60. Permutation Sequence

Solv

Hard Topics Companies

The set `[1, 2, 3, ..., n]` contains a total of `n!` unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for `n = 3`:

1. `"123"`
2. `"132"`
3. `"213"`
4. `"231"`
5. `"312"`
6. `"321"`

Given `n` and `k`, return the `k^th` permutation sequence.

Example 1:

```
Input: n = 3, k = 3
Output: "213"
```

Example 2:

```
Input: n = 4, k = 9
Output: "2314"
```

Example 3:

```
Input: n = 3, k = 1
Output: "123"
```

Constraints:

- `1 <= n <= 9`

Code

Java Auto

```java
class Solution {
    public String getPermutation(int n, int k) {
      int[] f=new int[n];
      f[0]=1;
      for(int i=1;i<n;i++) f[i]=f[i-1]*i;
      List<Integer> l=new ArrayList<>();
      for(int i=1; i<=n; i++) l.add(i);
      k--;
      StringBuilder sb = new StringBuilder();

      for(int i=1; i<=n; i++) {
        int index = k/f[n-i];
        sb.append(l.get(index));
        k=k%f[n-i];
        l.remove(index);  //used, so remove it as i can't be reused again
      }
      return sb.toString();
    }
}
```

Saved to local

Testcase Test Result

Description Editorial Solutions Submissions

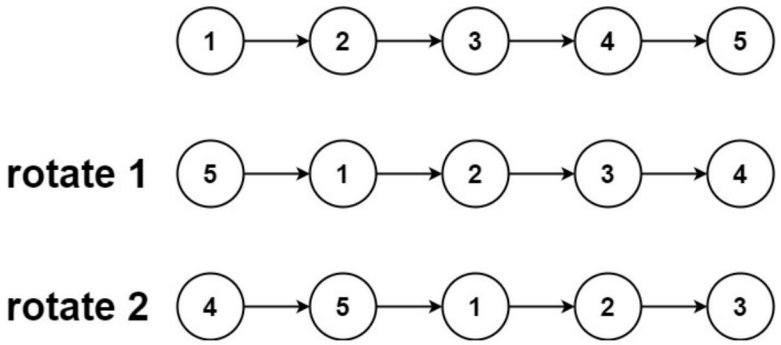
61. Rotate List

Solv

Medium Topics Companies

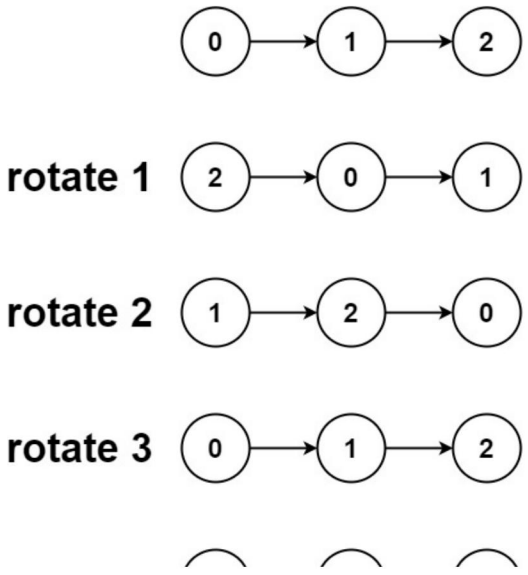
Given the `head` of a linked list, rotate the list to the right by `k` places.

Example 1:




```
Input: head = [1,2,3,4,5], k = 2
Output: [4,5,1,2,3]
```

Example 2:



9K 43

Code

Java Auto

```java
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode(int x) { val = x; }
 * }
 */
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
      if(head==null) return null;
      ListNode cur=head;
      int n=1;
      while(cur.next!=null) {
        n++;
        cur=cur.next;
      }
      k=k%n;
      cur.next=head;
      for(int i=0; i<n-k; i++) {
        cur=cur.next;
      }
      head=cur.next;
      cur.next=null;
      return head;
    }
}
```

Description Editorial Solutions Submissions

62. Unique Paths

Solv

Medium Topics Companies

There is a robot on an `m x n` grid. The robot is initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries to move to th **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

Given the two integers `m` and `n`, return *the number of possible unique paths that the robot can take to reach the bottom-right corner*.

The test cases are generated so that the answer will be less than or equal to `2 * 10^9`.

Example 1:


```
Input: m = 3, n = 7
Output: 28
```

Example 2:

```
Input: m = 3, n = 2
Output: 3
Explanation: From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:
1. Right -> Down -> Down
2. Down -> Down -> Right
3. Down -> Right -> Down
```

Constraints:

Code

Java Auto

```java
class Solution {
    public int uniquePaths(int m, int n) {

        int[] dp = new int[n];
        Arrays.fill(dp, 1);
        for (int i = 1; i < m; ++i) {
            for (int j = 1; j < n; ++j) {
                dp[j] += dp[j - 1];
            }
        }
        return dp[n - 1];

    }
}


// 这道题其实还有另一种很数学的解法，参见网友 Code Ganker 的博客，实际相当于机器人总共走了 m + n - 2步，其中 m - 1 步向右走，n - 1 步向下走，那么总共不同的方法个数就相当于在步数里面 m - 1 和 n - 1 中较小的那个数的取法，实际上是一道组合数的问题，写出代码如下：
//     int uniquePaths(int m, int n) {
//         double num = 1, denom = 1;
//         int small = m > n ? n : m;
//         for (int i = 1; i <= small - 1; ++i) {
//             num *= m + n - 1 - i;
//             denom *= i;
//         }
//         return (int)(num / denom);
//     }
```

Saving... Ln

Testcase Test Result

Description Editorial Solutions Submissions

63. Unique Paths II

Solv

Medium Topics Companies Hint

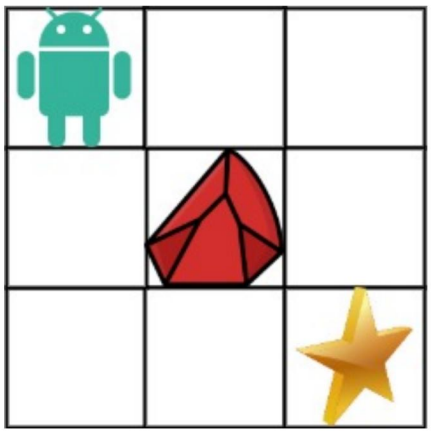
You are given an `m x n` integer array `grid`. There is a robot initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries move to the **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

An obstacle and space are marked as `1` or `0` respectively in `grid`. A path that the robot takes cannot include **any** square that is an obstacle.

Return *the number of possible unique paths that the robot can take to reach the bottom-right corner*.

The testcases are generated so that the answer will be less than or equal to `2 * 10^9`.

Example 1:



```
Input: obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
Output: 2
Explanation: There is one obstacle in the middle of the 3x3 grid above.
There are two ways to reach the bottom-right corner:
1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right
```

Example 2:

Code

Java Auto

```java
class Solution {
    public int soldp2d(int[][] obstacleGrid) {
      int rows=obstacleGrid.length;
      int cols=obstacleGrid[0].length;
      int[][] dp = new int[rows+1][cols+1];
      if(obstacleGrid[0][0]==0)
        dp[0][1]=1;
      for(int i=1; i<=rows; i++) {
        for(int j=1; j<=cols; j++) {
          if(obstacleGrid[i-1][j-1]!=0)
            dp[i][j]=0;
          else
            dp[i][j]=dp[i-1][j]+dp[i][j-1];
        }
      }

      return dp[rows][cols];
    }

    public int soldp1d(int[][] obstacleGrid) {
      int rows=obstacleGrid.length;
      int cols=obstacleGrid[0].length;
      int[] dp = new int[cols+1];
      if(obstacleGrid[0][0]==0)
        dp[1]=1;
      for(int i=1; i<=rows; i++) {
        for(int j=1; j<=cols; j++) {
          if(obstacleGrid[i-1][j-1]!=0)
            dp[j]=0;
          else
            dp[j]+=dp[j-1];
        }
      }

      return dp[cols];
    }

    public int uniquePathsWithObstacles(int[][] obstacleGrid) {
      //return soldp2d(obstacleGrid);
      return soldp1d(obstacleGrid);
    }


}
```

Saved to local Ln

Testcase Test Result

64. Minimum Path Sum

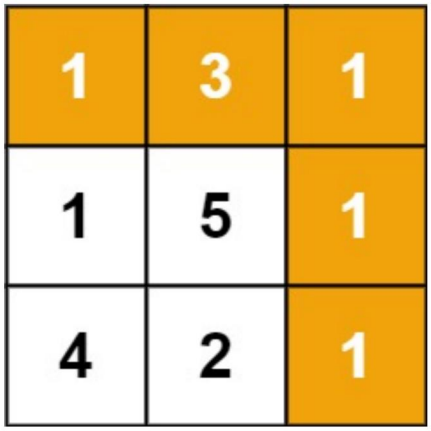
Solv

Medium Topics Companies

Given a `m x n` `grid` filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all num along its path.

Note: You can only move either down or right at any point in time.

Example 1:




```
Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.
```

Example 2:

```
Input: grid = [[1,2,3],[4,5,6]]
Output: 12
```

Constraints:

- `m == grid.length`
- `n == grid[i].length`

Code

Java Auto

```java
class Solution {
    public int minPathSum(int[][] grid) {
      int m = grid.length;
      int n = grid[0].length;
      int[][] dp = new int[m][n];
      dp[0][0]=grid[0][0];

      for(int i=1;i<n;i++) dp[0][i]=dp[0][i-1]+grid[0][i];
      for(int i=1;i<m;i++) dp[i][0]=dp[i-1][0]+grid[i][0];

      for(int i=1; i<m; i++){
        for(int j=1; j<n; j++) {
          dp[i][j]=Math.min(dp[i-1][j], dp[i][j-1])+grid[i][j] ;

        }

      }
      return dp[m-1][n-1];
    }
}
```

Saved to local Ln

Testcase Test Result

65. Valid Number

Solv

Hard Topics Companies

A **valid number** can be split up into these components (in order):

1. A **decimal number** or an **integer**.
2. (Optional) An `'e'` or `'E'`, followed by an **integer**.

A **decimal number** can be split up into these components (in order):

1. (Optional) A sign character (either `'+'` or `'-'`).
2. One of the following formats:
    1. One or more digits, followed by a dot `'.'`.
    2. One or more digits, followed by a dot `'.'`, followed by one or more digits.
    3. A dot `'.'`, followed by one or more digits.

An **integer** can be split up into these components (in order):

1. (Optional) A sign character (either `'+'` or `'-'`).
2. One or more digits.

For example, all the following are valid numbers: `["2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6`
`1", "53.5e93", "-123.456e789"]`, while the following are not valid numbers: `["abc", "1a", "1e", "e3", "99e2.5", "--6", "-+`
`"95a54e53"]`.

Given a string `s`, return `true` *if* `s` *is a* ***valid number***.

Example 1:

```
Input: s = "0"
Output: true
```

Example 2:

```
Input: s = "e"
Output: false
```

Code

Java
Auto

```java
class Solution {
    public boolean isNumber(String s) {
      boolean has_e = false;
      boolean has_num = false;
      boolean has_dot = false;
      boolean has_num_e = true;  //number after e
      boolean has_sign = false;
      s=s.trim();
      for(int i=0; i<s.length(); i++) {
        if(s.charAt(i)==' ')  {
          //if(i<s.length()-1 && s.charAt(i+1)!=' ' && (has_e || has_num || has_dot || has_sign) ) return false;
          return false;
        }else if(s.charAt(i)=='e' || s.charAt(i)=='E') {
          if(has_e || !has_num) return false;
          has_e=true;
          has_num_e=false;
        }else if(s.charAt(i)=='.') {
          if(has_dot || has_e) return false;
          has_dot=true;
        }else if(s.charAt(i)=='+' || s.charAt(i)=='-') {
          if(i>0 && s.charAt(i-1)!='e' && s.charAt(i-1)!='E' && s.charAt(i-1)!=' ') return false;
          has_sign=true;
        }else if('0'<=s.charAt(i) && s.charAt(i)<='9') {

          has_num=true;
          if(has_e) has_num_e=true;
        }else return false;

      }//for
      return has_num && has_num_e;
    }
}
```

Saved to local

Testcase
Test Result

66. Plus One

Solv

Easy Topics Companies

You are given a **large integer** represented as an integer array `digits`, where each `digits[i]` is the `i^th` digit of the integer. The digi are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading `0`'s.

Increment the large integer by one and return *the resulting array of digits*.

Example 1:

```
Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].
```

Example 2:

```
Input: digits = [4,3,2,1]
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
Incrementing by one gives 4321 + 1 = 4322.
Thus, the result should be [4,3,2,2].
```

Example 3:

```
Input: digits = [9]
Output: [1,0]
Explanation: The array represents the integer 9.
Incrementing by one gives 9 + 1 = 10.
Thus, the result should be [1,0].
```

Constraints:

- `1 <= digits.length <= 100`
- `0 <= digits[i] <= 9`

Code

Java
Auto

```java
class Solution {
    public int[] plusOne(int[] digits) {
      int n = digits.length;
      for(int i=n-1; i>=0; i--) {
          if(digits[i] < 9) {
              digits[i]++;
              return digits;
          }

         digits[i] = 0;
    }

    int[] newNumber = new int [n+1];
    newNumber[0] = 1;

    return newNumber;
    }
}
```

Saved to local

Testcase
Test Result

67. Add Binary

Solv

Easy Topics Companies

Given two binary strings `a` and `b`, return *their sum as a binary string*.

Example 1:

```
Input: a = "11", b = "1"
Output: "100"
```

Example 2:

```
Input: a = "1010", b = "1011"
Output: "10101"
```

Constraints:

- `1 <= a.length, b.length <= 10^4`
- `a` and `b` consist only of `'0'` or `'1'` characters.
- Each string does not contain leading zeros except for the zero itself.

Seen this question in a real interview before? 1/4

Yes No

Accepted 1.3M Submissions 2.5M Acceptance Rate 52.9%

Topics

Companies

Similar Questions

8.9K 97

Code

Java Auto

```java
class Solution {
    public String addBinary(String a, String b) {
      StringBuilder sb = new StringBuilder();
      int i=a.length()-1;
      int j=b.length()-1;
      int carry=0;
      while(i>=0 || j>=0) {
        //System.out.println(""+i+" "+j+" "+ carry);
        int sum=carry;
        if(i>=0) sum+=a.charAt(i--)-'0';
        if(j>=0) sum+=b.charAt(j--)-'0';
        carry = sum/2;
        sb.append(sum%2);
      }
      if(carry>0)
        sb.append(carry);
      return sb.reverse().toString();
    }
}
```

Saved to local

Testcase Test Result

68. Text Justification

Solv

Hard Topics Companies

Given an array of strings `words` and a width `maxWidth`, format the text such that each line has exactly `maxWidth` characters and is ful (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces `' '` when necessary so that each line has exactly `maxWidth` characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly betw words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left-justified, and no extra space is inserted between words.

Note:

- A word is defined as a character sequence consisting of non-space characters only.
- Each word's length is guaranteed to be greater than `0` and not exceed `maxWidth`.
- The input array `words` contains at least one word.

Example 1:

```
Input: words = ["This", "is", "an", "example", "of", "text", "justification."], maxWidth = 16
Output:
[
   "This    is    an",
   "example  of text",
   "justification.  "
]
```

Example 2:

```
Input: words = ["What","must","be","acknowledgment","shall","be"], maxWidth = 16
Output:
[
  "What   must   be",
  "acknowledgment  ",
  "shall be        "
```

Code

Java Auto

```java
class Solution {
    public List<String> fullJustify(String[] words, int maxWidth) {
      List<String> ret = new ArrayList<>()    ;
      int i=0;
      while(i<words.length) {
        int start=i;
        int cnt=words[start].length();
        i++;
        while(i<words.length && cnt+words[i].length()+1<=maxWidth) {
          cnt+=words[i++].length()+1;
        }//masWidth
        int n = i-start;
        StringBuilder sb = new StringBuilder();
        if(n==1 || i==words.length) {
          for(int j=start; j<i; j++) {
              sb.append(words[j]);
              sb.append(' ');
          }
          sb.deleteCharAt(sb.length()-1);
          for(int j=sb.length(); j<maxWidth; j++)
            sb.append(' ');
        }else {

          int rest=maxWidth-cnt;
          int rest1=rest/(n-1);
          int rest2=rest%(n-1);
          //System.out.println("start:"+start+" i:"+i+" cnt:"+cnt +" rest1:"+rest1+" rest2:"+rest2);
          for(int j=start; j<i; j++) {
            sb.append(words[j]);
            if(j<i-1) {
              sb.append(' ');
              for(int k=1; k<=rest1; k++) {
                sb.append(' ');
              }
              if(rest2-->0)
                sb.append(' ');
            }
          }
        }//adjust
        ret.add(sb.toString());
      }//words
      return ret;
    }
}
```

Saving... Ln

Testcase Test Result

69. Sqrt(x)

Solv

Easy Topics Companies Hint

Given a non-negative integer `x`, return *the square root of* `x` *rounded down to the nearest integer*. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

- For example, do not use `pow(x, 0.5)` in c++ or `x ** 0.5` in python.

Example 1:

```
Input: x = 4
Output: 2
Explanation: The square root of 4 is 2, so we return 2.
```

Example 2:

```
Input: x = 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integ
2 is returned.
```

Constraints:

- `0 <= x <= 2^31 - 1`

Seen this question in a real interview before? 1/4

Yes No

Accepted 1.7M Submissions 4.5M Acceptance Rate 38.1%

Topics

7.6K 142

Code

Java Auto

```java
class Solution {
    public int mySqrt(int x) {
      if(x<=1) return x;
      long left = 1;
      long right = x;
      while (left<=right) {
        long mid = left + (right - left) / 2;
        if (mid * mid == x)
          return (int)mid;
        if (mid * mid < x)
          left = mid + 1;
        else
          right = mid - 1;
      }

      //now left == right + 1;
      return (int)right;
    }
}
```

Saved to local

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public int climbStairs(int n) {
      if(n<=2) return n;
      int[] dp = new int[n+1];
      dp[1]=1;
      dp[2]=2;
      for(int i=3; i<=n; i++) {
        dp[i]=dp[i-1]+dp[i-2];
      }
      return dp[n];
    }
}
```

Saved to local

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public String simplifyPath(String path) {
      String[] strs=path.split("/");
      Deque<String> list=new LinkedList<>();
      for( String s : strs){
        if(s.equals(".") || s.length()==0) continue;
        if(s.equals("..")) {
          if(list.size()>0) {
            list.removeLast();
          }
          continue;
        }
        list.offerLast(s);

      }
      return "/" + String.join("/", list);
    }
}
```

Saved to local

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public int minDistance(String word1, String word2) {
      int n1=word1.length();
      int n2=word2.length();
      int[][] dp = new int[n1+1][n2+1];

      for(int i=0; i<=n2; i++) dp[0][i]=i;
      for(int i=0; i<=n1; i++) dp[i][0]=i;
      for(int i=1; i<=n1; i++) {
        for(int j=1; j<=n2; j++) {
          if(word1.charAt(i-1)==word2.charAt(j-1)) {
            dp[i][j]=dp[i-1][j-1];
          }
          else {

            dp[i][j]=Math.min(Math.min(dp[i][j-1],dp[i-1][j]), dp[i-1][j-1])+1;
          }
        }
      }
      return dp[n1][n2];
    }
}
```

Saved to local

Testcase Test Result

Code

Java
Auto

```java
class Solution {
    public void setZeroes(int[][] matrix) {
      boolean row0=false;
      boolean col0=false;
      int m = matrix.length;
      int n = matrix[0].length;
      //use 1st row and 1st col as indicator,
      //so need save if row0 or col0 is true
      for(int i=0; i<n; i++) if(matrix[0][i]==0) row0=true;
      for(int i=0; i<m; i++) if(matrix[i][0]==0) col0=true;
      for(int i=1; i<m; i++) {
        for(int j=1; j<n; j++) {
          if(matrix[i][j]==0) {
            matrix[i][0]=0;
            matrix[0][j]=0;
          }
        }
      }
      for(int i=1; i<m; i++) {
        for(int j=1; j<n; j++) {
          if(matrix[i][0]==0 || matrix[0][j]==0) {
            matrix[i][j]=0;
          }
        }
      }
      if(row0==true) for(int i=0; i<n; i++) matrix[0][i]=0;
      if(col0==true) for(int i=0; i<m; i++) matrix[i][0]=0;

    }
}
```

Saved to local

Ln

Testcase
Test Result

Code

Java Auto

```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
      if(matrix.length==0 || matrix[0].length==0) return false;
      int m = matrix.length;
      int n = matrix[0].length;
      int left=0; int right=m*n-1;
      while(left<=right) {
        int mid = (left+right)/2;
        int row = mid/n;
        int col = mid%n;
        int val = matrix[row][col];
        if(val == target) return true;
        if(val<target) left=mid+1;
        else right=mid-1;
      }
      //return matrix[left/n][left%n]==target;
      //now left == right + 1
      return false;
    }
}
```

Saved to local

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public void sortColors(int[] nums) {
      int red=0;
      int blue=nums.length-1;
      for(int i=0; i<=blue; i++) {

        if(nums[i]==0) {
          int tmp = nums[i];
          nums[i]=nums[red];
          nums[red]=tmp;
          red++;
        }else if(nums[i]==2) {
          int tmp=nums[i];
          nums[i]=nums[blue];
          nums[blue]=tmp;

          blue--;
          i--;
        }
      }
    }
}
```

Saved to local Ln

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public String minWindow(String s, String t) {
      if(s == null || s.length() < t.length() || s.length() == 0){
        return "";
      }
      Map<Character, Integer> m=new HashMap<>() ;
      for(int i=0; i<t.length(); i++) {
        m.compute(t.charAt(i), (k,v)-> v==null ? 1 : v+1);
      }
      int left=0;
      int cnt=0;
      int minleft=0;
      int minlen=Integer.MAX_VALUE;
      for(int right=0; right<s.length(); right++) {
        Character c=s.charAt(right);
        if(!m.containsKey(c)) continue;
        if(m.get(c)>0) cnt++;
        m.compute(c, (k,v)->v-1);
        while(cnt==t.length()){
          int len=right-left+1;
          if(len<minlen) {
            minleft=left;
            minlen=len;
          }
          if(m.containsKey(s.charAt(left))) {
            m.compute(s.charAt(left), (k,v)->v+1);
            if(m.get(s.charAt(left)) > 0)
              cnt--;
          }
          left++;

        }//while

      }
      if(minlen>s.length()) return "";
      return s.substring(minleft, minleft+minlen);
    }
}
```

Saved to local Ln

Testcase Test Result

Code

Java
Auto

```java
class Solution {
    public List<List<Integer>> combine(int n, int k) {
      List<List<Integer>> ret = new ArrayList<>();
      helper(ret, new ArrayList<Integer>(), n, k, 1);
      return ret;
    }

    public void helper(List<List<Integer>> ret, List<Integer> cur, int n, int k, int start  ) {
      if(k==0){
        ret.add(new ArrayList<Integer>(cur));
        return;
      }
      for(int i=start; i<=n; i++) {
        cur.add(i);
        helper(ret, cur, n, k-1, i+1);
        cur.remove(cur.size()-1);
      }
    }
}
```

Saved to local

Ln

Testcase
Test Result

Code

Java Auto

```java
class Solution {

    public List<List<Integer>> sol_1(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>();
      ret.add(new ArrayList<Integer>());

      for(int i=0;  i<nums.length; i++) {
        int n=ret.size();
        for(int j=0; j<n; j++) {
          ret.add(new ArrayList<Integer>(ret.get(j)));
          ret.get(ret.size()-1).add(nums[i]);
        }
      }

      return ret;
    }


    public void helper(int[] nums, List<List<Integer>> ret, List<Integer> curList, int start ) {
      ret.add(new ArrayList<Integer>(curList));
      for(int i=start; i<nums.length; i++ ){
        curList.add(nums[i]);
        helper(nums, ret, curList, i+1);
        curList.remove(curList.size()-1);
      }
    }
    public List<List<Integer>> sol_2(int[] nums) {
      List<List<Integer>> ret = new ArrayList<>();
      helper(nums, ret, new ArrayList<Integer>(), 0);
      return ret;
    }

    public List<List<Integer>> subsets(int[] nums) {
       return sol_2(nums);
    }
}
```

Saved to local Ln

Testcase Test Result

Code

Java Auto

```java
class Solution {
    public boolean exist(char[][] board, String word) {
      int m=board.length;
      int n=board[0].length;
      for(int i=0; i<m; i++) {
        for(int j=0; j<n; j++) {
          if(helper(board, i, j, word, 0))
            return true;
        }
      }
      return false;
    }
    public boolean helper(char[][] board, int x, int y, String word, int start) {
      if(start==word.length()) return true;
      if(x<0 || y<0 || x>=board.length || y>=board[0].length) return false;
      if(board[x][y]!=word.charAt(start)) return false;
      char tmp = board[x][y];
      board[x][y]='0';
      if(helper(board, x-1,y, word, start+1)) return  true;
      if(helper(board, x+1,y, word, start+1)) return  true;
      if(helper(board, x,y-1, word, start+1)) return  true;
      if(helper(board, x,y+1, word, start+1)) return  true;
      board[x][y]=tmp;
      return false;
    }
}
```

Saved to local

Testcase Test Result