DOMAIN WINTER WINNING CAMP

StudentName: Divya Sharma **UID:**22BCS14192
Branch:CSE **Section/Group:**FL_IOT-603/B

Day 9 :BackTracking

Very Easy:

1. Generate Numbers with a Given Sum

Generate all numbers of length n whose digits sum up to a target value sum, The digits of the number will be between 0 and 9,and we will generate combinations of digits such that their sum equals the target.

Example 1:
Input:n=2andsum=5 Output:
14 23 32 41 50

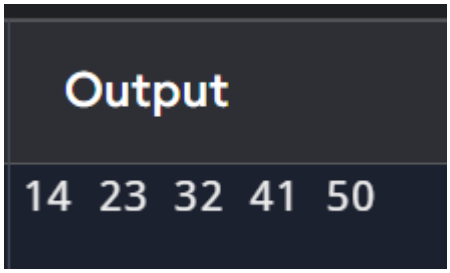
Example 2:
Input:n = 3 and sum = 5
Output:104113122131140203212221230302311320401410500

Constraints:
1<=n<= 9:Thenumberofdigitsmustbebetween1and 9.
1<=sum<=100:Thesumofthedigitsmustbebetween1and100. The first digit cannot be zero if n > 1.

CODE:

```
#include <iostream>
#include <vector>
usingnamespacestd;

Void generateNumbers(intn,intsum,stringcurrent,vector<string>&result){ if (n
   == 0 && sum == 0) {
      result.push_back(current);
      return;
   }
   if (n == 0 || sum < 0) return;

   intstart=current.empty()?1:0; for
   (int i = start; i <= 9; ++i) {
      generateNumbers(n-1,sum-i,current+to_string(i),result);
```

```
    }
}

intmain() {
    int n = 2, sum = 5;
    vector<string>result;
    generateNumbers(n,sum,"",result); for
    (const string &num : result) {
        cout<< num <<"";
    }
    return0;
}
```

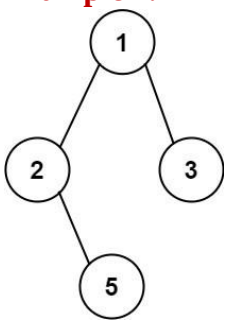


Easy:

2. Binary Tree Paths

Given the root of a binary tree,return all root-to-leaf paths in any order.
A leaf is a node with no children.

Example 1:

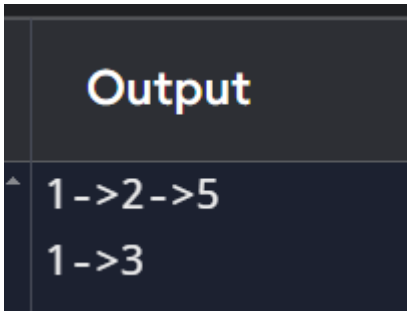


Input:root=[1,2,3,null,5]Output:["1->2->5","1->3"]
Example 2:
Input:root=[1]Output:["1"]

Constraints:
Thenumberofnodesinthetreeisintherange[1,100].
-100<= Node.val<= 100

CODE:

```
#include <iostream>
#include <vector>
#include <string>
usingnamespacestd;

structTreeNode{
    int val;
    TreeNode*left,*right;
    TreeNode(intx):val(x),left(nullptr),right(nullptr){}
};

voiddfs(TreeNode*root,stringpath,vector<string>&paths){ if
    (!root) return;
    path+=to_string(root->val);
    if(!root->left&&!root->right){
        paths.push_back(path);return;
    }
    path+="->";
    dfs(root->left, path, paths);
    dfs(root->right,path,paths);
}

vector<string>binaryTreePaths(TreeNode*root){
    vector<string> paths;
    dfs(root,"",paths);
    return paths;
}

intmain() {
    TreeNode*root=newTreeNode(1);
    root->left = new TreeNode(2);
    root->right= newTreeNode(3);
    root->left->right=newTreeNode(5);

    vector<string>result=binaryTreePaths(root);
    for (const string &path : result) {
        cout<< path<<endl;
    }
    return0;
}
```



Medium:

3. Combinations

Given two integers nandk ,return all possible combinations of k numbers chosen from the range [1, n].

You may return the answer in any order.

Example 1:

Input: n = 4, k = 2
Output:[[1,2],[1,3],[1,4],[2,3],[2,4],[3,4]]
Explanation:There are 4 choose 2=6 total combinations.
Note that combinations are unordered, i.e. ,[1,2]and[2,1]are considered to be the same combination.
Example 2:

Input:n=1,k=1
Output: [[1]]
Explanation:Thereis1choose1=1totalcombination.

Constraints:

1 <= n <= 20
1<=k<=n CODE:

```
#include <iostream>
#include <vector>
usingnamespacestd;

voidcombineHelper(intstart,intn,intk,vector<int>&current,vector<vector<int>>&result
) {
    if (k == 0) {
        result.push_back(current);
```

```
        return;
    }
    for(inti=start;i<=n;++i){
        current.push_back(i);
        combineHelper(i+1,n,k-1,current,result);
        current.pop_back();
    }
}

vector<vector<int>>combine(intn,intk){ vector<vector<int>>
    result;
    vector<int> current;
    combineHelper(1,n,k,current,result);
    return result;
}

intmain() {
    int n = 4, k = 2;
    vector<vector<int>>result=combine(n,k); for
    (const auto &comb : result) {
        cout<<"[";
        for(intnum:comb)cout<<num<<""; cout
        <<"]"<< endl;
    }
    return0;
}
```

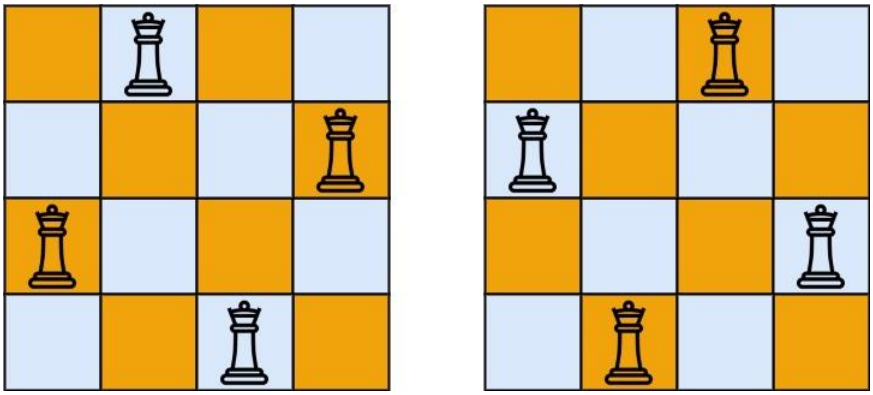
```
Output
[1 2 ]
[1 3 ]
[1 4 ]
[2 3 ]
[2 4 ]
[3 4 ]
```

Hard:

4. N-QueensII

Then-queenspuzzleistheproblemofplacingnqueensonannxnchessboardsuch that no two queens attack each other.
Givenanintegern,returnthenumberofdistinctsolutionstothen-queens puzzle.

Example 1:



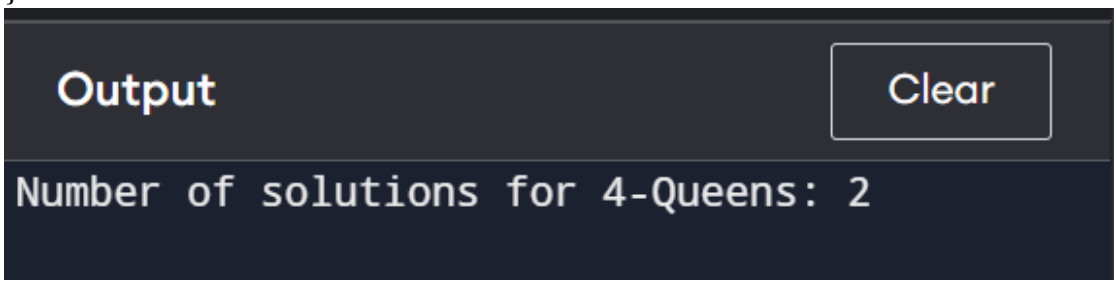
Input:n=4
Output: 2
Explanation:Therearetwodistinctsolutionstothe4-queenspuzzleas shown.

Example 2:
Input:n=1
Output: 1

Constraints:
1<=n<=9 CODE:

```
#include <iostream>
#include <vector>
usingnamespacestd;

voidsolve(introw,intn,vector<int>&cols,vector<int>&diags1,vector<int>&diags2,
int &count) {
   if(row==n) {
      ++count;
      return;
   }
   for(int col =0; col <n; ++col) {
      if(cols[col]||diags1[row-col+n-1]||diags2[row+col])continue; cols[col]
      = diags1[row - col + n - 1] = diags2[row + col] = 1; solve(row + 1, n,
      cols, diags1, diags2, count);
```

```
        cols[col]=diags1[row-col+n-1]=diags2[row+col]=0;
    }
}

inttotalNQueens(intn){
    vector<int>cols(n,0),diags1(2*n-1,0),diags2(2*n-1,0); int count
    = 0;
    solve(0,n,cols,diags1,diags2,count);
    return count;
}

intmain(){ int
    n = 4;
    cout<<"Numberofsolutionsfor"<<n<<"-Queens:"<<totalNQueens(n)<<endl;
    return0;
}
```



VeryHard:

5. WordLadderII

Atransformation sequence from word beginWord to word endWord using a dictionarywordListisasequenceofwordsbeginWord->s1->s2->...->sksuch that:
Everyadjacentpairofwordsdiffersbyasingle letter.
Everysifor1<=i<=kisinwordList.NotethatbeginWorddoesnotneedtobe in wordList.
sk== endWord
Giventwowords,beginWordandendWord,andadictionarywordList,return all theshortesttransformationsequencesfrombeginWordtoendWord,oranemptylistif no such sequence exists. Each sequence should be returned as a list of the words[beginWord,s1,s2,..., sk].

Example 1:
Input:beginWord="hit",endWord="cog",wordList= ["hot","dot","dog","lot","log","cog"]
Output:[["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]

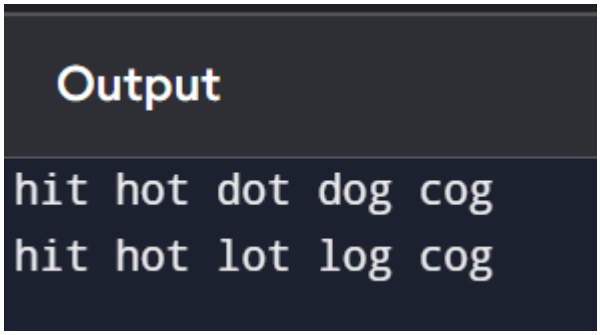
Explanation:Thereare2shortesttransformationsequences: "hit" ->"hot" ->"dot" ->"dog" ->"cog"
"hit"->"hot"->"lot"->"log"->"cog"

Example 2:
Input:beginWord="hit",endWord="cog",wordList= ["hot","dot","dog","lot","log"]
Output:[]
Explanation:TheendWord"cog"isnotinwordList,thereforethereisnovalid transformation sequence.

Constraints:
1 <= beginWord.length <= 5
endWord.length == beginWord.length
1 <= wordList.length <= 500
wordList[i].length== beginWord.length
beginWord,endWord,andwordList[i]consistoflowercaseEnglishletters. beginWord != endWord
Allthewordsinwordlistare unique.
Thesumofallshortesttransformationsequencesdoesnotexceed105. CODE:

```
#include <iostream>
#include <vector>
#include<unordered_set>
#include <queue>
usingnamespacestd;

vector<vector<string>>findLadders(stringbeginWord,stringendWord,
vector<string>&wordList) {
    unordered_set<string>dict(wordList.begin(),wordList.end());
    vector<vector<string>> result;
    if(dict.find(endWord)==dict.end())returnresult;

    queue<vector<string>>paths;
    paths.push({beginWord});
    intlevel=1,minLevel=INT_MAX;
    unordered_set<string> visited;

    while (!paths.empty()) {
        vector<string>path=paths.front();
        paths.pop();
        if(path.size()>level){
            for(conststring&word:visited)dict.erase(word); visited.clear();
            level=path.size();
```

```
            if(level>minLevel)break;
        }

        stringlast = path.back();
        for(inti=0;i<last.size();++i){ string
            next = last;
            for(charc='a';c<='z';++c){ next[i]
                = c;
                if (!dict.count(next)) continue;
                visited.insert(next);
                vector<string>newPath=path;
                newPath.push_back(next);
                if (next == endWord) {
                    result.push_back(newPath);
                    minLevel = level;
                }else{
                    paths.push(newPath);
                }
            }
        }
    }
    returnresult;
}

intmain() {
    stringbeginWord="hit",endWord="cog";
    vector<string> wordList = {"hot", "dot", "dog", "lot", "log", "cog"};
    vector<vector<string>>result=findLadders(beginWord,endWord,wordList); for
    (const auto &path : result) {
        for(conststring&word:path){
            cout << word <<"";
        }
        cout<<endl;
    }
    return0;
}
```


Output
hit hot dot dog cog
hit hot lot log cog