Algorithm Problems

1021. Remove Outermost Parentheses (https://leetcode.com/problems/remove-outermost-parentheses/)

Easy

A valid parentheses string is either empty (""), "(" + A + ")", or A + B, where A and B are valid parentheses strings, and + represents string concatenation. For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings.

A valid parentheses string S is primitive if it is nonempty, and there does not exist a way to split it into S = A+B, with A and B nonempty valid parentheses strings.

Given a valid parentheses string S, consider its primitive decomposition: S = P_1 + P_2 + ... + P_k, where P_i are primitive valid parentheses strings.

Return S after removing the outermost parentheses of every primitive string in the primitive decomposition of S.

Example 1:

```
Input: "((()))(())"
Output: "()()()"
```

Explanation:

```
The input string is "((()))(())", with primitive decomposition "(()())" + "
(())".
After removing outer parentheses of each part, this is "()()" + "()" = "()()
()".
```

Example 2:

```
Input: "((())())(()(()))"
Output: "()())()(())"
```

Explanation:

```
The input string is "((())())(()(()))", with primitive decomposition "(()())"
+ "(())" + "(()(()))".
After removing outer parentheses of each part, this is "()()" + "()" + "()(())"
= "()()()()(())".
```

Example 3:

```
Input: "()()"
Output: ""
```

Explanation:

```
The input string is "()()", with primitive decomposition "()" + "()".
After removing outer parentheses of each part, this is "" + "" = "".
```

Note:

- S.length <= 10000
- S[i] is "(" or ")"

- S is a valid parentheses string

Performance

- Runtime: 44 ms, faster than 92.83% of Python3 online submissions for Remove Outermost Parentheses.
- Memory Usage: 13.2 MB, less than 100.00% of Python3 online submissions for Remove Outermost Parentheses.

Complexity Analysis

- O(n) in time.
- O(1) in space.

In [1]:
```
class Solution:
    def removeOuterParentheses(self, S: 'str') -> 'str':
        open_count = 0
        primatives = ''
        primative = ''
        for b in S:
            if b == '(': open_count += 1
            if b == ')': open_count -= 1
            primative += b
            if open_count == 0:
                primatives += primative[1:-1]
                primative = ''
        return primatives

my_sol = Solution()
print('should print ()()(): ', my_sol.removeOuterParentheses("((())())()"))
print('should print ()()()()(()): ', my_sol.removeOuterParentheses("((())())(()(()())))"
print('should print : ', my_sol.removeOuterParentheses("()()"))
```

```
should print ()()():  ()()()
should print ()()()()(()):  ()()()()(())
should print :
```

14. Longest Common Prefix (https://leetcode.com/problems/longest-common-prefix/)

Easy

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:

```
Input: ["flower","flow","flight"]
Output: "fl"
```

Example 2:

```
Input: ["dog","racecar","car"]
Output: ""
```

Explanation: There is no common prefix among the input strings.

Note:

All given inputs are in lowercase letters a-z.

Performance

- Runtime: 36 ms, faster than 98.74% of Python3 online submissions for Longest Common Prefix.
- Memory Usage: 12.7 MB, less than 100.00% of Python3 online submissions for Longest Common Prefix.

Complexity Analysis

```
O(nm) in time.
O(1)  in space.
```

In [3]:
```python
from functools import reduce

class Solution:
    def longestCommonPrefix(self, strs: 'List[str]') -> 'str':
        """
        :type strs: List[str]
        :rtype: str
        """
        if strs == []:
            return ''
        if len(strs) == 1:
            return strs[0]
        return reduce(self.lCP, strs)

    def lCP(self, str1, str2):
        # shorter string first
        if len(str2) < len(str1):
            return self.lCP(str2, str1)
        # empty string
        if len(str1) == 0:
            return ''
        lcp = str1
        for idx, char in enumerate(str1):
            if str2[idx] != char:
                lcp = lcp[0:idx]
                break
        return lcp

my_sol = Solution()
print("Should print 'fl':", my_sol.longestCommonPrefix(["flower","flow","flight"]))
print("Should print '':", my_sol.longestCommonPrefix(["dog","racecar","car"]))
print("Should print 'c':", my_sol.longestCommonPrefix(["c","c","c"]))
```

```
Should print 'fl': fl
Should print '':
Should print 'c': c
```

929. Unique Email Addresses (https://leetcode.com/problems/unique-email-addresses/)

Easy

Every email consists of a local name and a domain name, separated by the @ sign.

For example, in alice@leetcode.com (mailto:alice@leetcode.com), alice is the local name, and leetcode.com is the domain name.

Besides lowercase letters, these emails may contain '.'s or '+'s.

If you add periods ('.') between some characters in the local name part of an email address, mail sent there will be forwarded to the same address without dots in the local name. For example, "alice.z@leetcode.com (mailto:alice.z@leetcode.com)" and "alicez@leetcode.com (mailto:alicez@leetcode.com)" forward to the same email address. (Note that this rule does not apply for domain names.)

If you add a plus ('+') in the local name, everything after the first plus sign will be ignored. This allows certain emails to be filtered, for example m.y+name@email.com (mailto:m.y+name@email.com) will be forwarded to my@email.com (mailto:my@email.com). (Again, this rule does not apply for domain names.)

It is possible to use both of these rules at the same time.

Given a list of emails, we send one email to each address in the list. How many different addresses actually receive mails?

Example 1:

```
Input: ["test.email+alex@leetcode.com","test.e.mail+bob.cathy@leetcode.com","te
stemail+david@lee.tcode.com"]
Output: 2
```

Explanation: "testemail@leetcode.com (mailto:testemail@leetcode.com)" and "testemail@lee.tcode.com (mailto:testemail@lee.tcode.com)" actually receive mails

Note:

```
1 <= emails[i].length <= 100
1 <= emails.length <= 100
Each emails[i] contains exactly one '@' character.
All local and domain names are non-empty.
Local names do not start with a '+' character.
```

Performance

- **Complexity Analysis**
- O() in time.
- O() in

In [ ]:

125. Valid Palindrome (https://leetcode.com/problems/valid-palindrome/)

Easy

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

Note: For the purpose of this problem, we define empty string as valid palindrome.

Example 1:

```
Input: "A man, a plan, a canal: Panama"
Output: True
```

Example 2:

```
Input: "race a car"
Output: False
```

Performance

- Runtime: 60 ms, faster than 84.31% of Python3 online submissions for Valid Palindrome.
- Memory Usage: 12.7 MB, less than 100.00% of Python3 online submissions for Valid Palindrome.

Complexity Analysis

```
O(n) in time.
O(n) in space.
```

In [4]:
```python
class Solution:
    def isPalindrome(self, s: 'str') -> 'bool':
        ls = s.lower()
        ns = ''
        for c in ls:
            if ('0'<=c<='9') or ('a'<=c<='z'):
                ns = ns + c
        return ns == ns[::-1]

my_sol = Solution()
print("Should print True:", my_sol.isPalindrome('A man, a plan, a canal: Panama'))
print("Should print False:", my_sol.isPalindrome('race a car'))
print("Should print False:", my_sol.isPalindrome('0P'))
```

```
Should print True: True
Should print False: False
Should print False: False
```

151. Reverse Words in a String (https://leetcode.com/problems/reverse-words-in-a-string/)

Medium

Given an input string, reverse the string word by word.

Example:

```
Input: "the sky is blue",
Output: "blue is sky the".
```

Note:

- A word is defined as a sequence of non-space characters.
- Input string may contain leading or trailing spaces. However, your reversed string should not contain leading or trailing spaces.
- You need to reduce multiple spaces between two words to a single space in the reversed string.

Performance

- Runtime: 20 ms, faster than 99.66% of Python online submissions for Reverse Words in a String.
- Memory Usage: 11.1 MB, less than 100.00% of Python online submissions for Reverse Words in a String.

Complexity Analysis

O(n) in time.
O(n) in space.

In [5]:
```
import re

class Solution(object):
    def reverseWords(self, s):
        """
        :type s: str
        :rtype: str
        """
        new_s = re.sub(' +', ' ',s.strip())
        return ' '.join(new_s.split(' ')[::-1])

my_sol = Solution()
print("Should print 'Panama canal: a plan, a man, A':", my_sol.reverseWords('A man, a p
print("Should print '':", my_sol.reverseWords(' '))
```

```
Should print 'Panama canal: a plan, a man, A': Panama canal: a plan, a man, A
Should print '':
```

In [6]:
```
# Solution without using the re library

class Solution(object):
    def reverseWords(self, s):
        """
        :type s: str
        :rtype: str
        """
        list_of_words = []
        word = ''
        for i in range(len(s)):
            if s[i] != ' ':
                word = word + s[i]
            if s[i] == ' ' and len(word) > 0:
                list_of_words.append(word)
                word = ''
        if len(word) > 0:
            list_of_words.append(word)
        return ' '.join(list_of_words[::-1])

my_sol = Solution()
print("Should print 'Panama canal: a plan, a man, A':", my_sol.reverseWords('A man, a p
print("Should print '':", my_sol.reverseWords(' '))
```

```
Should print 'Panama canal: a plan, a man, A': Panama canal: a plan, a man, A
Should print '':
```

475. Heaters (https://leetcode.com/problems/heaters/)

Easy

Winter is coming! Your first job during the contest is to design a standard heater with fixed warm radius to warm all the houses.

Now, you are given positions of houses and heaters on a horizontal line, find out minimum radius of heaters so that all houses could be covered by those heaters.

So, your input will be the positions of houses and heaters seperately, and your expected output will be the minimum radius standard of heaters.

Note:

- Numbers of houses and heaters you are given are non-negative and will not exceed 25000.
- Positions of houses and heaters you are given are non-negative and will not exceed 10^9.
- As long as a house is in the heaters' warm radius range, it can be warmed.
- All the heaters follow your radius standard and the warm radius will the same.

Example 1:

```
Input: [1,2,3],[2]
Output: 1
```

Explanation: The only heater was placed in the position 2, and if we use the radius 1 standard, then all the houses can be warmed.

Example 2:

```
Input: [1,2,3,4],[1,4]
Output: 1
```

Explanation: The two heater was placed in the position 1 and 4. We need to use radius 1 standard, then all the houses can be warmed.

Performance

- Runtime: 108 ms, faster than 90.12% of Python3 online submissions for Heaters.
- Memory Usage: 15.1 MB, less than 100.00% of Python3 online submissions for Heaters.

Complexity Analysis

```
O([n + m] log n) in time
O(n) in space
```

In [7]:
```python
import bisect

class Solution:
    def findRadius(self, houses: 'List[int]', heaters: 'List[int]') -> 'int':
        heaters.sort()
        max_heater_dist = 0
        min_dist = 0
        for house in houses:
            left_i = bisect.bisect_left(heaters, house)
            if left_i == 0:
                min_dist = heaters[0] - house
            elif left_i == len(heaters):
                min_dist = house - heaters[len(heaters)-1]
            else:
                min_dist = min(house-heaters[left_i-1], heaters[left_i]-house)
            max_heater_dist = max(max_heater_dist, min_dist)
            #if max_heater_dist < min_dist:
            #    max_heater_dist = min_dist
        return max_heater_dist

my_sol = Solution()
print('Should print 1:', my_sol.findRadius([1,2,3],[2]))
print('Should print 1:', my_sol.findRadius([1,2,3,4],[1,4]))
```

```
Should print 1: 1
Should print 1: 1
```

349. Intersection of Two Arrays (https://leetcode.com/problems/intersection-of-two-arrays/)

Easy

Given two arrays, write a function to compute their intersection.

Example 1:

```
Input: nums1 = [1,2,2,1], nums2 = [2,2]
Output: [2]
```

Example 2:

```
Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [9,4]
```

Note:

- Each element in the result must be unique.
- The result can be in any order.

Performance

- Runtime: 36 ms, faster than 100.00% of Python3 online submissions for Intersection of Two Arrays.
- Memory Usage: 12.6 MB, less than 100.00% of Python3 online submissions for Intersection of Two Arrays.

Complexity Analysis

```
O(nm) in time
O(n+m) in space
```

In [8]:
```
class Solution:
    def intersection(self, nums1: 'List[int]', nums2: 'List[int]') -> 'List[int]':
        return list(set(nums1) & set(nums2))

my_sol = Solution()
print('Should print [2]:',  my_sol.intersection([1,2,2,1], [2,2]))
print('Should print [9,4]:',  my_sol.intersection([4,9,5], [9,4,9,8,4]))
```

```
Should print [2]: [2]
Should print [9,4]: [9, 4]
```

5. Longest Palindromic Substring (https://leetcode.com/problems/longest-palindromic-substring/)

Medium

Given a string s, find the longest palindromic substring in s. You may assume that the maximum length of s is 1000.

Example 1:

```
Input: "babad"
Output: "bab"
```

Note: "aba" is also a valid answer.

Example 2:

```
Input: "cbbd"
Output: "bb"
```

Performance

- Runtime: 1716 ms, faster than 47.00% of Python3 online submissions for Longest Palindromic Substring.
- Memory Usage: 13.3 MB, less than 25.00% of Python3 online submissions for Longest Palindromic Substring.

Complexity Analysis

```
O(log n) average case, O(n) worst case in time.
O(log n) in space.
```

In [9]:
```python
class Solution:
    def longestPalindrome(self, s: str) -> str:
        n = len(s)
        if n > 0:
            longest_pal = s[0]
            for i in range(2*n-1):
                if i%2 == 0:
                    pal = s[int(i/2)]
                    left = int(i/2) - 1
                    right = int(i/2) + 1
                else:
                    pal = ''
                    left = int(i/2)
                    right = int(i/2) + 1
                while 0 <= left and right < n:
                    if s[left] == s[right]:
                        pal = s[left:right+1]
                        if len(longest_pal) < len(pal):
                            longest_pal = pal
                        left -= 1
                        right += 1
                    else:
                        break
            return longest_pal
        return ''

my_sol = Solution()
print('Should print bab', my_sol.longestPalindrome("babad"))
print('Should print bb', my_sol.longestPalindrome("cbbd"))
```

```
Should print bab bab
Should print bb bb
```

64. Minimum Path Sum (https://leetcode.com/problems/minimum-path-sum/)

Medium

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 numbers along its path.

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

Example:

```
Input:
[[1,3,1],
 [1,5,1],
 [4,2,1]]
Output: 7
```

Explanation: Because the path 1→3→1→1→1 minimizes the sum.

Performance

- Runtime: 52 ms, faster than 86.74% of Python3 online submissions for Minimum Path Sum.
- Memory Usage: 13.4 MB, less than 88.73% of Python3 online submissions for Minimum Path Sum.

Complexity Analysis

```
O(nm) in time.
O(n) in space.
```

In [10]:
```
class Solution:
    def minPathSum(self, grid: 'List[List[int]]') -> 'int':
        nrows, ncols = len(grid), len(grid[0])
        if nrows > 0 or  ncols > 0:
            if nrows == 1: return sum(grid[0])
            if ncols == 1: return sum([row[0] for row in grid])
            path_sum = [0]*(nrows + ncols)
            path_sum[-1] = sum(grid[0][:-1]) + sum([row[-1] for row in grid])
            path_sum[0] = grid[0][0]
            self.minPathSumRec(grid, (0,0), path_sum)
            return path_sum[-1]

    def minPathSumRec(self, grid, node, path_sum):
        i, j = node
        imax, jmax = len(grid)-1, len(grid[0])-1
        if i+j > 0:
            path_sum[i+j] = path_sum[i+j-1] + grid[i][j]
        #print(i, j, path_sum)
        if i < imax:
            self.minPathSumRec(grid, (i+1, j), path_sum)
        if j < jmax:
            self.minPathSumRec(grid, (i, j+1), path_sum)
        if i == imax and j == jmax and path_sum[i+j] < path_sum[-1]:
            path_sum[-1] = path_sum[i+j]

my_sol = Solution()
print('Should print 7', my_sol.minPathSum([[1,3,1],
                                           [1,5,1],
                                           [4,2,1]]))
print('Should print 3', my_sol.minPathSum([[1,2],
                                           [1,1]]))
```

```
Should print 7 7
Should print 3 3
```

In [11]:
```
class Solution:
    def minPathSum(self, grid: 'List[List[int]]') -> 'int':
        nrows, ncols = len(grid), len(grid[0])
        """
        path_sum = [0] * ncols
        path_sum[0] = grid[0][0]
        for j in range(1,ncols):
            path_sum[j] = path_sum[j-1] + grid[0][j]
        """
        path_sum = [sum(grid[0][0:1+j]) for j in range(ncols)]
        for i in range(1, nrows):
            path_sum[0] += grid[i][0]
            for j in range(1, ncols):
                path_sum[j] = min(path_sum[j], path_sum[j-1]) + grid[i][j]
        return path_sum[-1]

my_sol = Solution()
print('Should print 7', my_sol.minPathSum([[1,3,1],
                                           [1,5,1],
                                           [4,2,1]]))
print('Should print 3', my_sol.minPathSum([[1,2],
                                           [1,1]]))
```

```
Should print 7 7
Should print 3 3
```

401. Binary Watch (https://leetcode.com/problems/binary-watch/)

Easy

A binary watch has 4 LEDs on the top which represent the hours (0-11), and the 6 LEDs on the bottom represent the minutes (0-59).

Each LED represents a zero or one, with the least significant bit on the right.

For example,

```
0 ["4:00", "8:00", "0:02", "0:04", "0:32"]
1 ["1:00", "2:00", "0:01", "0:08", "0:16"]
```

the above binary watch reads "3:25".

Given a non-negative integer n which represents the number of LEDs that are currently on, return all possible times the watch could represent.

Example:

```
Input: n = 1
Return: ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:1
6", "0:32"]
```

Note:

- The order of output does not matter.
- The hour must not contain a leading zero, for example "01:00" is not valid, it should be "1:00".
- The minute must be consist of two digits and may contain a leading zero, for example "10:2" is not valid, it should be "10:02".

Performance

- Runtime: 40 ms, faster than 60.57% of Python3 online submissions for Binary Watch.
- Memory Usage: 13.2 MB, less than 5.88% of Python3 online submissions for Binary Watch.

Complexity Analysis

```
O(1) in time.
O(1) in space.
```

In [12]:
```python
class Solution:
    def readBinaryWatch(self, num: 'int') -> 'List[str]':
        output = []
        for hour in range(12):
            for minute in range(60):
                num_ones = bin(hour).count('1') + bin(minute).count('1')
                if num_ones == num:
                    if minute < 10:
                        output.append(str(hour)+':0'+str(minute))
                    else:
                        output.append(str(hour)+':'+str(minute))
        return output

my_sol = Solution()
print('Should print ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0
      my_sol.readBinaryWatch(1))
```

```
Should print ["1:00", "2:00", "4:00", "8:00", "0:01", "0:02", "0:04", "0:08", "0:16",
"0:32"]
 ['0:01', '0:02', '0:04', '0:08', '0:16', '0:32', '1:00', '2:00', '4:00', '8:00']
```

461. Hamming Distance (https://leetcode.com/problems/hamming-distance/)

Easy

The Hamming distance between two integers is the number of positions at which the corresponding bits are different.

Given two integers x and y, calculate the Hamming distance.

Note: 0 ≤ x, y < 231.

Example:

```
Input: x = 1, y = 4

Output: 2
```

Explanation:

```
1   (0 0 0 1)
4   (0 1 0 0)
      ↑   ↑
```

The above arrows point to positions where the corresponding bits are different.

Performance

- Runtime: 48 ms, faster than 16.48% of Python3 online submissions for Hamming Distance.

- Memory Usage: 13.1 MB, less than 5.84% of Python3 online submissions for Hamming Distance.

In [13]:
```
class Solution:
    def hammingDistance(self, x: int, y: int) -> int:
        if x > y:
            return self.hammingDistance(y, x)
        ybin = bin(y)[2:]
        xbin = bin(x)[2:].zfill(len(ybin))
        dist = 0
        for i in range(len(ybin)):
            if xbin[i] != ybin[i]:
                dist += 1
        return dist

my_sol = Solution()
print('Should print 2: ', my_sol.hammingDistance(1, 4))
```

```
Should print 2:  2
```

77. Combinations (https://leetcode.com/problems/combinations/)

Medium

Given two integers n and k, return all possible combinations of k numbers out of 1 ... n.

Example:

```
Input: n = 4, k = 2

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

Performance

- Runtime: 116 ms, faster than 94.42% of Python3 online submissions for Combinations.
- Memory Usage: 15.1 MB, less than 7.73% of Python3 online submissions for Combinations.

Complexity Analysis

```
O(1) in time.
O(1) in space.
```

In [14]:
```
from itertools import combinations

class Solution:
    def combine(self, n: 'int', k: 'int') -> 'List[List[int]]':
        return list(combinations(range(1, n+1), k))

my_sol = Solution()
print('Should print [(1,2,3),(1,2,4),(1,3,4),(2,3,4)] :', my_sol.combine(4,3))
print('Should print [(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)] :', my_sol.combine(4,2))
print('Should print [(1,2),(1,3),(2,3)] :', my_sol.combine(3,2))
print('Should print [(1,2,3)] :', my_sol.combine(3,3))
```

```
Should print [(1,2,3),(1,2,4),(1,3,4),(2,3,4)] : [(1, 2, 3), (1, 2, 4), (1, 3, 4),
(2, 3, 4)]
Should print [(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)] : [(1, 2), (1, 3), (1, 4), (2, 3),
(2, 4), (3, 4)]
Should print [(1,2),(1,3),(2,3)] : [(1, 2), (1, 3), (2, 3)]
Should print [(1,2,3)] : [(1, 2, 3)]
```

200. Number of Islands (https://leetcode.com/problems/number-of-islands/)

Medium

Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Example 1:

```
Input:
11110
11010
11000
00000

Output: 1
```

Example 2:

```
Input:
11000
11000
00100
00011

Output: 3
```

Performance

- Runtime: 116 ms, faster than 15.13% of Python3 online submissions for Number of Islands.
- Memory Usage: 19.4 MB, less than 5.13% of Python3 online submissions for Number of Islands.

Complexity Analysis

- O(n m) in time.
- O(n m) in space.

In [67]:
```
class Solution:
    def numIslands(self, grid: 'List[List[str]]') -> 'int':
        if len(grid) == 0 or len(grid[0]) == 0:
            return 0
        nr, nc = len(grid), len(grid[0])
        n_islands = 0
        visited = set()
        for i in range(nr):
            for j in range(nc):
                if not((i,j) in visited) and grid[i][j] == '1':
                    self.rmIsland(grid, (i, j), visited)
                    n_islands += 1
        return n_islands

    def rmIsland(self, grid, loc, visited):
        visited.add(loc)
        i, j = loc
        nr, nc = len(grid), len(grid[0])
        if 0 <= i < nr and 0 <= j < nc and grid[i][j] == '1':
            grid[i][j] = '0'
            self.rmIsland(grid, (i + 1, j), visited)
            self.rmIsland(grid, (i - 1, j), visited)
            self.rmIsland(grid, (i, j + 1), visited)
            self.rmIsland(grid, (i, j - 1), visited)

my_sol = Solution()
print('Should print 1:', my_sol.numIslands([['1','1','1','1','0'],
                                            ['1','1','0','1','0'],
                                            ['1','1','0','0','0'],
                                            ['0','0','0','0','0']]))
print('Should print 3:', my_sol.numIslands([['1','1','0','0','0'],
                                            ['1','1','0','0','0'],
                                            ['0','0','1','0','0'],
                                            ['0','0','0','1','1']]))
```

```
Should print 1: 1
Should print 3: 3
```

528. Random Pick with Weight (https://leetcode.com/problems/random-pick-with-weight/)

Medium

Given an array `w` of positive integers, where `w[i]` describes the weight of index `i`, write a function pickIndex which randomly picks an index in proportion to its weight.

Note:

- 1 <= w.length <= 10000
- 1 <= w[i] <= 10^5
- pickIndex will be called at most 10000 times.

Example 1:

```
Input:

["Solution","pickIndex"]
[[[1]],[]]

Output: [null,0]
```

Example 2:

```
Input:

["Solution","pickIndex","pickIndex","pickIndex","pickIndex","pickIndex"]
[[[1,3]],[],[],[],[],[]]

Output: [null,0,1,1,1,0]
```

Explanation of Input Syntax:

The input is two lists: the subroutines called and their arguments. Solution's constructor has one argument, the array w. pickIndex has no arguments. Arguments are always wrapped with a list, even if there aren't any.

Performance

- Runtime: 1340 ms, faster than 5.93% of Python3 online submissions for Random Pick with Weight.
- Memory Usage: 17.3 MB, less than 7.69% of Python3 online submissions for Random Pick with Weight.

Complexity Analysis

- O(log(n)) in time.
- O(n) in space.

In [22]:
```python
import bisect
import random

class Solution:
    def __init__(self, w: 'List[int]'):
        self.w = [sum(w[:i+1]) for i in range(len(w))]
        self.n = self.w[-1]

    def pickIndex(self) -> int:
        return bisect.bisect_right(self.w, random.randint(0,self.n-1))

# Your Solution object will be instantiated and called as such:
# obj = Solution(w)
# param_1 = obj.pickIndex()
```

139. Word Break (https://leetcode.com/problems/word-break/)

Medium

Given a non-empty string s and a dictionary wordDict containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.

Note:

- The same word in the dictionary may be reused multiple times in the segmentation.
- You may assume the dictionary does not contain duplicate words.

Example 1:

```
Input: s = "leetcode", wordDict = ["leet", "code"]
Output: true
```

Explanation: Return true because "leetcode" can be segmented as "leet code".

Example 2:

```
Input: s = "applepenapple", wordDict = ["apple", "pen"]
Output: true
```

Explanation: Return true because "applepenapple" can be segmented as "apple pen apple". Note that you are allowed to reuse a dictionary word.

Example 3:

```
Input: s = "catsandog", wordDict = ["cats", "dog", "sand", "and", "cat"]
Output: false
```

Performance

- Runtime: 44 ms, faster than 74.53% of Python3 online submissions for Word Break.
- Memory Usage: 13.2 MB, less than 5.11% of Python3 online submissions for Word Break.

Complexity Analysis

- O() in time.
- O() in space.

In [60]:
```python
# Timed out!
from collections import defaultdict

class Solution:
    def wordBreak(self, s: 'str', wordDict: 'List[str]') -> 'bool':
        w_dict = defaultdict(bool)
        for w in wordDict:
            w_dict[w] = True
        return self.wordBreakRec(s, w_dict)

    def wordBreakRec(self, s, w_dict):
        if len(s) == 0:
            return True
        for i in range(len(s)):
            #print('\ntest:', i, s[:i+1], )
            if w_dict[s[:i+1]] and self.wordBreakRec(s[i+1:], w_dict):
                return True
        return False

my_sol = Solution()
print('Should print True:', my_sol.wordBreak("leetcode", ["leet", "code"]))
print('Should print True:', my_sol.wordBreak("applepenapple", ["apple", "pen"]))
print('Should print False:', my_sol.wordBreak("catsandog", ["cats", "dog", "sand", "and
```

```
Should print True: True
Should print True: True
Should print False: False
```

In [61]:
```
from collections import deque

class Solution:
    def wordBreak(self, s: 'str', wordDict: 'List[str]') -> 'bool':
        queue = deque([s])
        w_set = set(wordDict)
        q_set = set()
        while queue:
            word = queue.popleft()
            if word in w_set:
                return True
            for i in range(len(word)):
                if word[:i] in w_set and not(word[i:] in q_set):
                    queue.append(word[i:])
                    q_set.add(word[i:])
        return False

my_sol = Solution()
print('Should print True:', my_sol.wordBreak("leetcode", ["leet", "code"]))
print('Should print True:', my_sol.wordBreak("applepenapple", ["apple", "pen"]))
print('Should print False:', my_sol.wordBreak("catsandog", ["cats", "dog", "sand", "and
```

```
Should print True: True
Should print True: True
Should print False: False
```

503. Next Greater Element II (https://leetcode.com/problems/next-greater-element-ii/)

Medium

Given a circular array (the next element of the last element is the first element of the array), print the Next Greater Number for every element. The Next Greater Number of a number x is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, output -1 for this number.

Example 1:

```
Input: [1,2,1]
Output: [2,-1,2]
```

Explanation:

- The first 1's next greater number is 2;
- The number 2 can't find next greater number;
- The second 1's next greater number needs to search circularly, which is also 2.

Note: The length of given array won't exceed 10000.

In [ ]:
```
class Solution:
    def nextGreaterElements(self, nums: List[int]) -> List[int]:
```

80. Remove Duplicates from Sorted Array II (https://leetcode.com/problems/remove-duplicates-from-sorted-array-ii/)

Medium

Given a sorted array nums, remove the duplicates in-place such that duplicates appeared at most twice and return the new length.

Do not allocate extra space for another array, you must do this by modifying the input array in-place with O(1) extra memory.

Example 1:

```
Given nums = [1,1,1,2,2,3],

Your function should return length = 5,
with the first five elements of nums being 1, 1, 2, 2 and 3 respectively.

It doesn't matter what you leave beyond the returned length.
```

Example 2:

```
Given nums = [0,0,1,1,1,1,2,3,3],

Your function should return length = 7,
with the first seven elements of nums being modified to 0, 0, 1, 1, 2, 3 and 3
 respectively.

It doesn't matter what values are set beyond the returned length.
```

Clarification:

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by reference, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

```
// nums is passed in by reference. (i.e., without making a copy)
int len = removeDuplicates(nums);

// any modification to nums in your function would be known by the caller.
// using the length returned by your function, it prints the first len element
s.
for (int i = 0; i < len; i++) {
    print(nums[i]);
}
```

In [ ]:
```
class Solution:
    def removeDuplicates(self, nums: List[int]) -> int:

```

Cipher (https://www.hackerrank.com/challenges/cipher/problem)

Jack and Daniel are friends. They want to encrypt their conversations so that they can save themselves from interception by a detective agency so they invent a new cipher.

Every message is encoded to its binary representation. Then it is written down times, shifted by bits. Each of the columns is XORed together to get the final encoded string.

If and it looks like so:

```
1001011     shift 0
01001011    shift 1
001001011   shift 2
0001001011  shift 3
----------
1110101001  <- XORed/encoded string s
```

Now we have to decode the message. We know that . The first digit in so our output string is going to start with . The next two digits are also , so they must have been XORed with . We know the first digit of our shifted string is a as well. Since the digit of is , we XOR that with our and now know there is a in the position of the original string. Continue with that logic until the end.

Then the encoded message and the key are sent to Daniel.

Jack is using this encoding algorithm and asks Daniel to implement a decoding algorithm. Can you help Daniel implement this?

Function Description

Complete the cipher function in the editor below. It should return the decoded string.

cipher has the following parameter(s):

k: an integer that represents the number of times the string is shifted s: an encoded string of binary digits

Input Format

The first line contains two integers and , the length of the original decoded string and the number of shifts. The second line contains the encoded string consisting of ones and zeros.

Constraints

It is guaranteed that is valid.

Output Format

Return the decoded message of length , consisting of ones and zeros.

```
Sample Input 0

7 4
1110100110

Sample Output 0

1001010
```

Explanation 0

```
1001010
 1001010
  1001010
   1001010
----------
1110100110
```

```
Sample Input 1

6 2
1110001
Sample Output 1

101111
```

Explanation 1

```
101111
 101111
-------
1110001
```

```
Sample Input 2

10 3
1110011011
Sample Output 2

10000101
```

Explanation 2

```
10000101 010000101

0010000101
1110011011
```

In [ ]:
```python
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the cipher function below.
def cipher(k, s):

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    nk = input().split()

    n = int(nk[0])

    k = int(nk[1])

    s = input()

    result = cipher(k, s)

    fptr.write(result + '\n')

    fptr.close()
```

179. Largest Number (https://leetcode.com/problems/largest-number/)

Medium

Given a list of non negative integers, arrange them such that they form the largest number.

Example 1:

```
Input: [10,2]
Output: "210"
```

Example 2:

```
Input: [3,30,34,5,9]
Output: "9534330"
```

Note: The result may be very large, so you need to return a string instead of an integer.

In [ ]:
```python
class Solution:
    def largestNumber(self, nums: List[int]) -> str:
```

120. Triangle (https://leetcode.com/problems/triangle/)

Medium

Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

For example, given the following triangle

```
  [[2],
  [3,4],
 [6,5,7],
[4,1,8,3]]
```

The minimum path sum from top to bottom is 11 (i.e., 2 + 3 + 5 + 1 = 11).

Note:

Bonus point if you are able to do this using only O(n) extra space, where n is the total number of rows in the triangle.

In [ ]:
```
class Solution:
    def minimumTotal(self, triangle: List[List[int]]) -> int:

```

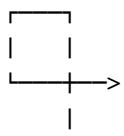
335. Self Crossing (https://leetcode.com/problems/self-crossing/)

Hard

You are given an array x of n positive numbers. You start at point (0,0) and moves x[0] metres to the north, then x[1] metres to the west, x[2] metres to the south, x[3] metres to the east and so on. In other words, after each move your direction changes counter-clockwise.

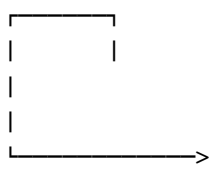
Write a one-pass algorithm with O(1) extra space to determine, if your path crosses itself, or not.

Example 1:



```
Input: [2,1,1,2]
Output: true
```

Example 2:



```
Input: [1,2,3,4]
Output: false
```

Example 3:

```
Input: [1,1,1,1]
Output: true
```

In [ ]:
```
class Solution:
    def isSelfCrossing(self, x: List[int]) -> bool:
```

91. Decode Ways (https://leetcode.com/problems/decode-ways/)

Medium

A message containing letters from A-Z is being encoded to numbers using the following mapping:

```
'A' -> 1
'B' -> 2
...
'Z' -> 26
```

Given a non-empty string containing only digits, determine the total number of ways to decode it.

Example 1:

```
Input: "12"
Output: 2
```

Explanation: It could be decoded as "AB" (1 2) or "L" (12).

Example 2:

```
Input: "226"
Output: 3
```

Explanation: It could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).

In [ ]:
```
class Solution:
    def numDecodings(self, s: str) -> int:
```

187. Repeated DNA Sequences (https://leetcode.com/problems/repeated-dna-sequences/)

Medium

All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: "ACGAATTCCG". When studying DNA, it is sometimes useful to identify repeated sequences within the DNA.

Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.

Example:

```
Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC", "CCCCCAAAAA"]
```

In [ ]:
```
class Solution:
    def findRepeatedDnaSequences(self, s: str) -> List[str]:
```

215. Kth Largest Element in an Array (https://leetcode.com/problems/kth-largest-element-in-an-array/)

Medium

Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example 1:

```
Input: [3,2,1,5,6,4] and k = 2
Output: 5
```

Example 2:

```
Input: [3,2,3,1,2,4,5,5,6] and k = 4
Output: 4
```

Note:

- You may assume k is always valid, 1 ≤ k ≤ array's length.

In [ ]:
```
class Solution:
    def findKthLargest(self, nums: List[int], k: int) -> int:

```

In [ ]:
```

```

In [ ]:
```

```