(class 4) 25-03-2023

Backtracking

Date.....

Ques Generate Paranthesis.
(Leetcode Ques NO. 22)

Given 'n' pairs of paranthesis, write a function to generate all combina-tions of well-formed paranthesis.

Example :-

Input, n = 3
Output, ["((()))", "(()())", "(())()", "()(())", "()()()"]

Example :-

Input, n = 4
Output, ["(((())))", "((()()))", "((())())", "((()))()", "(()(()))", "(()()())", "(())(())", "(())()()", "()((()))", "()(()())", "()(())()", "()()(())", "()()()()"]

Approach :- The approach is that, we have given the value of paranthesis, i.e., 'n' & we have the same number of open & same number of close brackets, i.e., if n = 4, then openBrackets = 4 and closeBrackets = 4.

Date.....

we create one output string & vector of string to store the multiple possible outputs.
when we input open & close brackets then number of count will also be decremented.
So, // base case is when openBracket & closeBracket both is equal to 0 then it means we found the paranthesis in output string & we push_back the string in ans vector, i.e.,

```
if(openBrackets == 0 and
                closeBrackets == 0)
{
    ans.push_back(output);
    return;
}
```

Now, we only include the string, i.e., openBracket if it is available, means if its count is greater than 0.
So, what we do is we include 1 openBracket in output string & recursively call for next brackets with output string & decrement the count of openBracket, then the Backtracking step is we pop_back the previously pushed backet, i.e.,

Date.....

```
if(openBrackets > 0)
{
    output.psh_back('(');
    generateBrackets(n, ans,
        output, openBrackets-1,
            closeBrackets);
    output.pop_back();
}
```

Now, we include the closeBracket in the string. But there is a condition to include closeBracket, the condition is that if closeBracket count is greater than openBracket count, then only we push_back closing bracket.

Agr closingBracket == openBracket hoga, it means string already valid hai, is case me closingBracket extra add krenge to string invalid ho jayegi.
Agr closingBracket < openBracket hoga, means string already invalid hai to hum extra closingBracket add nahi krenge.

Date.....

```
if (closeBrackets > openBracket)
{
    output.push_back (')');
    generateBrackets (n, ans,
        output, openBrackets,
            closeBrackets - 1);
    output.pop_back();
}
```

The above are the only 3 condition which we use to generate Paranthesis.

code:-

```
#include<bits/stdc++.h>
using namespace std;
void generateBrackets(int n, vector
  <string> &ans, string output,
  int openBracket, int closeBracket)
{
   // base case
   if(openBracket == 0 and closeBracket
                          == 0)
   {
      ans.push_back(output);
      return;
   }
   // include openBracket
   if(openBracket > 0)
   {
      output.push_back('(');
      // recursive call
      generateBrackets(n, ans, output
        , openBracket-1, closeBracket);
      // backtracking
      output.pop_back();
   }
   // include closeBracket
   if(closeBracket > openBracket)
   {
      output.push_back(')');
      // recursive call
      generateBrackets(n, ans, output,
        openBracket, closeBracket-1);
```

Teacher's Sign .....

Date.....

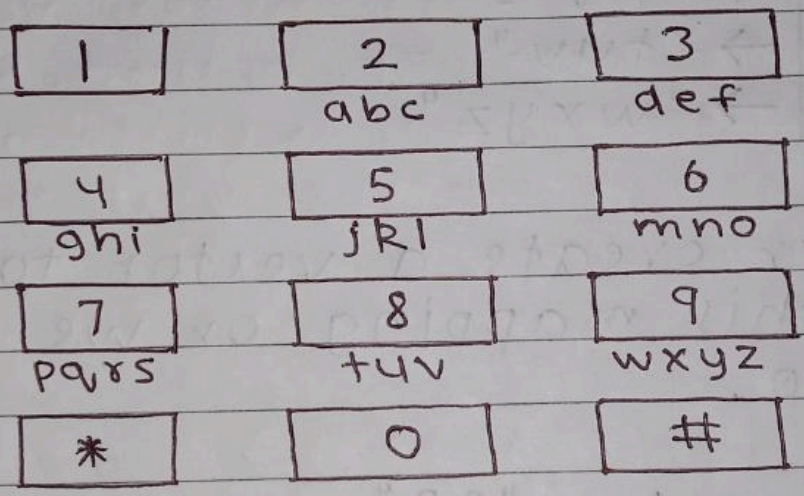
```
        // backtracking
        output.pop_back();
    }
}
vector<string> generateParanthesis
                                (int n).
{
    vector<string> ans;
    string output = "";
    int openBracket = n;
    int closeBracket = n;
    generateBrackets(n, ans,
      output, openBracket, closeBra
        -cket);
    return ans;
}
int main()
{
    int n = 4;
    vector<string> ans(generatepara
                        -nthesis(n));
    for(int i = 0; i < ans.size(); i++)
    {
        cout << ans[i] << " ";
    }
    cout << endl;
    return 0;
}
```

(most asked Question in "Deshaw" & "Arcesium")

Date.....

Ques Letter combination of a phone number. (leetcode Ques No. 17)

we've given a string containing digits from 2 to 9, like - 423, 129, etc. we have to return all possible letter combination that the number could represent.



For example:-

Input: digits = "23"
output: ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"]

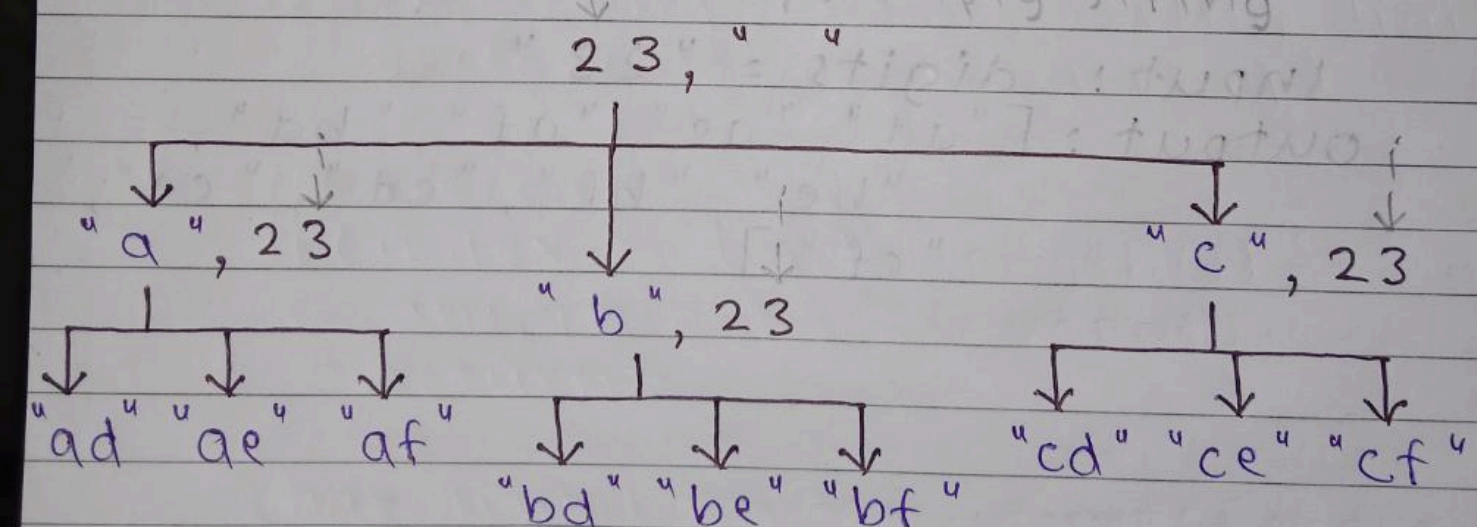
In this problem, we have mapping of numbers with characters, i.e;

2	→ "abc"
3	→ "def"
4	→ "ghi"
5	→ "jkl"
6	→ "mno"
7	→ "pqrs"
8	→ "tuv"
9	→ "wxyz"

mapping

We can either create a vector to represent this mapping or we can use map also.

Pattern:- Input :- "23"



this is include exclude pattern type problem

Date.....

Approach:- what we do is, first we map the numbers with letters, as 2 is mapped with "abc" & 7 is mapped with "pqrs". we use vector of strings to map numbers with characters, i.e.,

```
vector<string> mapping(10);
mapping[2] = "abc";
mapping[3] = "def";
mapping[4] = "ghi";
mapping[5] = "jkl";
mapping[6] = "mno";
mapping[7] = "pqrs";
mapping[8] = "tuv";
mapping[9] = "wxyz";
```

Now, we use these mappings to generate all possible combinations of the given digits.

Next step, we take first character from the given digit string, i.e., suppose given digit is "23", then we first pick the first character, i.e., '2'. convert this '2' from character to integer, & then fetch the mapped characters to this integer. Like, our integer is 2 & mapped characters with this 2 are "abc", & save these characters in the string.

Teacher's Sign.....

Date.....

Now, we run a loop on this string, & print all possible combinations with the string & recursively call the function for next number of the string, i.e., first we call the function for '2' in "23", now we are calling for '3' in "23" for the string & do the same for '3' also.
when we call for next index value then index is out of bound. So, we return from the call, & push all the combinations in the vector.

code:-

```
#include<bits/stdc++.h>
using namespace std;
void solve(vector<string> &ans, int
  index, string output, string digits,
    vector<string> &mapping)
{
  //base case
  if(index >= digits.length()) {
     ans.push_back(output);
     return;
  }
  char digitcharacter = digits[index];
  int digitInteger = digitcharacter
                        -'0';
  string value = mapping[digitInte
                        -ger];
```

Date.....

```
    for(int i = 0; i < value.length();
                                    i++)
    {
        char ch = value[i];
        output.push_back(ch);
        solve(ans, index + 1, output,
              digits, mapping);
        output.pop_back();
    }
}
vector<string> letterCombinations(stri
                                -ng digits)
{
    vector<string> ans;
    if(digits.length() == 0)
    {
        return ans;
    }
    int index = 0;
    string output = "";
    vector<string> mapping(10);
    mapping[2] = "abc";
    mapping[3] = "def";
    mapping[4] = "ghi";
    mapping[5] = "jkl";
    mapping[6] = "mno";
    mapping[7] = "pqrs";
    mapping[8] = "tuv";
    mapping[9] = "wxyz";
    solve(ans, index, output, digits,
                              mapping);

    return ans;
```

Teacher's Sign .....

Date.....

```
int main()
{
    string digits = "23";
    vector<string> ans (lettercombina
                        -tions(digits));
    cout << endl;
    for(int i=0; i<ans.size(); i++)
    {
        cout << ans[i] << " ";
    }
    return 0;
}
```