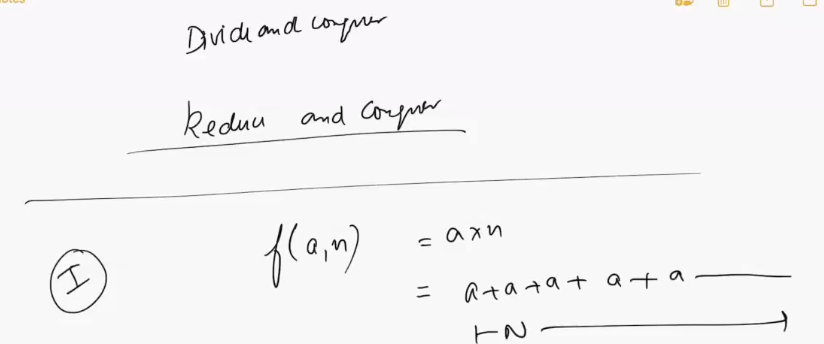
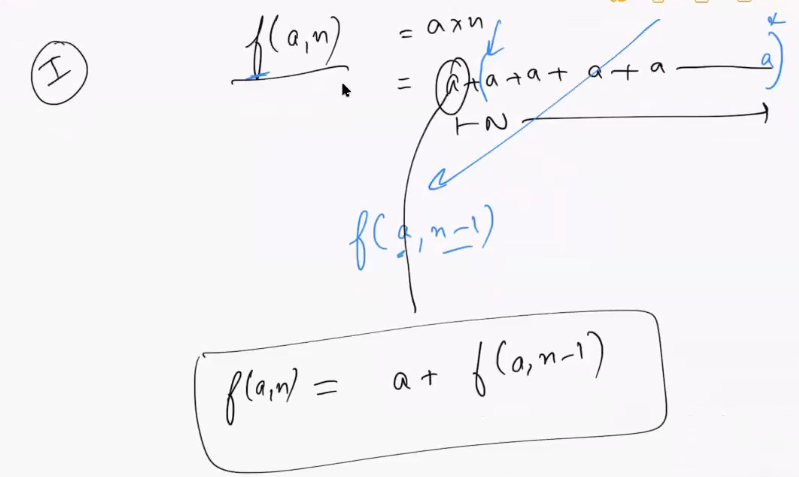
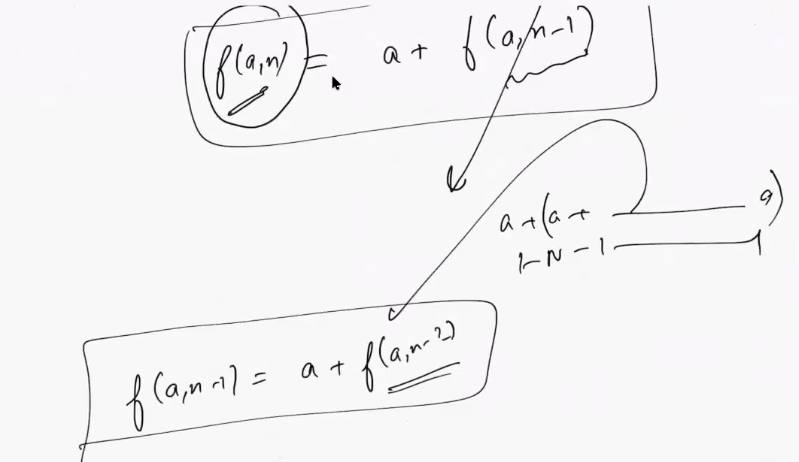
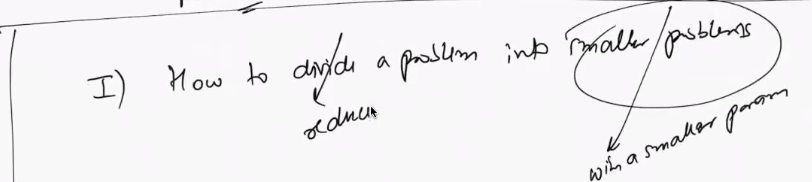
RECURSION



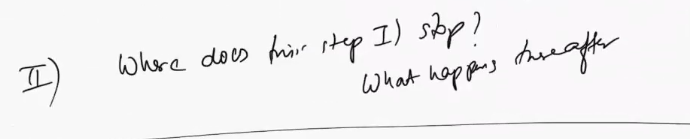


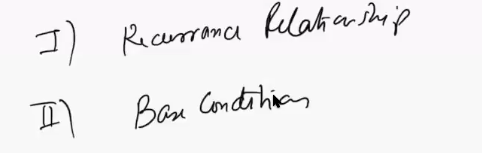


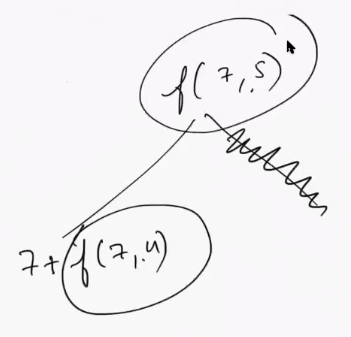
For solving any recursion problem you have to think about 2 things

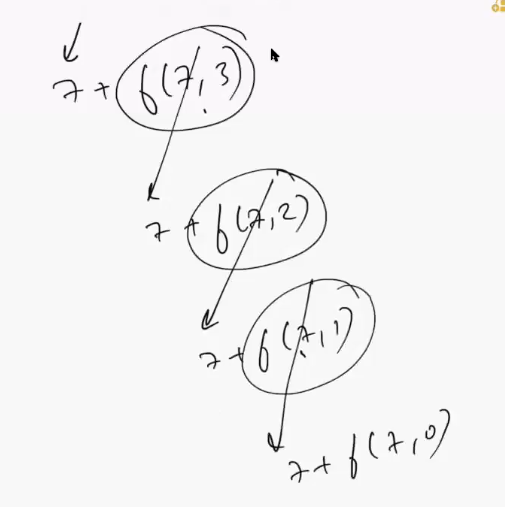


2nd I have to think where do I need to stop and once I stop what to do



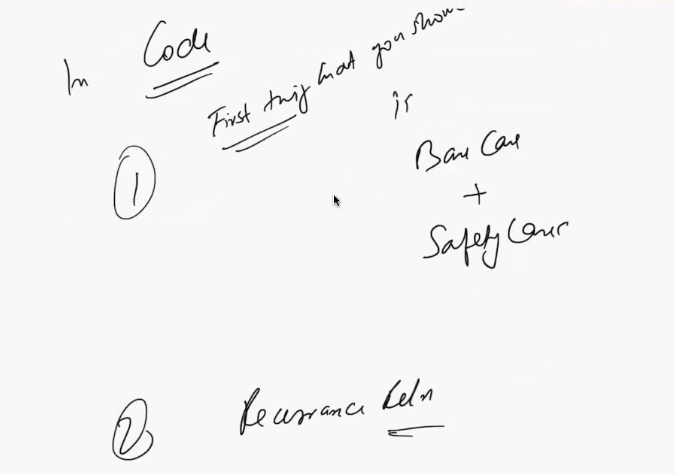


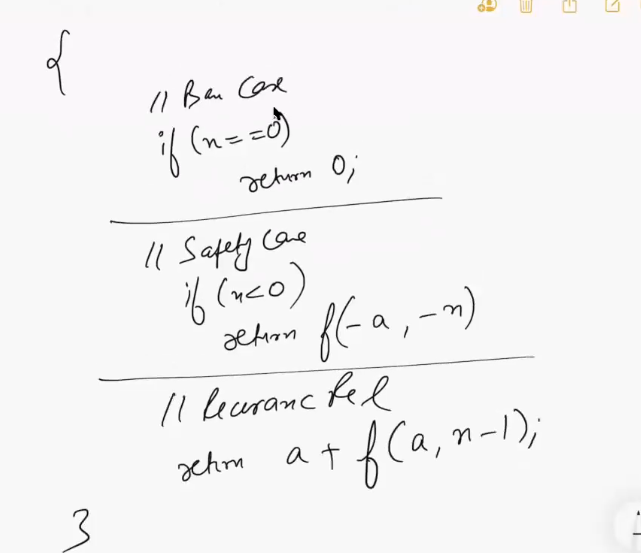




Answer starts to aggregate when program hits the base case

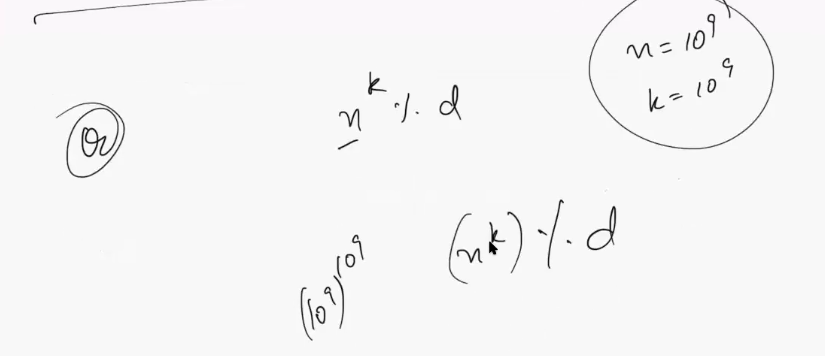
Recursion is very useful while solving complex problems

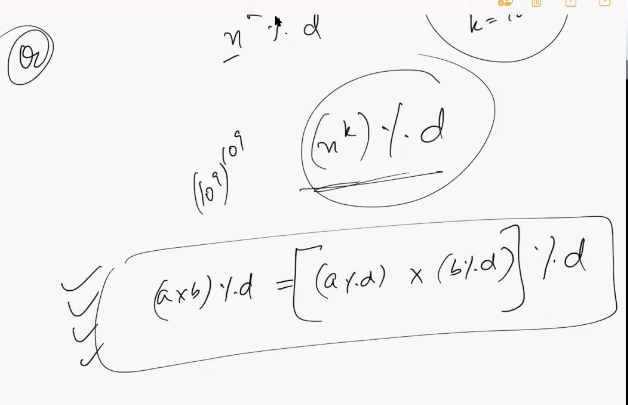


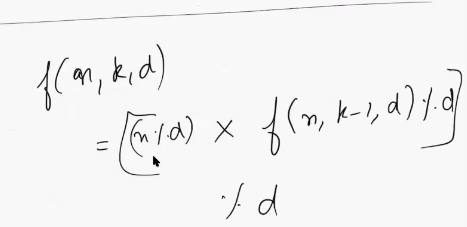


43:00

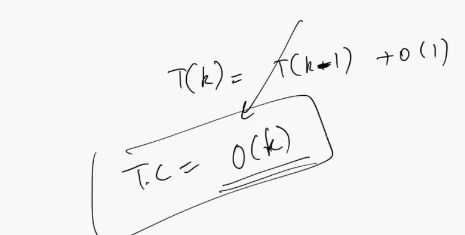
2nd question







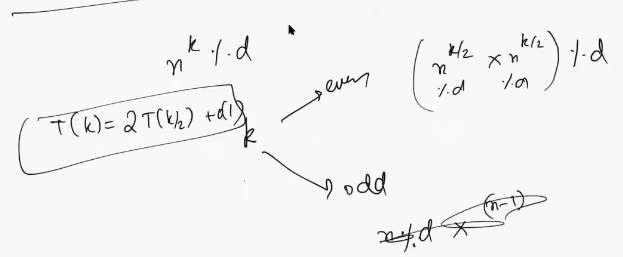
Time complexity

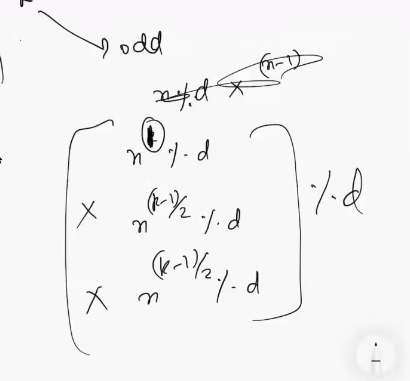


This is not a good solution because if k is 10^9 then the time complexity would be very high

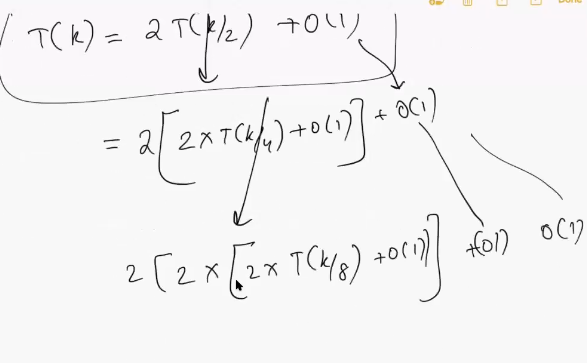
Although this is a correct solution but it is not the most efficient solution

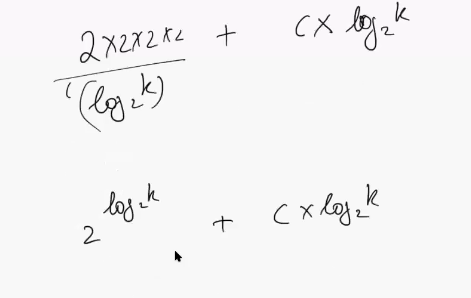
Try to come up with the better recurrence relationship

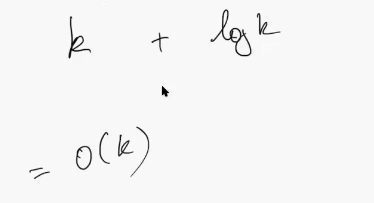




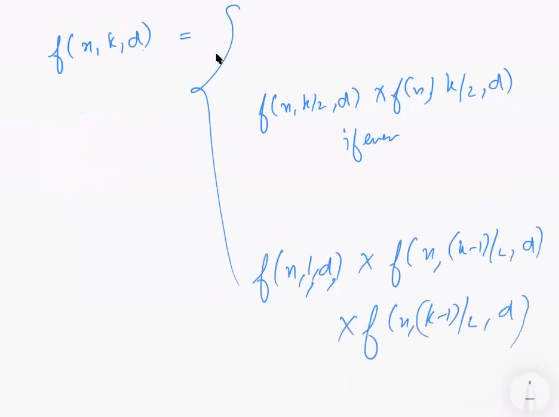
Time complexity



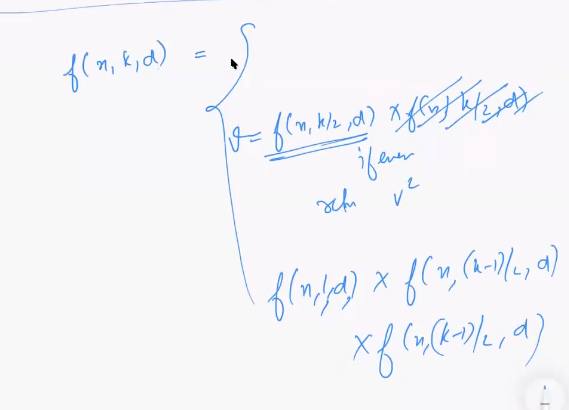


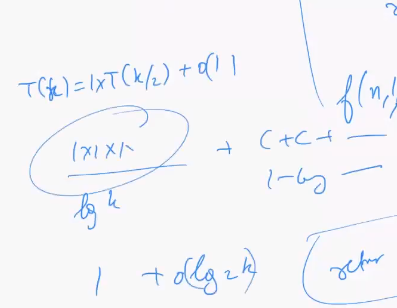


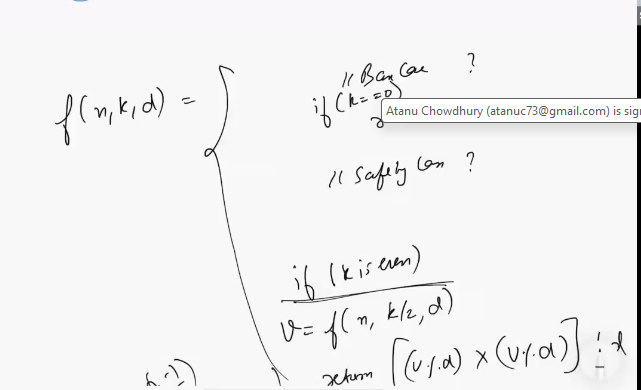
At present our recurrence looks like this



We can store the value in a variable v and return v\*\*2



+

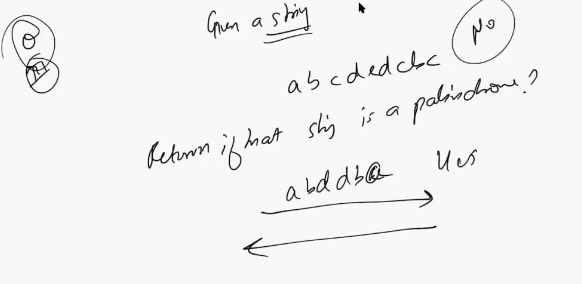


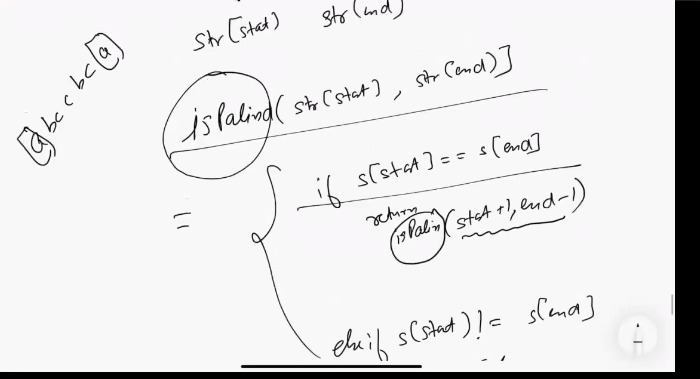
1:08

2nd question

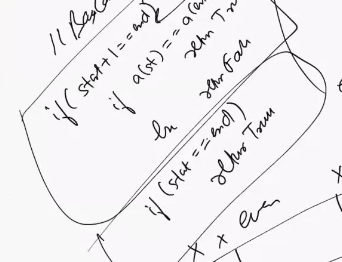
Return if a string is palindrome

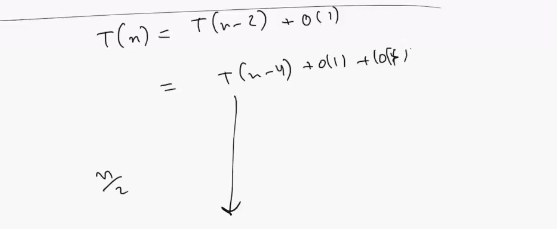
https://leetcode.com/problems/palindrome-number/description/

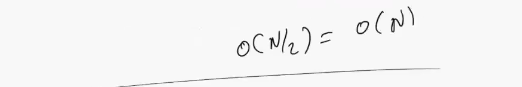


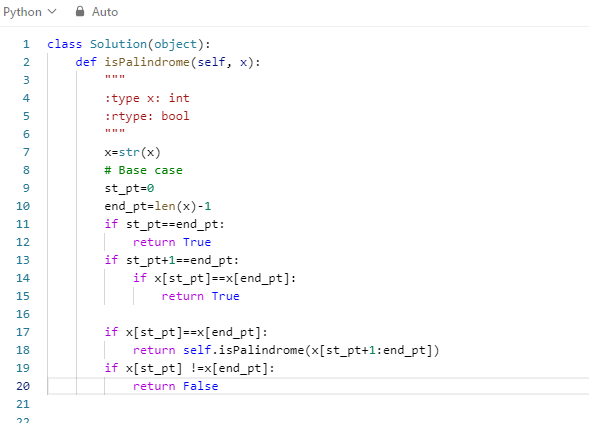


****

****

****

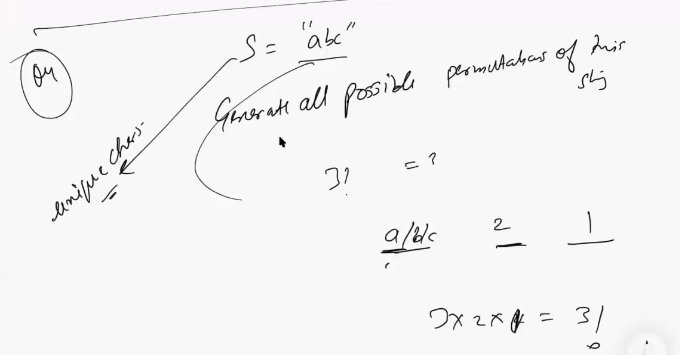
****

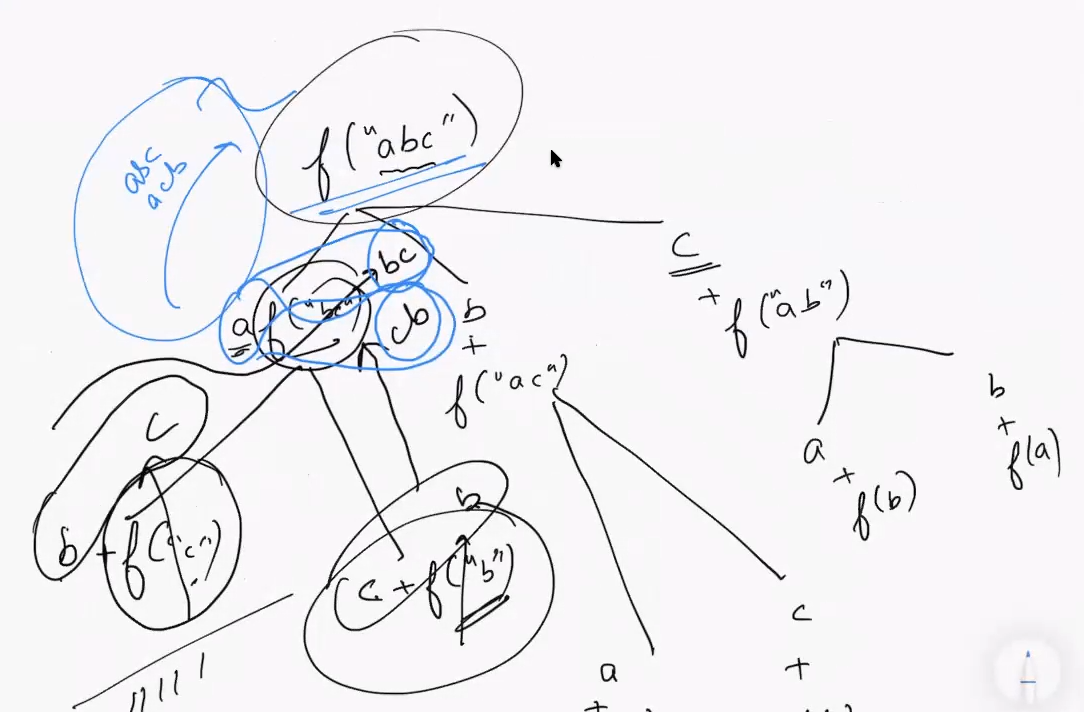
****

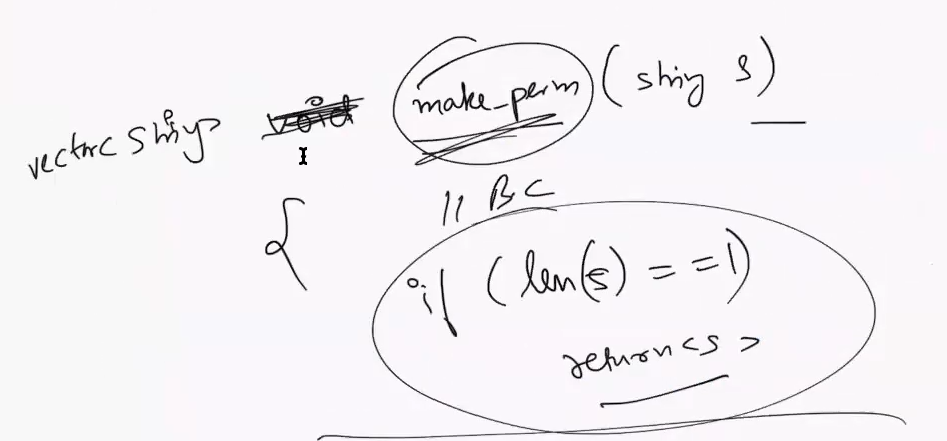
**1:23**

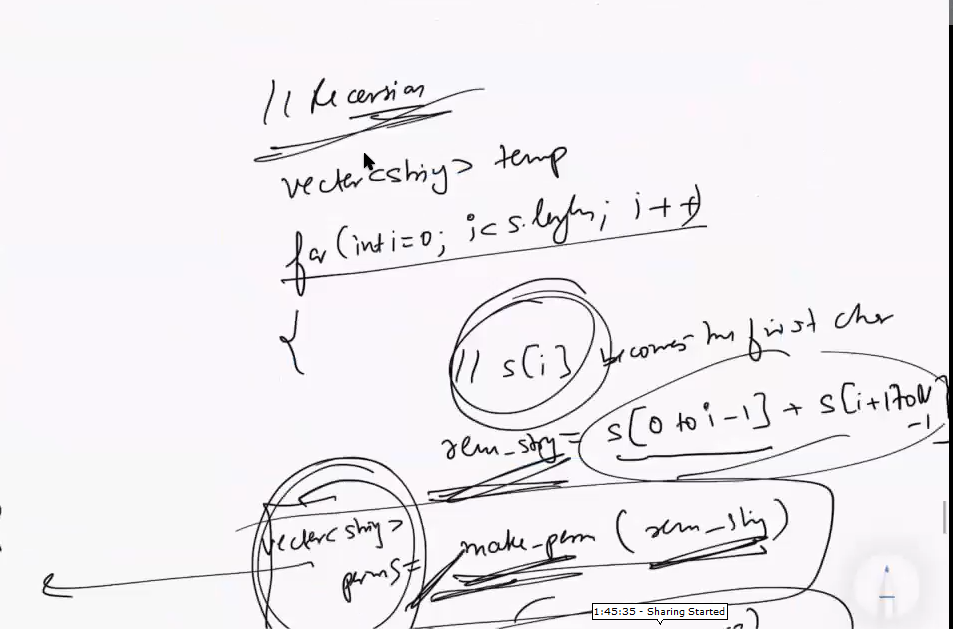
**3rd question**

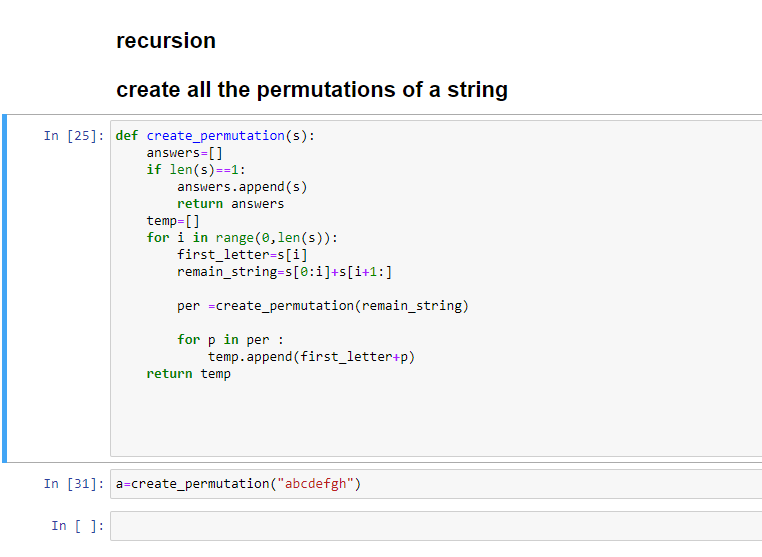
**Generate all the permutation of a string**

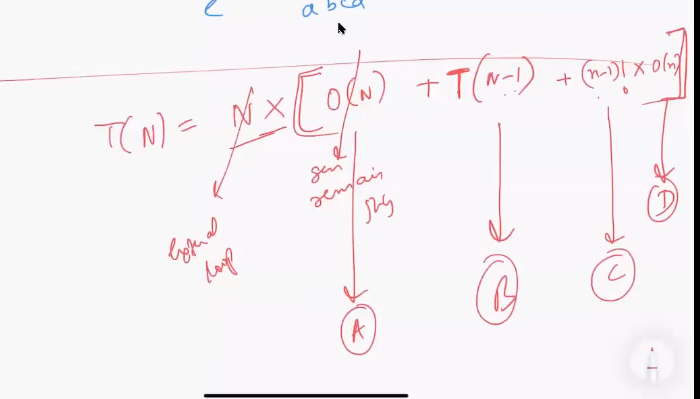
****

****

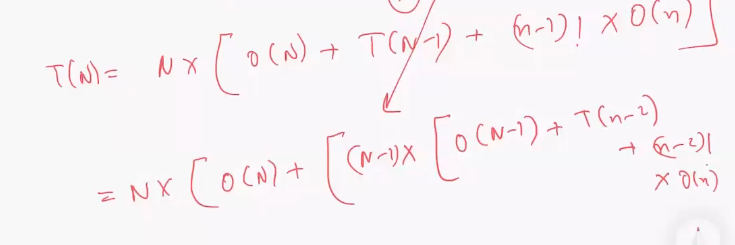
****

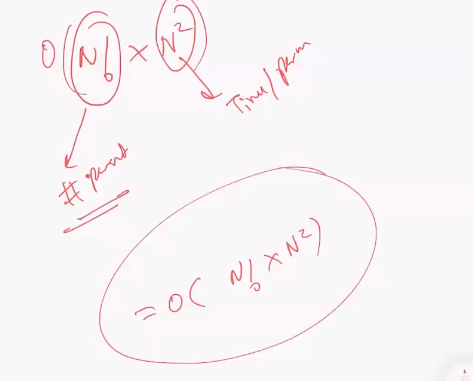
****

****

****

1. **Traverse through the string**
2. **Recursion for the remaining string**
3. **From remaining string we can get (n-1)! Combinations**
4. **It is for copying the solution to the temp**

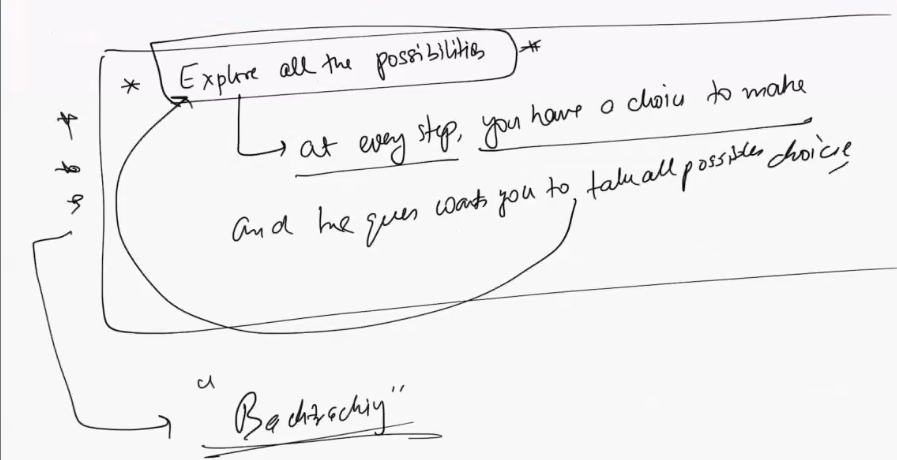


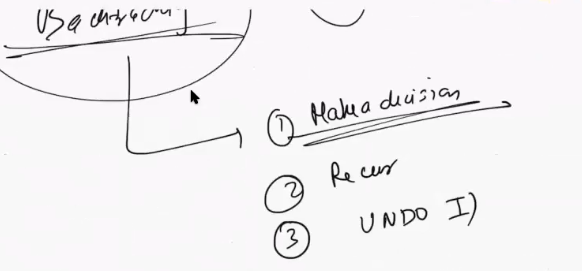
****

**You have O(n) and outside u have N you have O{n^2) atleast**

**Backtracking**

**A whenever a question asks you to explore all the possibilities. That mean for every step you have a choice to make.**

****

**m**

**Give me the permutation of a sentence is also a backtracking problem**

**Backtracking is 3 steps**

**Step 1: make a decision**

**Step 2: recurr after the decision**

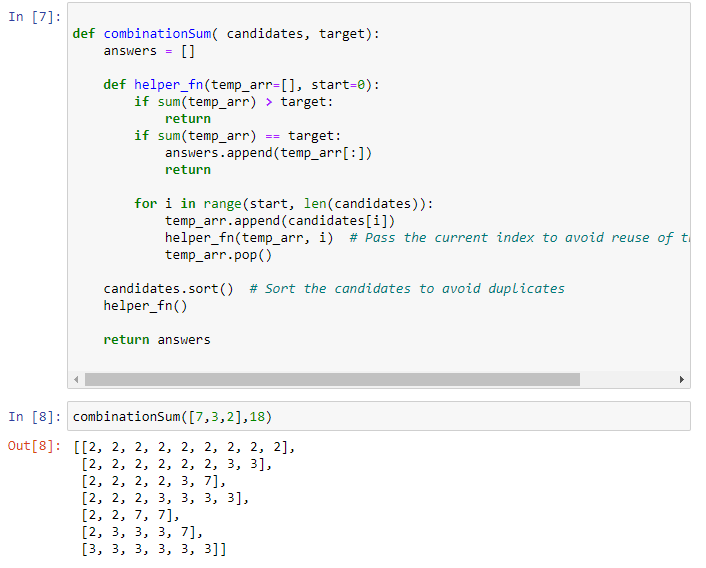
**Step 3: UNDO step 1**

**Question Combinattion Sum 1 : Striver SDE**

****

**Above code was my try which gives wrong output as [3,3,3,3,3,3] is not present**

**The correct code is**

****

**Remember the index thing will only work if you do the candidate.sort() otherwise it will not work**