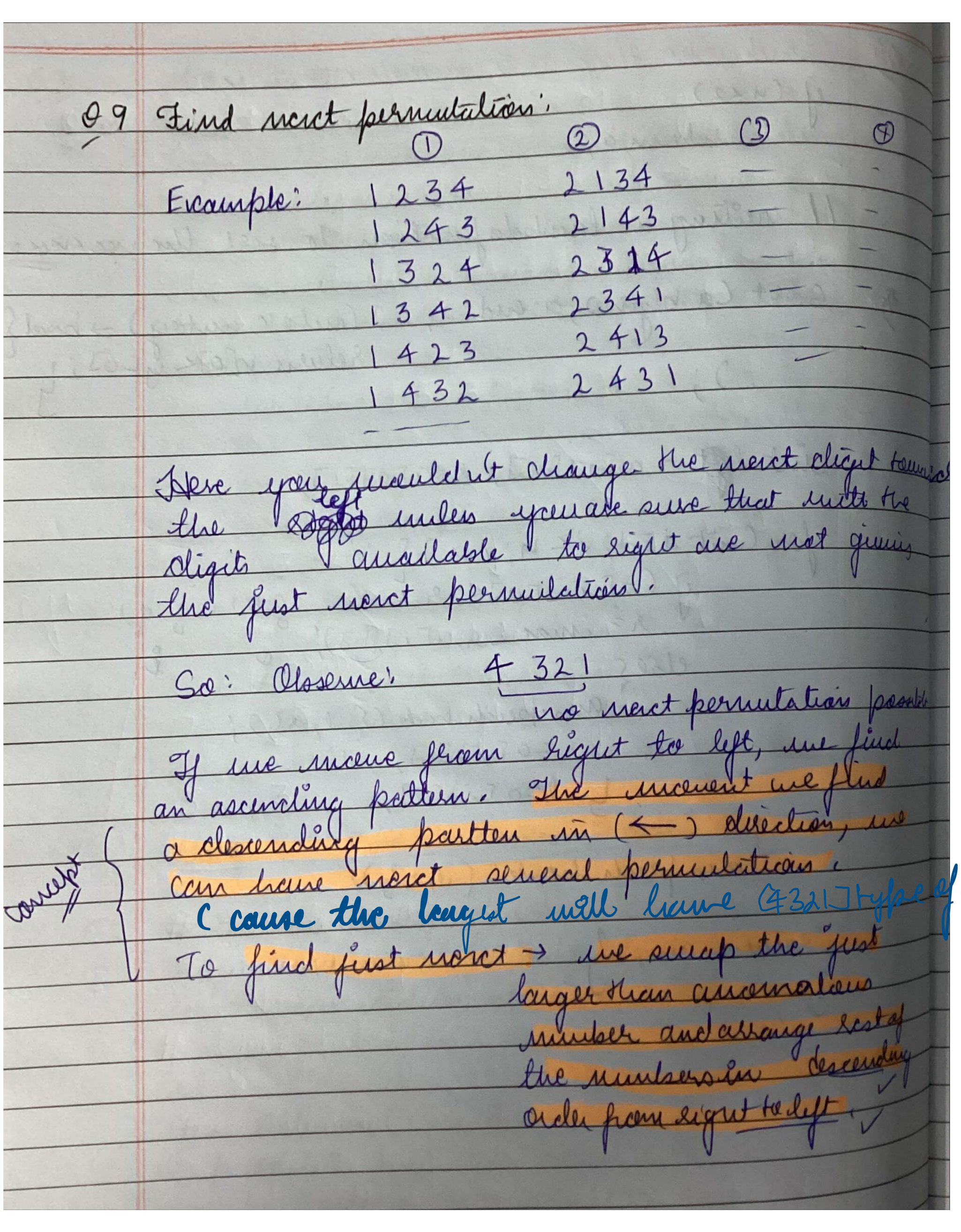
- But I will follow iterative approach.

```
vector<vector<int>> pascalTriangle(int numRows)
{
    //  The first and last element of each row is one, which provide us a
cushion to not go out of bounds in start or end.
    //  Rest of the elements we fill by simple math.
    vector<vector<int>> r(numRows);
    for (int i = 0; i < numRows; i++)
    {
        r[i].resize(i + 1);    // increase the size of next row by 1
        r[i][0] = r[i][i] = 1; // initialize first and last element of each row
as 1
        for (int j = 1; j < i; j++)
            r[i][j] = r[i - 1][j - 1] + r[i - 1][j]; // perform simple addition
of elements of above row
    }
    return r;
}
```

- How to approach?
  - Observe the pattern and try to write the formula for it.
  - first and last digit always 1
  - Each row has 1 larger size vector
  - each middle element = sum of above 2.

    $r[i][j] = r[i-1][j] + r[i-1][j-1]$
  - run a loop and check for out of bound errors.

Q3 Next Permutation :



Q9 Find next permutation:

	①	②	③	④
Example:	1234	2134	–	–
	1243	2143	–	–
	1324	2314	–	–
	1342	2341	–	–
	1423	2413	–	–
	1432	2431	–	

Here you wouldn't change the next digit towards the left unless you are sure that with the digits available to right are not giving the first next permutation.

So: Observe: 4321 no next permutation possible

If we move from right to left, we find an ascending pattern. The moment we find a descending pattern in (←) direction, we can have next several permutations.
(cause the largest will have (4321) type of order.)
To find first next → we swap the first larger than anomalous number and arrange rest of the numbers in descending order from right to left.

concept

```
void nextPerm(vector<int> &nums)
{
    int ano = n - 1;
    while (ano > 0 && nums[ano] <= nums[ano - 1])
    {
        ano--;
    }
    if (ano == 0)
    {
        reverse(nums.begin(), nums.end());
    }
    else
    {
        int pin = n - 1;
        while (pin > ano && nums[pin] <= nums[ano - 1])
        {
            pin--;
        }
        swap(nums[ano - 1], nums[pin]);
        reverse(nums.begin() + ano, nums.end());
    }
    return nums;
}
```

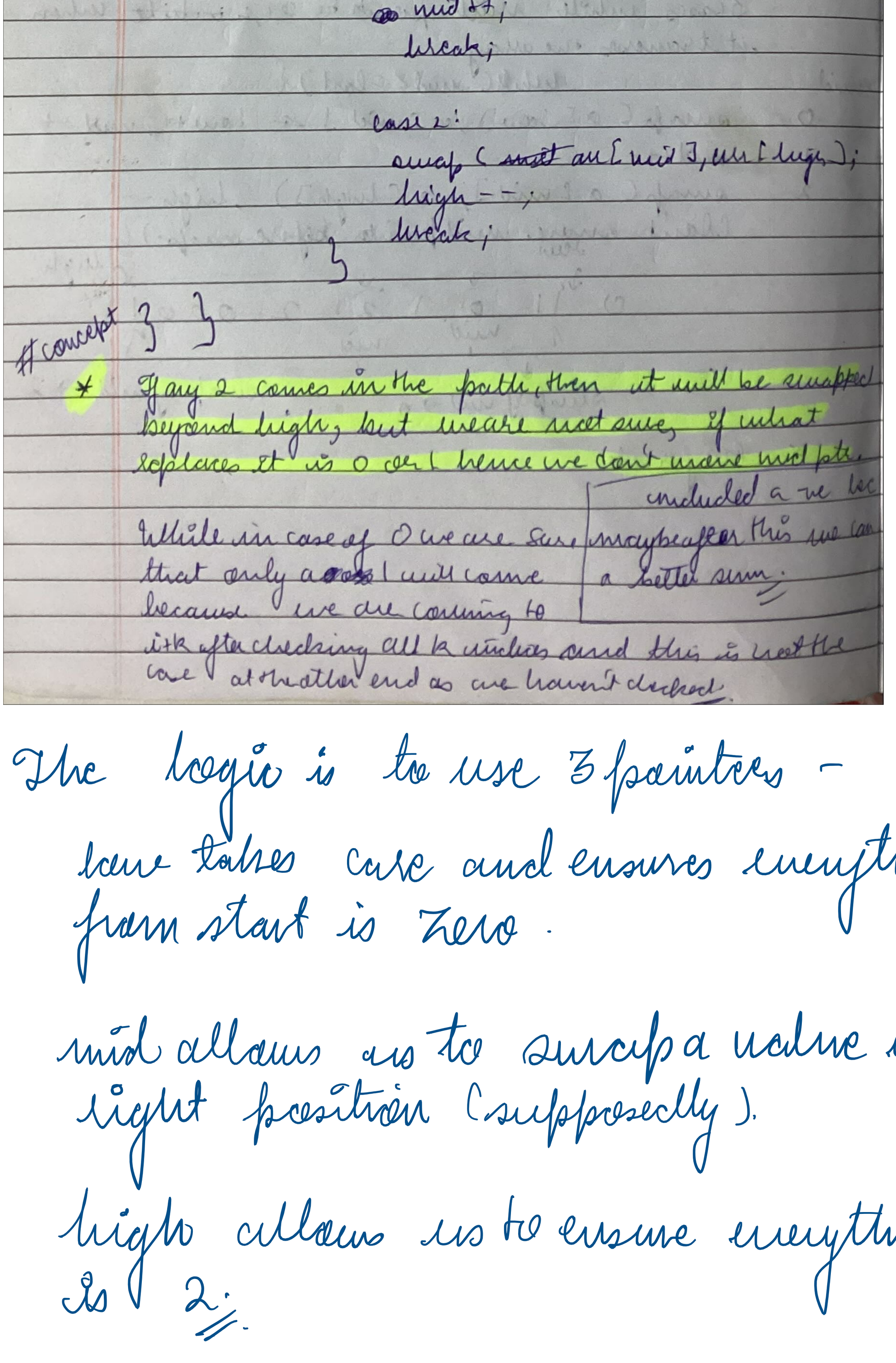
Q4 : Kadane's algorithm : Maximum sum subarray

- This algorithm is greedy approach and it asks us to keep track of 2 things :

  (a) current Best (b) Best So Far
- In the end we return best so far value.

```
#include <bits/stdc++.h>
long long maxSubarraySum(int arr[], int n)
{
    long long curr = 0, overall = 0;
    for (int i = 0; i < n; i++)
    {
        curr += arr[i];
        overall = overall < curr ? curr : overall;
        curr = curr < 0 ? 0 : curr;
    }
    return overall;
}
```

Q5 Dutch National Flag :



```
void sortColors(vector<int>&arr){
  int low =0, mid =0, high = arr.size()-1;
  while(mid<=high){
    switch(arr[mid]){
      case 0:
      swap(arr[low], arr[mid]);
      low++; mid++;
      break;
      case 1:
      mid++;
      break;
      case 2:
      swap(arr[mid], arr[high]);
      high--;
      break;
    }
  }
}
```

#concept
* If any 2 comes in the path, then it will be swapped beyond high, but we are not sure of what replaces it is 0 or 1 hence we don't move mid pts.

While in case of 0 we are sure that only a 1 will come because we are coming to it after checking all the indices and this is not the case at the other end as we haven't checked.

included a ve ... maybe after this we can a better sum.

- The logic is to use 3 pointers –

  (a) low takes care and ensures everything upto it from start is zero.

  (b) mid allows us to swap a value with its right position (supposedly).

  (c) high allows us to ensure everything beyond it is 2.

Q6 Buy and Sell stocks

- Take inspiration from Kadane's algo and keep track of – lowest price so far, maximum profit so far
- Note : update the profit later, after updating the minimum price to keep all transaction in sequence.