Python Coding Questions

|  |  |
| --- | --- |
| S.No | Description |
| 1. | Given a string s, return the longest Palindromic substring in s.  **Example 1:**  Input: s = "babad" Output: "bab"  Explanation: "aba" is also a valid answer.  **Example 2:**  Input: s = "cbbd" Output: "bb"  **Constraints:**  1 <= s.length <= 1000  s consist of only digits and English letters. |
| 2. | Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.  You may assume that each input would have exactly one solution, and you may not use the same element twice.  You can return the answer in any order.  **Example 1:**  Input: nums = [2,7,11,15], target = 9 Output: [0,1]  Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].  **Example 2:**  Input: nums = [3,2,4], target = 6 Output: [1,2]  **Example 3:**  Input: nums = [3,3], target = 6 Output: [0,1]  **Constraints:**  2 <= nums.length <= 10^4  -10^9 <= nums[i] <= 10^9  -10^9 <= target <= 10^9 Only one valid answer exists. |
| 3. | You are given the heads of two sorted linked lists list1 and list2. |

|  |  |
| --- | --- |
|  | Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.  Return the head of the merged linked list.  **Example 1:**  Input: list1 = [1,2,4], list2 = [1,3,4]  Output: [1,1,2,3,4,4]  **Example 2:**  Input: list1 = [], list2 = [] Output: []  **Example 3:**  Input: list1 = [], list2 = [0] Output: [0]  **Constraints:**  The number of nodes in both lists is in the range [0, 50].  -100 <= Node.val <= 100  Both list1 and list2 are sorted in non-decreasing order. |
| 4. | Given an integer array nums sorted in non-decreasing order, remove the duplicates [in-place](https://en.wikipedia.org/wiki/In-place_algorithm) such that each unique element appears only once. The relative order of the elements should be kept the same. Then return the number of unique elements in nums.  Consider the number of unique elements of nums to be k, to get accepted, you need to do the following things:  Change the array nums such that the first k elements of nums contain the unique elements in the order they were present in nums initially. The remaining elements of nums are not important as well as the size of nums.  Return k.  **Example 1:**  Input: nums = [1,1,2] Output: 2, nums = [1,2,\_]  Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.  It does not matter what you leave beyond the returned k (hence they are underscores). |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: nums = [0,0,1,1,1,2,2,3,3,4]  Output: 5, nums = [0,1,2,3,4,\_,\_,\_,\_,\_]  Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.  It does not matter what you leave beyond the returned k (hence they are underscores).  **Constraints:**  1 <= nums.length <= 3 \* 10^4  -100 <= nums[i] <= 100  nums is sorted in non-decreasing order. |
| 5. | Roman numerals are represented by seven different  **symbols: I, V, X, L, C, D and M.**  Symbol Value I 1  V 5  X 10  L 50  C 100  D 500  M 1000  For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.  Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:  I can be placed before V (5) and X (10) to make 4 and 9.  X can be placed before L (50) and C (100) to make 40 and 90.  C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer.  **Example 1:**  Input: s = "III" Output: 3 Explanation: III = 3. **Example 2:**  Input: s = "LVIII" Output: 58  Explanation: L = 50, V= 5, III = 3.  **Example 3:** |

|  |  |
| --- | --- |
|  | Input: s = "MCMXCIV"  Output: 1994  Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.  **Constraints:**  1 <= s.length <= 15  s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').  It is guaranteed that s is a valid roman numeral in the range [1, 3999]. |
| 6. | Given an integer x, return true if x is a palindrome, and false otherwise.  **Example 1:**  Input: x = 121 Output: true  Explanation: 121 reads as 121 from left to right and from right to left.  **Example 2:**  Input: x = -121 Output: false  Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.  **Example 3:**  Input: x = 10 Output: false  Explanation: Reads 01 from right to left. Therefore it is not a palindrome.  **Constraints:**  -2^31 <= x <= 2^31 - 1 |
| 7. | Given a string s consisting of words and spaces, return the length of the last word in the string.  A word is a maximal substring consisting of non-space characters only.  **Example 1:**  Input: s = "Hello World" Output: 5  Explanation: The last word is "World" with length 5.  **Example 2:**  Input: s = " fly me to the moon " Output: 4  Explanation: The last word is "moon" with length 4.  **Example 3:**  Input: s = "luffy is still joyboy" Output: 6  Explanation: The last word is "joyboy" with length 6.  **Constraints:** |

|  |  |
| --- | --- |
|  | 1 <= s.length <= 10^4  s consists of only English letters and spaces ' '. There will be at least one word in s. |
| 8. | A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.  Given a string s, return true if it is a palindrome, or false otherwise.  **Example 1:**  Input: s = "A man, a plan, a canal: Panama" Output: true  Explanation: "amanaplanacanalpanama" is a palindrome.  **Example 2:**  Input: s = "race a car" Output: false  Explanation: "raceacar" is not a palindrome.  **Example 3:**  Input: s = " " Output: true  Explanation: s is an empty string "" after removing non-alphanumeric characters.  Since an empty string reads the same forward and backward, it is a palindrome.  **Constraints:**  1 <= s.length <= 2 \* 10^5  s consists only of printable ASCII characters. |
| 9. | Given an integer numRows, return the first numRows of Pascal's triangle.  **Example 1:**  Input: numRows = 5  Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]  **Example 2:**  Input: numRows = 1 Output: [[1]]  **Constraints:**  1 <= numRows <= 30 |
| 10. | Given an integer rowIndex, return the rowIndexth (0-indexed) row of the Pascal's triangle.  **Example 1:**  Input: rowIndex = 3 Output: [1,3,3,1] |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: rowIndex = 0 Output: [1]  **Example 3:**  Input: rowIndex = 1 Output: [1,1]  **Constraints:**  0 <= rowIndex <= 33 |
| 11. | Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.  You must implement a solution with a linear runtime complexity and use only constant extra space.  **Example 1:**  Input: nums = [2,2,1] Output: 1  **Example 2:**  Input: nums = [4,1,2,1,2] Output: 4  **Example 3:**  Input: nums = [1] Output: 1  **Constraints:**  1 <= nums.length <= 3 \* 10^4  -3 \* 10^4 <= nums[i] <= 3 \* 10^4  Each element in the array appears twice except for one element which appears only once. |
| 12. | Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.  **Example 1:**  Input: haystack = "sadbutsad", needle = "sad" Output: 0  Explanation: "sad" occurs at index 0 and 6.  The first occurrence is at index 0, so we return 0.  **Example 2:**  Input: haystack = "PythonCode", needle = "Pythoc" Output: -1  Explanation: "Pythoc" did not occur in "PythonCode", so we return -1.  **Constraints:**  1 <= haystack.length, needle.length <= 10^4  haystack and needle consist of only lowercase English characters. |

|  |  |
| --- | --- |
|  |  |
| 13. | Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.  You must write an algorithm with O(log n) runtime complexity.  **Example 1:**  Input: nums = [1,3,5,6], target = 5 Output: 2  **Example 2:**  Input: nums = [1,3,5,6], target = 2 Output: 1  **Example 3:**  Input: nums = [1,3,5,6], target = 7 Output: 4  **Constraints:**  1 <= nums.length <= 10^4  -10^4 <= nums[i] <= 10^4  nums contains distinct values sorted in ascending order.  -10^4 <= target <= 10^4 |
| 14. | 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: strs = ["flower","flow","flight"] Output: "fl"  **Example 2:**  Input: strs = ["dog","racecar","car"] Output: ""  Explanation: There is no common prefix among the input strings.  **Constraints:**  1 <= strs.length <= 200  0 <= strs[i].length <= 200  strs[i] consists of only lowercase English letters. |
| 15. | Given an integer n, return true if it is a power of two. Otherwise, return false.  An integer n is a power of two, if there exists an integer x such that n == 2x.  **Example 1:** Input: n = 1 Output: true  Explanation: 2^0 = 1 |

|  |  |
| --- | --- |
|  | **Example 2:** Input: n = 16 Output: true  Explanation: 2^4 = 16  **Example 3:** Input: n = 3 Output: false  **Constraints:**  -2^31 <= n <= 2^31 - 1 |
| 16. | Given two strings s and t, return true if t is an anagram of s, and false otherwise.  An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.  **Example 1:**  Input: s = "anagram", t = "nagaram" Output: true  **Example 2:**  Input: s = "rat", t = "car" Output: false  **Constraints:**  1 <= s.length, t.length <= 5 \* 10^4  s and t consist of lowercase English letters. |
| 17. | Given an integer array nums, return the third distinct maximum number in this array. If the third maximum does not exist, return the maximum number.  **Example 1:**  Input: nums = [3,2,1] Output: 1 Explanation:  The first distinct maximum is 3. The second distinct maximum is 2. The third distinct maximum is 1.  **Example 2:**  Input: nums = [1,2] Output: 2 Explanation:  The first distinct maximum is 2. The second distinct maximum is 1.  The third distinct maximum does not exist, so the maximum (2) is returned |

|  |  |
| --- | --- |
|  | instead.  **Example 3:**  Input: nums = [2,2,3,1] Output: 1  Explanation:  The first distinct maximum is 3.  The second distinct maximum is 2 (both 2's are counted together since they have the same value).  The third distinct maximum is 1.  **Constraints:**  1 <= nums.length <= 10^4  -2^31 <= nums[i] <= 2^31 - 1 |
| 18. | Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a string.  You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.  **Example 1:**  Input: num1 = "11", num2 = "123" Output: "134"  **Example 2:**  Input: num1 = "456", num2 = "77" Output: "533"  **Example 3:**  Input: num1 = "0", num2 = "0" Output: "0"  **Constraints:**  1 <= num1.length, num2.length <= 10^4 num1 and num2 consist of only digits.  num1 and num2 don't have any leading zeros except for the zero itself. |
| 19. | Given a string s containing just the characters **'(', ')', '{', '}', '[' and ']',**  determine if the input string is valid. An input string is valid if:  Open brackets must be closed by the same type of brackets. Open brackets must be closed in the correct order.  Every close bracket has a corresponding open bracket of the same type.  **Example 1:** Input: s = "()" Output: true |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: s = "()[]{}" Output: true **Example 3:**  Input: s = "(]" Output: false  **Constraints:**  1 <= s.length <= 10^4  s consists of parentheses only '()[]{}'. |
| 20. | Given a string s, check if it can be constructed by taking a substring of it and appending multiple copies of the substring together.  **Example 1:**  Input: s = "abab" Output: true  Explanation: It is the substring "ab" twice.  **Example 2:**  Input: s = "aba" Output: false **Example 3:**  Input: s = "abcabcabcabc" Output: true  Explanation: It is the substring "abc" four times or the substring "abcabc" twice.  **Constraints:**  1 <= s.length <= 10^4  s consists of lowercase English letters. |
| 21. | Given a list of numbers, you have to sort them in non decreasing order.  **Input Format**  The first line contains a single integer N, denoting the number of integers in the list.  The next N lines contain a single integer each, denoting the elements of the list.  **Output Format**  Output N lines, containing one integer each, in non-decreasing order.  **Constraints**  1≤N≤10^6  0≤0≤ elements of the list ≤10^6≤10^6  Input :5 5 3 6 7 1  Output :1 3 5 6 7 |
| 22. | Recently, Ravi visited his doctor. The doctor advised Chef to drink at  least 2000 ml of water each day. |

|  |  |
| --- | --- |
|  | Ravi drank X ml of water today. Determine if Ravi followed the doctor's advice or not.  **Input Format**  The first line contains a single integer T — the number of test cases. Then the test cases follow.  The first and only line of each test case contains one integer X — the amount of water Ravi drank today.  **Output Format**  For each test case, output YES if Ravi followed the doctor's advice of drinking at least 2000 ml of water. Otherwise, output NO.  You may print each character of the string in uppercase or lowercase (for example, the strings YES, yEs, yes, and yeS will all be treated as identical).  Constraints 1≤T≤2000  1≤X≤4000  **Input : Output:**  3 YES  2999 NO  1450 YES  2000 |
| 23. | Sita and Geetha are playing with dice. In one turn, both of them roll their dice at once.  They consider a turn to be good if the sum of the numbers on their dice is greater than 6.  Given that in a particular turn Sita and Geetha got X and Y on their respective dice, find whether the turn was good.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case contains two space-separated integers X and Y — the numbers Sita and Geetha got on their respective dice.  **Output Format**  For each test case, output on a new line, YES, if the turn was good and NO otherwise.  Each character of the output may be printed in either uppercase or lowercase. That is, the strings NO, no, nO, and No will be treated as equivalent.  Constraints 1≤T≤100  1≤X,Y≤6  **Input :** 4 **Output:**  1 4 NO   1. 4 YES 2. 2 NO   2 6 YES |

|  |  |
| --- | --- |
| 24. | Harsh was recently gifted a book consisting of N pages. Each page contains |
|  | exactly M words printed on it. As he was bored, he decided to count the |
|  | number of words in the book. |
|  | Help Harsh find the total number of words in the book. |
|  | **Input Format** |
|  | The first line of input will contain a single integer T, denoting the number of |
|  | test cases. |
|  | Each test case consists of two space-separated integers on a single |
|  | line, N and M — the number of pages and the number of words on each |
|  | page, respectively. |
|  | **Output Format** |
|  | For each test case, output on a new line, the total number of words in the |
|  | book. |
|  | **Constraints** |
|  | 1≤T≤100 |
|  | 1≤N≤100 |
|  | 1≤M≤100 |
|  | **Input :**4 **Output:** |
|  | 1 1 1 |
|  | 4 2 8 |
|  | 2 4 8 |
|  | 95 42 3990 |
| 25. | Chef is fond of burgers and decided to make as many burgers as possible. |
|  | Chef has A patties and B buns. To make 1 burger, Chef needs 1 patty |
|  | and 1 bun. |
|  | Find the maximum number of burgers that Chef can make. |
|  | **Input Format** |
|  | The first line of input will contain an integer T — the number of test cases. |
|  | The description of T test cases follows. |
|  | The first and only line of each test case contains two space-separated |
|  | integers A and B, the number of patties and buns respectively. |
|  | **Output Format** |
|  | For each test case, output the maximum number of burgers that Chef can |
|  | make. |
|  | **Constraints** |
|  | 1≤T≤1000 |
|  | 1≤A,B≤105 |
|  | **Input:** 4 **Output:** |
|  | 2 2 2 |
|  | 2 3 2 |
|  | 3 2 2 |
|  | 23 17 17 |
| 26. | Sunil aims to be the richest person in Iceland by his new restaurant  franchise. Currently, his assets are worth A billion dollars and have no |

|  |  |
| --- | --- |
|  | liabilities. He aims to increase his assets by X billion dollars per year. |
| Also, all the richest people in Iceland are not planning to grow and maintain |
| their current worth. |
| To be the richest person in Iceland, he needs to be worth at least B billion |
| dollars. How many years will it take Sunil to reach his goal if his value |
| increases by X billion dollars each year? |
| **Input** |
| The first line contains an integer T, the number of test cases. Then the test |
| cases follow. |
| Each test case contains a single line of input, three integers A, B, X. |
| **Output** |
| For each test case, output in a single line the answer to the problem. |
| **Constraints** |
| 1≤T≤21 000 |
| 100≤A<B≤200 |
| 1≤X≤50 |
| X divides B−A |
| **Input :** 3 **Output:** |
| 100 200 10 10 |
| 111 199 11 8 |
| 190 200 10 1 |
| 27. | Chef will have N guests in his house today. He wants to serve at least one |
|  | dish to each of the N guests. Chef can make two types of dishes. He needs |
|  | one fruit and one vegetable to make the first type of dish and one vegetable |
|  | and one fish to make the second type of dish. Now Chef |
|  | has A fruits, B vegetables, and C fishes in his house. Can he prepare at |
|  | least N dishes in total? |
|  | **Input Format** |
|  | First line will contain T, number of testcases. Then the testcases follow. |
|  | Each testcase contains of a single line of input, four integers N,A,B,C. |
|  | **Output Format** |
|  | For each test case, print "YES" if Chef can prepare at least N dishes, |
|  | otherwise print "NO". Print the output without quotes. |
|  | **Constraints** |
|  | 1≤T≤100 |
|  | 1≤N,A,B,C≤100 |
|  | **Input :** 4 **Output:** |
|  | 2 1 2 1 YES |
|  | 3 2 2 2 NO |
|  | 4 2 6 3 YES |
|  | 3 1 3 1 NO |
| 28. | You are given two integers N and K. You may perform the following |

|  |  |
| --- | --- |
|  | operation any number of times (including zero): change N to N−K, i.e. subtract K from N. Find the smallest non-negative integer value of N you can obtain this way.  **Input**  The first line of the input contains a single integer T denoting the number of test cases. The description of T test cases follows.  The first and only line of each test case contains two space-separated integers N and K.  **Output**  For each test case, print a single line containing one integer — the smallest value you can get.  **Constraints**  1≤T≤10^5  1≤N≤10^9  0≤K≤10^9  **Input :** 3 **Output:**  5 2 1  4 4 0  2 5 2 |
| 29. | Alice and Bob are playing a game of [Blobby Volley](https://blobbyvolley.de/data/bv2browser/index.html). In this game, in each turn, one player is the server and the other player is the receiver. Initially, Alice is the server, and Bob is the receiver.  If the server wins the point in this turn, their score increases by 1, and they remain as the server for the next turn.  But if the receiver wins the point in this turn, their score does not increase. But they become the server in the next turn.  In other words, your score increases only when you win a point when you are the server.  Please see the Sample Inputs and Explanation for more detailed explanation.  They start with a score of 00 each, and play N turns. The winner of each of those hands is given to you as a string consisting of 'A's and 'B's. 'A' denoting that Alice won that point, and 'B' denoting that Bob won that point. Your job is the find the score of both of them after the N turns.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of two lines of input.  The first line of each test case contains one integer N — the number of turns.  The line contains a string S of length N.  If the ith character of this string is 'A', then Alice won that point. |

|  |  |
| --- | --- |
|  | If the ith character of this string is 'B', then Bob won that point.  **Output Format**  For each test case, output on a new line, two space-separated integers - Alice's final score, and Bob's final score.  Constraints 1≤T≤1000  1≤N≤1000  Length of ∣S∣ = N  S consists only of the characters 'A' and 'B'.  **Input:** 4 **Output:**  3 3 0  AAA 0 3  4 1 1  BBBB 0 0  5  ABABB 5 BABAB |
| 30. | John has a string S with him. John is happy if the string contains a contiguous substring of length strictly greater than 22 in which all its characters are vowels.  Determine whether John is happy or not.  Note that, in english alphabet, vowels are a, e, i, o, and u.  **Input Format**  First line will contain T, number of test cases. Then the test cases follow. Each test case contains of a single line of input, a string S.  **Output Format**  For each test case, if John is happy, print HAPPY else print SAD.  You may print each character of the string in uppercase or lowercase (for example, the strings hAppY, Happy, haPpY, and HAPPY will all be treated as identical).  **Constraints**  1≤T≤1000  3≤∣S∣≤1000, where ∣S∣ is the length of S. S will only contain lowercase English letters.  **Input:** 4 **Output:**  Aeiou Happy  Abxy Sad  Aebcdefghij Sad  Abcdeeafg Happy |
| 31. | You are given a positive integer . Print a numerical triangle of height like the one below:  1  22 |

|  |  |
| --- | --- |
|  | 333 |
| 4444 |
| 55555 |
| ...... |
| Can you do it using only arithmetic operations, a single for loop and print |
| statement? |
| Use no more than two lines. The first line (the for statement) is already |
| written for you. You have to complete the print statement. |
| Note: Using anything related to strings will give a score of . |
| **Input Format** |
| A single line containing integer, . |
| **Constraints** |
| 1<=N<=9 |
| **Output Format** |
| Print lines as explained above. |
| **Sample Input** |
| 5 |
| **Sample Output** |
| 1 |
| 22 |
| 333 |
| 4444 |
| 32. | You are given a positive integer . |
|  | Your task is to print a palindromic triangle of size . |
|  | For example, a palindromic triangle of size is: |
|  | 1 |
|  | 121 |
|  | 12321 |
|  | 1234321 |
|  | 123454321 |
|  | You can't take more than two lines. The first line (a for-statement) is already |
|  | written for you. |
|  | You have to complete the code using exactly one print statement. |
|  | Note:Using anything related to strings will give a score of . |
|  | Using more than one for-statement will give a score of . |
|  | **Input Format** |
|  | A single line of input containing the integer . |
|  | **Constraints** |
|  | 0<N<10 |
|  | **Output Format** |
|  | Print the palindromic triangle of size as explained above. |
|  | **Sample Input** |
|  | 5 |
|  | **Sample Output** |

|  |  |
| --- | --- |
|  | 1 |
| 121 |
| 12321 |
| 1234321 |
| 123454321 |
| 33. | The provided code stub will read in a dictionary containing key/value pairs of |
|  | name:[marks] for a list of students. Print the average of the marks array for |
|  | the student name provided, showing 2 places after the decimal. |
|  | **Input Format** |
|  | The first line contains the integer , N the number of students' records. The |
|  | next lines contain the names and marks obtained by a student, each value |
|  | separated by a space. The final line contains query\_name, the name of a |
|  | student to query. |
|  | **Constraints** |
|  | 2<=n<=10 |
|  | 0<=marks[I]<=100 |
|  | Length of marks arrays=3 |
|  | **Output Format** |
|  | Print one line: The average of the marks obtained by the particular student |
|  | correct to 2 decimal places. |
|  | **Sample Input 0** |
|  | 3 |
|  | Krishna 67 68 69 |
|  | Arjun 70 98 63 |
|  | Malika 52 56 60 |
|  | Malika |
|  | **Sample Output 0** |
|  | 56.00 |
|  | **Sample Input 1** |
|  | 2 |
|  | Harsh 25 26.5 28 |
|  | Anurag 26 28 30 |
|  | Harsh |
|  | **Sample Output 1** |
|  | 26.50 |
| 34. | There is an array of integers. There are also 2 disjoint sets, A and , B each containing integers. You like all the integers in set A and dislike all the integers in set B . Your initial happiness is 0. For each integer in the array, if I belongs to A, you add i to your happiness. If ,I belongs to B you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.  Note: Since A and B are sets, they have no repeated elements. However, the  array might contain duplicate elements. |

|  |  |
| --- | --- |
|  | **Constraints**  1<=n<=10^5  1<=m<=10^5  1<=Any integer in the input<=10^9  **Input Format**  The first line contains integers n and m separated by a space. The second line contains n integers, the elements of the array.  The third and fourth lines contain m integers, A and B, respectively.  **Output Format**  Output a single integer, your total happiness.  **Sample Input**  3 2  1 5 3  3 1  5 7  **Sample Output**  1 |
| 35. | You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.  **Function Description**  Complete the swap\_case function in the editor below. swap\_case has the following parameters:  string s: the string to modify  **Returns**  string: the modified string  **Input Format**  A single line containing a string s.  **Constraints**  0<len(s)<=1000  **Sample Input 0**  NriiT PresEnts CodinG ClaSses  **Sample Output 0**  nRIIt pRESeNTS cODINg cLAsSES |
| 36. | Consider the following:  A string,s , of length n where s=C0C1….Cn-1. An integer,k , where k is a factor of n.  We can split s into n/k substrings where each subtring, ti , consists of a contiguous block of characters in s. Then, use each ti to create string ui such that:  The characters in ui are a subsequence of the characters in ti.  Any repeat occurrence of a character is removed from the string such that each character in ui occurs exactly once. In other words, if the character at some index j in ti occurs at a previous index < j in ti, then do not include the character in string ui .  Given s and k , print n/k lines where each line denotes string .  **Example** |

|  |  |
| --- | --- |
|  | s=’AAABCADDE’ K=3  There are three substrings of length 3 to consider: 'AAA', 'BCA' and 'DDE'. The first substring is all 'A' characters, so u1=’A’. The second substring has all distinct characters, so u2=’BCA’ . The third substring has different characters, so u3=’DE’ . Note that a subsequence maintains the original order of characters encountered. The order of characters in each subsequence shown is important.  **Function Description**  Complete the merge\_the\_tools function in the editor below. merge\_the\_tools has the following parameters:  string s: the string to analyze  int k: the size of substrings to analyze  **Prints**  Print each subsequence on a new line. There will be n/k of them. No return value is expected.  Input Format  The first line contains a single string, s.  The second line contains an integer,k , the length of each substring.  **Constraints**  1<=n<=10^4, where n is the length of s 1<=k<=n  It is guaranteed that n is a multiple of k .  **Sample Input**  STDIN Function    AABCAAADA s = 'AABCAAADA' 3 k = 3  **Sample Output**  AB CA AD |
| 37. | You are given a string S and width w.  Your task is to wrap the string into a paragraph of width w.  **Input Format**  The first line contains a string, S.  The second line contains the width, w.  **Constraints**  0 < len(S) < 1000  0 < w < len(S)  **Output Format**  Print the text wrapped paragraph. **Sample Input 0** ABCDEFGHIJKLIMNOQRSTUVWXYZ |

|  |  |
| --- | --- |
|  | 4  **Sample Output 0**  ABCD EFGH IJKL IMNO QRST UVWX YZ |
| 38. | You are given a complex z. Your task is to convert it to polar coordinates.  **Input Format**  A single line containing the complex number z. Note: complex() function can be used in python to convert the input as a complex number.  **Constraints**  Given number is a valid complex number.  **Output Format**  Output two lines:  The first line should contain the value of r. The second line should contain the value of q. **Sample Input**  1+2j  **Sample Output**  2.23606797749979  1.1071487177940904 |
| 39. | Mr. Vincent works in a door mat manufacturing company. One day, he designed a new door mat with the following specifications:  Mat size must be N X M. (N is an odd natural number, and M is 3 times N.) The design should have ‘WELCOME’ written in the center.  The design pattern should only use |, . and – characters.  **Sample Designs**  Size: 7 x 21  .|.  .|..|..|.  ---.|..|..|..|..|.---  -------WELCOME-------  ---.|..|..|..|..|.---  .|..|..|.  .|.  Size: 11 x 33  .|.  .|..|..|.  .|..|..|..|..|.  ------.|..|..|..|..|..|..|.------ |

|  |  |
| --- | --- |
|  | ---.|..|..|..|..|..|..|..|..|.---  WELCOME  ---.|..|..|..|..|..|..|..|..|.---  ------.|..|..|..|..|..|..|.------  .|..|..|..|..|.  .|..|..|.  .|.  **Input Format**  A single line containing the space separated values of N and M.  **Constraints**  5 < N < 101  15 < M < 303  **Output Format**  Output the design pattern.  **Sample Input**  9 27  **Sample Output**  . | .  . | . . | . . | .  ------ . | . . | . . | . . | . . |. ------  --- . | . . | . . | . . | . . | . . | . . | . ---  WELCOME  --- . | . . | . . | . . | . . | . . | . . | . ---  ------ . | . . | . . | . . | . . | . ------  . | . . |. . | .  . | . |
| 40. | You are given a two lists A and B. Your task is to compute their cartesian |
|  | product A x B. |
|  | **Example** |
|  | A = [1, 2] |
|  | B = [3, 4] |
|  | A x B = [(1, 3), (1, 4), (2, 3), (2, 4)] |
|  | Note: A and B are sorted lists, and the cartesian product’s tuples should be |
|  | output in sorted order. |
|  | **Input Format** |
|  | The first line contains the space separated elements of list A. |
|  | The second line contains the space separated elements of list B. |
|  | Both lists have no duplicate integer elements. |
|  | **Constraints** |
|  | 0 < A < B |
|  | 0 < B < 30 |
|  | **Output Format** |

|  |  |
| --- | --- |
|  | Output the space separated tuples of the cartesian product.  **Sample Input**  1 2  3 4  **Sample Output**  (1, 3) (1, 4) (2, 3) (2, 4) |
| 41. | Raghu is a shoe shop owner. His shop has X number of shoes. He has a list containing the size of each shoe he has in his shop.  There are N number of customers who are willing to pay xi amount of money only if they get the shoe of their desired size.  Your task is to compute how much money Raghu earned.  **Input Format**  The first line contains X, the number of shoes.  The second line contains the space separated list of all the shoe sizes in the shop.  The third line contains N, the number of customers.  The next N lines contain the space separated values of the shoe size desired by the customer and xi, the price of the shoe.  **Constraints**  0 < X < 10^3  0 < N <= 10^3  20 < xi < 100  2 < shoe size < 20  **Output Format**  Print the amount of money earned by Raghu.  **Sample Input**  10  2 3 4 5 6 8 7 6 5 18  6  6 55  6 45  6 55  4 40  18 60  10 50  **Sample Output**  200 |
| 42. | There is a horizontal row of n cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if cube[i] is on top of cube[j] then sideLength|j| => sideLength|i|.  When stacking the cubes, you can only pick up either the leftmost or the rightmost cube each time. Print Yes if it is possible to stack the cubes. Otherwise, print No. |

|  |  |
| --- | --- |
|  | **Example**  blocks = [1, 2, 3, 8, 7] Result: No  After choosing the rightmost element, 7, choose the leftmost element, 1. After than, the choices are 2 and 8. These are both larger than the top block of size 1.  blocks = [1, 2, 3, 7, 8] Result: Yes  Choose blocks from right to left in order to successfully stack the blocks.  **Input Format**  The first line contains a single integer T, the number of test cases. For each test case, there are 2 lines.  The first line of each test case contains n, the number of cubes.  The second line contains n space separated integers, denoting the sideLengths of each cube in that order.  **Constraints**  1 <= T <= 5  1 <= n <= 10^5  1 <= sideLength < 2^31  **Output Format**  For each test case, output a single line containing either Yes or No. Sample Input  STDIN Function    2 T = 2  6 blocks[] size n = 6  4 3 2 1 3 4 blocks = [4, 3, 2, 1, 3, 4]  3 blocks[] size n = 3  1 3 2 blocks = [1, 3, 2]  **Sample Output**  Yes No |
| 43. | You are given a function f(X) = X2. You are also given K lists. The ith list consists of Ni elements.  You have to pick one element from each list so that the value from the equation below is maximized:  S = (f(X1) + f(X2) + . . . + f(Xk) % M  Xi denotes the element picked from the ith list . Find the maximized value Smax obtained. % denotes the modulo operator. |

|  |  |
| --- | --- |
|  | Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.  **Input Format**  The first line contains 2 space separated integers K and M.  The next K lines each contains an integer Ni, denoting the number of elements in the ith list, followed by Ni space separated integers denoting the elements in the list.  **Constraints**  1 <= K <= 7  1 <= M <= 1000  1 <= Ni <= 7  1 <= Magnitude of elements in list <= 109  **Output Format**  Output a single integer denoting the value Smax.  **Sample Input**  3 1000  2 5 4  3 7 8 9  5 5 7 8 9 10  **Sample Output**  206 |
| 44. | Mr. Anant Asankhya is the manager at the INFINITE hotel. The hotel has an infinite amount of rooms.  One fine day, a finite number of tourists come to stay at the hotel. The tourists consist of:  → A Captain.  → An unknown group of families consisting of K members per group where K ≠ 1.  The Captain was given a separate room, and the rest were given one room per group.  Mr. Anant has an unordered list of randomly arranged room entries. The list consists of the room numbers for all of the tourists. The room numbers will appear K times per group except for the Captain’s room.  Mr. Anant needs you to help him find the Captain’s room number.  The total number of tourists or the total number of groups of families is not |

|  |  |
| --- | --- |
|  | known to you.  You only know the value of K and the room number list.  **Input Format**  The first line consists of an integer, K, the size of each group.  The second line contains the unordered elements of the room number list.  **Constraints**  1 < K < 1000  **Output Format**  Output the Captain’s room number.  **Sample Input**  5  1 2 3 6 5 4 4 2 5 3 6 1 6 5 3 2 4 1 2 5 1 4 3 6 8 4 3 1 5 6 2  **Sample Output**  8 |
| 45. | You are given a set A and n other sets.  Your job is to find whether set A is a strict superset of each of the N sets. Print True, if A is a strict superset of each of the N sets. Otherwise, print False.  A strict superset has at least one element that does not exist in its subset.  **Example**  Set ([1 , 3, 4]) is a strict superset of set ([1 , 3]).  Set ([1 , 3, 4]) is not a strict superset of set ([1 , 3, 4]).  Set ([1 , 3, 4]) is not a strict superset of set ([1 , 3, 5]).  **Input Format**  The first line contains the space separated elements of set A. The second line contains integer n, the number of other sets.  The next n lines contains the space separated elements of the other sets.  **Constraints**  0 < len(set(A)) < 50^1 0 < N < 2^1  0 < len(otherSets) < 10^1  **Output Format**  Print True if set A is a strict superset of all other N sets. Otherwise, print False. |

|  |  |
| --- | --- |
|  | **Sample Input 0** |
| 1 2 3 4 5 6 7 8 9 10 11 12 23 45 84 78 |
| 2 |
| 1 2 3 4 5 |
| 100 11 12 |
| **Sample Output 0** |
| False |
| 46. | You are given a spreadsheet that contains a list of N athletes and their |
|  | details (such as age, height, weight and so on). You are required to sort the |
|  | data based on the Kth attribute and print the final resulting table. Follow the |
|  | example given below for better understanding. |
|  | Note that K is indexed from 0 to M – 1, where M is the number of attributes. |
|  | Note: If two attributes are the same for different rows, for example, if two |
|  | atheletes are of the same age, print the row that appeared first in the input. |
|  | **Input Format** |
|  | The first line contains N and M separated by a space. |
|  | The next N lines each contain M elements. |
|  | The last line contains K. |
|  | **Constraints** |
|  | 1 <= N, M <= 1000 |
|  | 0 <= K < M |
|  | Each element <= 1000 |
|  | **Output Format** |
|  | Print the N lines of the sorted table. Each line should contain the space |
|  | separated elements. Check the sample below for clarity. |
|  | **Sample Input 0** |
|  | 5 3 |
|  | 10 2 5 |
|  | 7 1 0 |
|  | 9 9 9 |
|  | 1 23 12 |
|  | 6 5 9 |
|  | 1 |
|  | **Sample Output 0** |
|  | 7 1 0 |
|  | 10 2 5 |
|  | 6 5 9 |
|  | 9 9 9 |
|  | 1 23 12 |

|  |  |
| --- | --- |
| 47. | You are given a string N.  Your task is to verify that N is a floating point number.  In this task, a valid float number must satisfy all of the following requirements:  ->Number can start with +, - or . symbol.  **For example:**  ✔+4.50  ✔-1.0  ✔.5  ✔-.7  ✔+.4  ✖ -+4.5  ->Number must contain at least 1 decimal value.  **For example:**  ✖ 12.  ✔12.0  ->Number must have exactly one . symbol.  ->Number must not give any exceptions when converted using float(N).  **Input Format**  The first line contains an integer T, the number of test cases. The next T line(s) contains a string N.  **Constraints**  0 < T < 10  **Output Format**  Output True or False for each test case.  **Sample Input 0**  4  4.0O0  -1.00  +4.54  SomeRandomStuff  **Sample Output 0**  False True True False |
| 48. | The National University conducts an examination of N students in X subjects. Your task is to compute the average scores of each student.  Average Score = Sum of Scores obtained in all subjects by a student / Total |

|  |  |
| --- | --- |
|  | number of students  The format for the general mark sheet is:  Student ID → 1 2 3 4 5 Subject 1 | 89 90 78 93 80  Subject 2 | 90 91 85 88 86  Subject 3 | 91 92 83 89 90.5  | Average 90 91 82 90 85.5  **Input Format**  The first line contains N and X separated by a space.  The next X lines contains the space separated marks obtained by students in a particular subject.  **Output Format**  Print the averages of all students on separate lines. The averages must be correct up to 1 decimal place.  **Sample Input**  5 3  89 90 78 93 80  90 91 85 88 86  91 92 83 89 90.5  **Sample Output**  90.0  91.0  82.0  90.0  85.5 |
| 49. | You are given a string S.  S contains alphanumeric characters only.  Your task is to sort the string in the following manner:  All sorted lowercase letters are ahead of uppercase letters. All sorted uppercase letters are ahead of digits.  All sorted odd digits are ahead of sorted even digits.  **Input Format**  A single line of input contains the string S.  **Constraints**  0 < len(S) < 1000  **Output Format**  Output the sorted string S. |

|  |  |
| --- | --- |
|  | **Sample Input**  Sorting1234  **Sample Output**  ginortS1324 |
| 50. | Rupal has a huge collection of country stamps. She decided to count the total number of distinct country stamps in her collection. She asked for your help. You pick the stamps one by one from a stack of N country stamps.  Find the total number of distinct country stamps.  **Input Format**  The first line contains an integer N, the total number of country stamps. The next N lines contains the name of the country where the stamp is from. **Constraints**  0 < N < 1000  **Output Format**  Output the total number of distinct country stamps on a single line.  **Sample Input**  7  UK  China USA  France  New Zealand UK  France  **Sample Output**  5 |
| 51. | A newly opened multinational brand has decided to base their company logo on the three most common characters in the company name. They are now trying out various combinations of company names and logos based on this condition. Given a string s , which is the company name in lowercase letters, your task is to find the top three most common characters in the string.  Print the three most common characters along with their occurrence count. Sort in descending order of occurrence count.  If the occurrence count is the same, sort the characters in alphabetical order.  For example, according to the conditions described above, GOOGLE would have it's logo with the letters G,O,E .  Input Format  A single line of input containing the string S.  **Constraints**  ->3<len(S)<=10^4  ->S has at least distinct characters |

|  |  |
| --- | --- |
|  | **Output Format**  Print the three most common characters along with their occurrence count each on a separate line.  Sort output in descending order of occurrence count.  If the occurrence count is the same, sort the characters in alphabetical order.  **Sample Input 0**  Aabbbccde  **Sample Output 0**  b 3  a 2  c 2 |
| 52. | You are given a string S. Suppose a character ‘c’ occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X,c) in the string.  For a better understanding of the problem, check the explanation.  **Input Format**  A single line of input consisting of the string S.  **Output Format**  A single line of output consisting of the modified string.  **Constraints**  All the characters of S denote integers between 0 and 9. 1<=|S|<=10^4  **Sample Input**  1222311  **Sample Output**  (1, 1) (3, 2) (1, 3) (2, 1)  **Explanation**  First, the character 1 occurs only once. It is replaced by (1,1). Then the character 2 occurs three times, and it is replaced by (3,2) and so on. |
| 53. | You and Fredrick are good friends. Yesterday, Fredrick received N credit cards from ABCD Bank. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!  A valid credit card from ABCD Bank has the following characteristics:   * It must start with a 4,5 or 6 . * It must contain exactly 16 digits. * It must only consist of digits (0-9). * It may have digits in groups of 4, separated by one hyphen "-". * It must NOT use any other separator like ' ' , '\_', etc. * It must NOT have 4 or more consecutive repeated digits. |

|  |  |
| --- | --- |
|  | **Examples:**  **Valid Credit Card Numbers**  4253625879615786  4424424424442444  5122-2368-7954-3214  **Invalid Credit Card Numbers**  42536258796157867 #17 digits in card number → Invalid 4424444424442444 #Consecutive digits are repeating 4 or more times  → Invalid  5122-2368-7954 - 3214 #Separators other than '-' are used → Invalid 44244x4424442444 #Contains non digit characters → Invalid 0525362587961578 #Doesn't start with 4, 5 or 6 → Invalid  **Input Format**  The first line of input contains an integer . The next lines contain credit card numbers.  **Constraints**  0 < N < 100  **Output Format**  Print 'Valid' if the credit card number is valid. Otherwise, print 'Invalid'. Do not print the quotes.  **Sample Input**  6  4123456789123456  5123-4567-8912-3456  61234-567-8912-3456  4123356789123456  5133-3367-8912-3456  5123 - 3567 - 8912 - 3456  **Sample Output**  Valid Valid Invalid Valid Invalid Invalid |
| 54. | You are the manager of a supermarket.  You have a list of items together with their prices that consumers bought on a particular day.  Your task is to print each item\_name and net\_price in order of its first  occurrence. |

|  |  |
| --- | --- |
|  | item\_name = Name of the item.  net\_price = Quantity of the item sold multiplied by the price of each item.  **Input Format**  The first line contains the number of items, .  The next lines contains the item's name and price, separated by a space.  **Constraints**  0 < N <=100  **Output Format**  Print the item\_name and net\_price in order of its first occurrence.  **Sample Input**  9  BANANA FRIES 12  POTATO CHIPS 30  APPLE JUICE 10  CANDY 5  APPLE JUICE 10  CANDY 5  CANDY 5  CANDY 5  POTATO CHIPS 30  **Sample Output**  BANANA FRIES 12  POTATO CHIPS 60  APPLE JUICE 20  CANDY 20 |
| 55. | you are given two complex numbers, and you have to print the result of their addition, subtraction, multiplication, division and modulus operations. The real and imaginary precision part should be correct up to two decimal places.  **Input Format**  One line of input: The real and imaginary part of a number separated by a space.  **Output Format**  For two complex numbers C and D, the output should be in the following sequence on separate lines:  -> C+D  ->C-D  ->C\*D  ->C/D  ->mod( C )  ->mod(D) |

|  |  |
| --- | --- |
|  | For complex numbers with non-zero real (A) and (B) complex part, the output should be in the following format: A+Bi  Replace the plus symbol (+) with a minus symbol (-) when B<0.  For complex numbers with a zero complex part i.e. real numbers, the output should be:  A+0.00i  For complex numbers where the real part is zero and the complex part is non-zero, the output should be:  0.00+Bi  **Sample Input**  2 1  5 6  **Sample Output**  7.00+7.00i  -3.00-5.00i 4.00+17.00i  0.26-0.11i  2.24+0.00i  7.81+0.00i |
| 56. | You are given a string, and you have to validate whether it's a valid Roman numeral. If it is valid, print True. Otherwise, print False. Try to create a regular expression for a valid Roman numeral.  **Input Format**  A single line of input containing a string of Roman characters.  **Output Format**  Output a single line containing True or False according to the instructions above.  **Constraints**  The number will be between 1 and 3999 (both included).  **Sample Input**  CDXXI  **Sample Output**  True |
| 57. | You are given some input, and you are required to check whether they are valid mobile numbers.  A valid mobile number is a ten digit number starting with a 7,8 or 9.  **Input Format**  The first line contains an integer N , the number of inputs. N lines follow, each containing some string.  **Constraints**  1<= N <=10  2<=len(Number)<=15  **Output Format** |

|  |  |
| --- | --- |
|  | For every string listed, print "YES" if it is a valid mobile number and "NO" if it is not on separate lines. Do not print the quotes.  **Sample Input**  2  9587456281  1252478965  **Sample Output**  YES NO |
| 58. | You are given two sets, A and B. |
|  | Your job is to find whether set A is a subset of set B . |
|  | If set A is subset of set B , print True. |
|  | If set A is not a subset of set B , print False. |
|  | **Input Format** |
|  | The first line will contain the number of test cases, T . |
|  | The first line of each test case contains the number of elements in set A. |
|  | The second line of each test case contains the space separated elements of |
|  | set A. |
|  | The third line of each test case contains the number of elements in set B. |
|  | The fourth line of each test case contains the space separated elements of |
|  | set B. |
|  | **Constraints** |
|  | 0 < T < 21 |
|  | 0 < Number of elements in each set < 1001 |
|  | **Output Format** |
|  | Output True or False for each test case on separate lines. |
|  | **Sample Input** |
|  | 3 |
|  | 5 |
|  | 1 2 3 5 6 |
|  | 9 |
|  | 9 8 5 6 3 2 1 4 7 |
|  | 1 |
|  | 2 |
|  | 5 |
|  | 3 6 5 4 1 |
|  | 7 |
|  | 1 2 3 5 6 8 9 |
|  | 3 |
|  | 9 8 2 |
|  | **Sample Output** |
|  | True |
|  | False |

|  |  |
| --- | --- |
|  | False |
| 59. | You are given a list of N lowercase English letters. For a given integer K, you can select any K indices (assume 1-based indexing) with a uniform probability from the list.  Find the probability that at least one of the K indices selected will contain the letter: 'a'.  **Input Format**  The input consists of three lines. The first line contains the integer N , denoting the length of the list. The next line consists of N space-separated lowercase English letters, denoting the elements of the list.  The third and the last line of input contains the integer K, denoting the number of indices to be selected.  **Output Format**  Output a single line consisting of the probability that at least one of the K indices selected contains the letter:'a'.  Note: The answer must be correct up to 3 decimal places.  **Constraints** 1<=N<=10 I<=K<=N  All the letters in the list are lowercase English letters.  **Sample Input**  4  a a c d 2  **Sample Output**  0.8333 |
| 60. | ABCXYZ company has up to 100 employees.  The company decides to create a unique identification number (UID) for each of its employees.  The company has assigned you the task of validating all the randomly generated UIDs.  A valid UID must follow the rules below:  It must contain at least 2 uppercase English alphabet characters. It must contain at least 3 digits (0 - 9).  It should only contain alphanumeric characters ( a-z ,A -Z & 0 -9 ). No character should repeat.  There must be exactly 10 characters in a valid UID.  **Input Format**  The first line contains an integer , the number of test cases. The next lines contains an employee's UID.  **Output Format**  For each test case, print 'Valid' if the UID is valid. Otherwise, print 'Invalid', on separate lines. Do not print the quotation marks. |

|  |  |
| --- | --- |
|  | **Sample Input** 2 B1CD102354 B1CDEF2354  **Sample Output**  Invalid Valid |
| 61. | Ravi belongs to a very rich family which owns many gold mines. Today, he |
|  | brought N gold coins and decided to form a triangle using these coins. Isn't it |
|  | strange? |
|  | Ravi has a unusual way of forming a triangle using gold coins, which is |
|  | described as follows: |
|  | He puts 1 coin in the 1st row. |
|  | then puts 2 coins in the 2nd row. |
|  | then puts 3 coins in the 3rd row. |
|  | and so on as shown in the given figure. |
|  |  |
|  | Ravi is interested in forming a triangle with maximum possible height using |
|  | at most N coins. Can you tell him the maximum possible height of the |
|  | triangle? |
|  | **Input** |
|  | The first line of input contains a single integer T denoting the number of test |
|  | cases. |
|  | The first and the only line of each test case contains an integer N denoting |
|  | the number of gold coins Chef has. |
|  | **Output** |
|  | For each test case, output a single line containing an integer corresponding |
|  | to the maximum possible height of the triangle that Chef can get. |
|  | **Constraints** |
|  | 1 ≤ T ≤ 100 |
|  | 1 ≤ N ≤ 10^9 |
|  | **Sample Input : Sample Output:** |
|  | 3 2 |
|  | 3 2 |
|  | 5 3 |

|  |  |
| --- | --- |
|  | 7 |
| 62. | You have a binary string S of length N. In one operation you can select a |
|  | substring of S and reverse it. For example, on reversing the substring |
|  | S[2,4] for S=11000, we change 11000→10010. |
|  | Find the minimum number of operations required to sort this binary string. |
|  | It can be proven that the string can always be sorted using the above |
|  | operation finite number of times. |
|  | **Input Format** |
|  | The first line of input will contain a single integer T, denoting the number of |
|  | test cases. |
|  | Each test case consists of 22 lines of input. |
|  | The first line of each test case contains a single integer N — the length of the |
|  | binary string. |
|  | The second line of each test case contains a binary string S of length N. |
|  | **Output Format** |
|  | For each test case, output on a new line — the minimum number of |
|  | operations required to sort the binary string. |
|  | **Constraints** |
|  | 1≤T≤2⋅ 10^5 |
|  | 1≤N≤2⋅ 10^5 |
|  | Sum of N over all test cases does not exceed 106106. |
|  | String S consists of only '00's and '11's. |
|  | **Sample Input: Sample Output:** |
|  | 4 0 |
|  | 3 1 |
|  | 000 2 |
|  | 4 2 |
|  | 1001 |
|  | 4 |
|  | 1010 |
|  | 6 |
|  | 010101 |
| 63. | A binary string is called alternating if no two adjacent characters of the string are equal. Formally, a binary string T of length M is called alternating if Ti !=Ti+1 for each 1≤i<M.  For example, 0, 1, 01, 10, 101, 010, 1010 are alternating strings  while 11, 001, 1110 are not.  You are given a binary string S of length N. You would like to rearrange the characters of S such that the length of the longest alternating substring of S is maximum. Find this maximum value.  A binary string is a string that consists of characters 0 and 1. A string a is a [substring](https://en.wikipedia.org/wiki/Substring) of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.  **Input Format** |

|  |  |
| --- | --- |
|  | The first line of input contains an integer T, denoting the number of test cases. The T test cases then follow:  The first line of each test case contains an integer N.  The second line of each test case contains the binary string S.  **Output Format**  For each test case, output the maximum possible length of the longest alternating substring of S after rearrangement.  **Constraints**  1≤T≤10^4  1≤N≤10^5  S contains only the characters 0 and 1.  Sum of N over all test cases does not exceed 2⋅ 10^5.  **Sample Input : Sample Output:**  4 3  3 4  110 1  4 5  1010  4  0000  7  1101101 |
| 64. | There are 33 hidden numbers A,B,C.  You somehow found out the values of min(A,B),min(B,C), and min(C,A). Determine whether there exists any tuple (A,B,C) that satisfies the given values of min(A,B),min(B,C),min(C,A).  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  The first and only line of each test case contains 33 space-separated integers denoting the values of min(A,B),min(B,C), and min(C,A).  **Output Format**  For each test case, output YES if there exists any valid tuple (A,B,C), and NO otherwise.  You can print each letter of the output in any case. For example YES, yes, yEs will all be considered equivalent.  **Constraints**  1≤T≤1000  1≤min(A,B),min(B,C),min(C,A)≤10  **Sample Input: Sample Output:**  3 YES  5 5 5 NO  2 3 4 YES |

|  |  |
| --- | --- |
|  | 2 2 4 |
| 65. | Initially, John is at coordinate 00 on X-axis. For each i=1,2,…,N in order, John |
|  | does the following: |
|  | If John is at a non-negative coordinate, he moves i steps backward (i.e, his |
|  | position's coordinate decreases by i), otherwise he moves i steps forward |
|  | (i.e, his position's coordinate increases by i). |
|  | You are given the integer N. Find the final position of Chef on the X-axis |
|  | after N operations. |
|  | **Input Format** |
|  | The first line of input contains an integer T, denoting the number of test |
|  | cases. The T test cases then follow: |
|  | The first and only line of each test case contains an integer N. |
|  | **Output Format** |
|  | For each test case, output in a single line the final position of Chef on the X- |
|  | axis after N operations. |
|  | **Constraints** |
|  | 1≤T≤10^5 |
|  | 1≤N≤10^9 |
|  | **Sample Input: Sample Output:** |
|  | 3 -1 |
|  | 1 1 |
|  | 2 -2 |
|  | 3 |
| 66. | Ram wants to buy a new laptop. However, he is confused about which laptop to buy out of 10 different laptops. He asks his N friends for their recommendation. The ith friend recommends the Chef to buy the Ai th laptop (1≤Ai≤10).  Ram will buy the laptop which is recommended by maximum number of friends. Determine which laptop Chef buys.  Print CONFUSED if there are multiple laptops having maximum number of recommendations.  **Input Format**  The first line contains a single integer T - the number of test cases. Then the test cases follow.  The first line of each test case contains an integer N - the number of Chef's friends.  The second line of each test case contains N space-separated integers A1 ,A2 ,…,AN where Ai denotes the recommendation of the ith friend.  **Output Format**  For each test case, output in a single line, the laptop which has the maximum number of recommendations. Print CONFUSED if there are multiple laptops having maximum number of recommendations.  You may print each character of CONFUSED in uppercase or lowercase (for example, Confused, coNFused, CONFused will be considered identical).  **Constraints** |

|  |  |  |
| --- | --- | --- |
|  | 1≤T≤200 |  |
| 1≤N≤1000 |  |
| 1≤Ai ≤10 |  |
| **Sample Input:** | **Sample Output:** |
| 4 | 4 |
| 5 | 6 |
| 4 4 4 2 1 | CONFUSED |
| 7 | CONFUSED |
| 1 2 3 4 5 6 6 |  |
| 6 |  |
| 2 2 3 3 10 8 |  |
| 4 |  |
| 7 7 8 8 |  |
| 67. | Hari wants to store some important numerical data on his personal | |
|  | computer. He is using a new data type that can store values only | |
|  | from 00 till N both inclusive. If this data type receives a value greater | |
|  | than N then it is cyclically converted to fit into the range 00 to N. For | |
|  | **Example:** | |
|  | Value N+1 will be stored as 00. | |
|  | Value N+2 will be stored as 11. | |
|  | and so on... | |
|  | Given X, the value chef wants to store in this new data type. Determine what | |
|  | will be the actual value in memory after storing X. | |
|  | **Input Format** | |
|  | First line will contain T, number of testcases. Then the testcases follow. | |
|  | Each testcase contains a single line of input, two space separated | |
|  | integers N,X - the maximum value a data type can store and the value Chef | |
|  | wants to store in the data type respectively. | |
|  | **Output Format** | |
|  | For each testcase, output in a single line the value which will be actually | |
|  | stored in memory. | |
|  | **Constraints** | |
|  | 1≤T≤3000 | |
|  | 1≤N≤50 | |
|  | 0≤X≤50 | |
|  | **Sample Input: Sample Output:** | |
|  | 5 0 | |
|  | 15 0 10 | |
|  | 15 10 0 | |
|  | 11 12 9 | |
|  | 27 37 49 | |
|  | 50 49 | |
| 68. | It is Anu’s birthday. You know that Anu's favourite number is X. You also  know that Anu loves averages. Therefore you decide it's best to gift | |

|  |  |
| --- | --- |
|  | Chef 33 integers A1 ,A2 ,A3 , such that:  The mean of A1 ,A2 and A3 is X. 1≤A1 ,A2 ,A3 ≤1000.  A1 ,A2 and A3 are distinct.  Output any suitable A1 ,A2 and A3 which you could gift to Anu.  As a reminder, the mean of three numbers P,Q,R is defined as: mean(P,Q,R)=P+Q+R/3.  For  example, (2,3,5)=2+3+5/3=10/3=3.333ˉ, mean(2,2,5)=2+2+5/3 =9/3 =3.  **Input Format**  The first line of input contains a single integer T, denoting the number of test cases. The description of T test cases follows.  The first and only line of each test case contains one integer X — Anu's favourite number.  **Output Format**  For each test case, one line containing 3 space-separated integers — A1 ,A2 , and A3 , which satisfy the given conditions. If there are multiple possible answers you may output any of them.  It can be shown that an answer always exists, under the given constraints.  **Constraints**  1≤T≤100  2≤X≤100  **Sample Input: Sample Output:**  3 1 3 5  3 1 6 8  5 3 5 7  5 |
| 69. | There are N piles where the ith pile consists of Ai stones.  Zack and Ryan are playing a game taking alternate turns with Zack starting first.  In his/her turn, a player can choose any non-empty pile and remove exactly 11 stone from it.  The game ends when exactly 11 pile becomes empty. The player who made the last move wins.  Determine the winner if both players play optimally.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of multiple lines of input.  The first line of each test case contains a single integer N denoting the number of piles.  Next line contains N space-separated integers A1 ,A2 ,…,AN - denoting the number of stones in each pile.  **Output Format**  For each test case, output Zack if Zack wins the game, otherwise output Ryan. |

|  |  |
| --- | --- |
|  | Note that the output is case-insensitive i.e. ZACK, Zack, ZaCk, and zack are all |
| considered the same. |
| **Constraints** |
| 1≤T≤1000 |
| 1≤N≤10^5 |
| 1≤Ai ≤10^9 |
| Sum of N over all test cases does not exceed 2⋅ 1052⋅ 105. |
| **Sample Input: Sample Output:** |
| 3 Ryan |
| 2 Ryan |
| 2 2 Zack |
| 1 |
| 10 |
| 3 |
| 1 5 6 |
| 70. | You are given a binary string S of length N. You can perform the following |
|  | operation on S: |
|  | Pick any set of indices such that no two picked indices are adjacent. |
|  | Flip the values at the picked indices (i.e. change 00 to 11 and 11 to 00). |
|  | For example, consider the string S=1101101. |
|  | If we pick the indices {1,3,6}, then after flipping the values at picked indices, |
|  | we will get 1101101-->0111111. |
|  | Note that we cannot pick the set {2,3,5} since 2 and 3 are adjacent indices. |
|  | Find the minimum number of operations required to convert all the |
|  | characters of S to 00. |
|  | **Input Format** |
|  | The first line contains a single integer T - the number of test cases. Then the |
|  | test cases follow. |
|  | The first line of each test case contains an integer N - the length of the |
|  | binary string S. |
|  | The second line of each test case contains a binary string S of length N. |
|  | **Output Format** |
|  | For each test case, output the minimum number of operations required to |
|  | convert all the characters of S to 00. |
|  | **Constraints** |
|  | 1≤T≤100 |
|  | 1≤N≤100 |
|  | **Sample Input: Sample Output:** |
|  | 3 1 |
|  | 6 0 |
|  | 101001 2 |
|  | 5 |
|  | 00000 |
|  | 3 |

|  |  |
| --- | --- |
|  | 111 |
| 71. | Swathi is fan of pairs and he likes all things that come in pairs. He even has a doll collection in which the dolls come in pairs. One day while going through his collection he found that there are odd number of dolls. Someone had stolen a doll!!!  Help Swathi find which type of doll is missing..  **Input**  The first line contains an integer T, the number of test cases.  The first line of each test case contains an integer N, the number of dolls. The next N lines are the types of dolls that are left.  **Output**  For each test case, display the type of doll that doesn't have a pair, in a new line.  **Constraints**  1<=T<=10  1<=N<=100000 (10^5)  0<=type<=100000  **Sample Input: Sample Output:**  1 2  3  1  2  1 |
| 72. | You are given an array A of size N. In one operation, you can do the following:  Select indices i and j (I !=j) and set Ai =Aj .  Find the minimum number of operations required to make all elements of the array equal.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of multiple lines of input.  The first line of each test case contains an integer N — the size of the array. The next line contains N space-separated integers, denoting the array A. **Output Format**  For each test case, output on a new line, the minimum number of operations required to make all elements of the array equal.  **Constraints**  1≤T≤1000  1≤N≤2⋅ 10^5  1≤Ai ≤N  The sum of N over all test cases won't exceed 2⋅ 10^5.  **Sample Input: Sample Output:** |

|  |  |
| --- | --- |
|  | 3 2  3 2  1 2 3 3  4  2 2 3 1  4  3 1 2 4 |
| 73. | There are N different types of colours numbered from 11 to N. Chef has Ai balls having colour i, (1≤i≤N).  Chef will arrange some boxes and put each ball in exactly one of those boxes.  Find the minimum number of boxes Chef needs so that no box contains two balls of same colour.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases. The description of the test cases follows.  The first line of each test case contains a single integer N, denoting the number of colors.  The second line of each test case contains N space-separated integers A1 ,A2 ,…,AN — denoting the number of balls having colour i. **Output Format**  For each test case, output the minimum number of boxes required so that no box contains two balls of same colour.  Constraints 1≤T≤1000  2≤N≤100  1≤Ai ≤10^5  **Sample Input : Sample Output:**  3 8  2 15  8 5 4  3  5 10 15  4  4 4 4 4 |
| 74. | Devu has n weird friends. Its his birthday today, so they thought that this is the best occasion for testing their friendship with him. They put up conditions before Devu that they will break the friendship unless he gives them a grand party on their chosen day. Formally, ith friend will break his friendship if he does not receive a grand party on dith day.  Devu despite being as rich as Gatsby, is quite frugal and can give at most one grand party daily. Also, he wants to invite only one person in a party. So he just wonders what is the maximum number of friendships he can save. Please help Devu in this tough task !! |

|  |  |
| --- | --- |
|  | **Input**  The first line of the input contains an integer T denoting the number of test cases. The description of T test cases follows.  First line will contain a single integer denoting n.  Second line will contain n space separated integers where ith integer corresponds to the day dith as given in the problem.  **Output**  Print a single line corresponding to the answer of the problem.  **Constraints**  1 ≤ T ≤ 10^4  1 ≤ n ≤ 50  1 ≤ di ≤ 100  **Sample Input: Sample Output:**  2 2  2 1  3 2  2  1 1 |
| 75. | You are given an array A of length N. An element X is said to be dominant if the frequency of X in A is strictly greater than the frequency of any other element in the A.  For example, if A=[2,1,4,4,4] then 44 is a dominant element since its frequency is higher than the frequency of any other element in A.  Find if there exists any dominant element in A.  **Input Format**  The first line of input contains a single integer T — the number of test cases. Then the test cases follow.  The first line of each test case contains an integer N — the size of the array A.  The second line of each test case contains N space-separated integers A1 ,A2 ,…,AN denoting the array A.  **Output Format**  For each test case, output YES if there exists any dominant element in A. Otherwise, output NO.  You may print each character of YES and NO in uppercase or lowercase (for example, yes, yEs, Yes will be considered identical).  **Constraints**  1≤T≤500  1≤N≤1000  1≤Ai ≤N  **Sample Input: Sample Output:**   1. YES 2. NO |

|  |  |
| --- | --- |
|  | 2 2 2 2 2 YES  4 NO  1 2 3 4  4  3 3 2 1  6  1 1 2 2 3 4 |
| 76. | Siva wants to become fit for which he decided to walk to the office and return home by walking. It is known that Siva's office is X km away from his home.  If his office is open on 5 days in a week, find the number of kilometers Siva travels through office trips in a week.  **Input Format**  First line will contain T, number of test cases. Then the test cases follow. Each test case contains of a single line consisting of single integer X. **Output Format**  For each test case, output the number of kilometers Siva travels through office trips in a week.  **Constraints**  1≤T≤10  1≤X≤10  **Sample Input: Sample Output:**  4 10  1 30  3 70  7 100  10 |
| 77. | Alex has X 5 rupee coins and Y 10 rupee coins. Alex goes to a shop to buy chocolates for Chefina where each chocolate costs Z rupees. Find the maximum number of chocolates that Alex can buy for Chefina.  **Input Format**  The first line contains a single integer T — the number of test cases. Then the test cases follow.  The first and only line of each test case contains three integers X, Y and Z — the number of 5 rupee coins, the number of 10 rupee coins and the cost of each chocolate.  **Output Format**  For each test case, output the maximum number of chocolates that Alex can buy for Chefina.  **Constraints**  1≤T≤100  1≤X,Y,Z≤1000  **Sample Input: Sample Output:**  4 15  10 10 10 3 |

|  |  |
| --- | --- |
|  | 3 1 8 16  8 1 3 0  4 4 1000 |
| 78. | Arun has started working at the candy store. The store has 100 chocolates in total.  Arun’s daily goal is to sell X chocolates. For each chocolate sold, he will get 1 rupee. However, if Arun exceeds his daily goal, he gets 2 rupees per chocolate for each extra chocolate.  If Arun sells Y chocolates in a day, find the total amount he made.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of two space-separated integers X and Y — the daily goal of Arun, and the number of chocolates he actually sells.  **Output Format**  For each test case, output on a new line the total amount Arun made in a day.  **Constraints**  1≤T≤100  1≤X,Y≤10  **Sample Input: Sample Output:**  4 1  3 1 5  5 5 10  4 7 4  2 3 |
| 79. | Alice likes numbers which are even, and are a multiple of 7. Bob likes numbers which are odd, and are a multiple of 9. Alice, Bob, and Charlie find a number A.  If Alice likes A, Alice takes home the number. If Bob likes A, Bob takes home the number.  If both Alice and Bob don't like the number, Charlie takes it home. Given A, find who takes it home.  Note: You can prove that there is no integer A such that both Alice and Bob like it.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of a single integer, A.  **Output Format**  For each test case, output on a new line who takes the number home - "Alice", "Bob", or "Charlie".  You may print each character in uppercase or lowercase. For example, Alice, alice, aLiCe, and ALICE are all considered identical.  **Constraints** |

|  |  |  |
| --- | --- | --- |
|  | 1≤T≤100  1≤A≤1000  **Sample Input:**  8  7  14  21  18  27  63  126  8 | **Sample Output:**  Charlie Alice Charlie Charlie Bob Bob Alice Charlie |
| 80. | In Chefland, a valid phone number consists of 5 digits with no leading zeros. | |
|  | For example, 98765,10000, and 71023 are valid phone numbers | |
|  | while 04123,9231, and 872310 are not. | |
|  | Chef went to a store and purchased N items, where the cost of each item | |
|  | is X. | |
|  | Find whether the total bill is equivalent to a valid phone number. | |
|  | **Input Format** | |
|  | The first line of input will contain a single integer T, denoting the number of | |
|  | test cases. | |
|  | Each test case consists of two space-separated integers N and X — the | |
|  | number of items Chef bought and the cost per item. | |
|  | **Output Format** | |
|  | For each test case, output on a new line, YES, if the total bill is equivalent to | |
|  | a valid phone number and NO otherwise. | |
|  | Each character of the output may be printed in either uppercase or | |
|  | lowercase. That is, the strings NO, no, nO, and No will be treated as | |
|  | equivalent. | |
|  | **Constraints** | |
|  | 1≤T≤100 | |
|  | 1≤N,X≤1000 | |
|  | **Sample Input: Sample Output:** | |
|  | 4 YES | |
|  | 25 785 NO | |
|  | 402 11 YES | |
|  | 100 100 NO | |
|  | 333 333 | |
| 81. | Kiran's phone has a total storage of S MB. Also, Chef has 2 apps already installed on his phone which occupy X MB and Y MB respectively.  He wants to install another app on his phone whose memory requirement is Z MB. For this, he might have to delete the apps already installed on his phone. Determine the minimum number of apps he has to delete from his  phone so that he has enough memory to install the third app. | |

|  |  |
| --- | --- |
|  | **Input Format**  The first line contains a single integer T — the number of test cases. Then the test cases follow.  The first and only line of each test case contains four integers S,X,Y and Z — the total memory of Kiran's phone, the memory occupied by the two already installed apps and the memory required by the third app.  **Output Format**  For each test case, output the minimum number of apps Chef has to delete from his phone so that he can install the third app.  **Constraints**  1≤T≤1000  1≤S≤500  1≤X≤Y≤S X+Y≤S  Z≤S  **Sample Input: Sample Output:**  4 0  10 1 2 3 1  9 4 5 1 2  15 5 10 15 1  100 20 30 75 |
| 82. | It is the World Cup Finals. Surya only finds a match interesting if the skill difference of the competing teams is less than or equal to D.  Given that the skills of the teams competing in the final are X and Y respectively, determine whether Surya will find the game interesting or not.  **Input Format**  The first line of input will contain a single integer T, denoting the number of testcases. The description of T testcases follows.  Each testcase consists of a single line of input containing three space- separated integers X, Y, and D — the skill levels of the teams and the maximum skill difference.  **Output Format**  For each testcase, output "YES" if Chef will find the game interesting, else output "NO" (without the quotes). The checker is case-insensitive, so "YeS" and "nO" etc. are also acceptable.  **Constraints**  1≤T≤2000  1≤X,Y≤100  0≤D≤100  **Sample Input: Sample Output:**  3 YES  5 3 4 NO  5 3 1 YES |

|  |  |
| --- | --- |
|  | 5 5 0 |
| 83. | After the phenomenal success of the 36th Chamber of Shaolin, San Te has decided to start 37th Chamber of Shaolin. The aim this time is to equip women with shaolin self-defence techniques.  The only condition for a woman to be eligible for the special training is that she must be between 1010 and 6060 years of age, inclusive of both 1010 and 6060.  Given the ages of N women in his village, please help San Te find out how many of them are eligible for the special training.  **Input Format**  The first line of input contains a single integer T, denoting the number of test cases. The description of T test cases follows.  The first line of each test case contains a single integer N, the number of women.  The second line of each test case contains N space-separated integers A1 ,A2 ,...,AN , the ages of the women.  **Output Format**  For each test case, output in a single line the number of women eligible for self-defence training.  **Constraints**  1≤T≤20  1≤N≤100  1≤Ai ≤100  **Sample Input: Sample Output:**  3 2  3 2  15 23 65 1  3  15 62 16  2  35 9 |
| 84. | There are two problems in a contest.  Problem A is worth 500 points at the start of the contest. Problem B is worth 1000 points at the start of the contest.  Once the contest starts, after each minute:  Maximum points of Problem A reduce by 2 points . Maximum points of Problem B reduce by 4 points.  It is known that Ajay requires X minutes to solve Problem A correctly and Y minutes to solve Problem B correctly.  Find the maximum number of points Ajay can score if he optimally decides the order of attempting both the problems. |

|  |  |
| --- | --- |
|  | **Input Format**  First line will contain T, number of test cases. Then the test cases follow. Each test case contains of a single line of input, two integers X and Y - the time required to solve problems A and B in minutes respectively.  **Output Format**  For each test case, output in a single line, the maximum number of points Chef can score if he optimally decides the order of attempting both the problems.  **Constraints**  1≤T≤1000  1≤X,Y≤100  **Sample Input: Sample Output:**  4 1360  10 20 1292  8 40 1380  15 15 1400  20 10 |
| 85. | JK is struggling to pass a certain college course.  The test has a total of N questions, each question carries 3 marks for a correct answer and −1 for an incorrect answer. JK is a risk-averse person so he decided to attempt all the questions. It is known that JK got X questions correct and the rest of them incorrect. For JK to pass the course he must score at least P marks.  Will JK be able to pass the exam or not?  **Input Format**  First line will contain T, number of testcases. Then the testcases follow. Each testcase contains of a single line of input, three integers N,X,P. **Output Format**  For each test case output "PASS" if Chef passes the exam and "FAIL" if JK fails the exam.  You may print each character of the string in uppercase or lowercase (for example, the strings "pASs", "pass", "Pass" and "PASS" will all be treated as identical).  **Constraints**  1≤T≤1000  1≤N≤100  0≤X≤N  0≤P≤3⋅ N  **Sample Input: Sample Ouput:**  3 PASS  5 2 3 FAIL  5 2 4 FAIL |

|  |  |
| --- | --- |
|  | 4 0 0 |
| 86. | You are given the sizes of angles of a simple quadrilateral (in |
|  | degrees) A, B, C and D, in some order along its perimeter. Determine |
|  | whether the quadrilateral is cyclic. |
|  | Note: A quadrilateral is cyclic if and only if the sum of opposite angles |
|  | is 180∘ . |
|  | **Input** |
|  | The first line of the input contains a single integer T denoting the number of |
|  | test cases. The description of T test cases follows. |
|  | The first and only line of each test case contains four space-separated |
|  | integers A, B, C and D. |
|  | **Output** |
|  | Print a single line containing the string "YES" if the given quadrilateral is |
|  | cyclic or "NO" if it is not (without quotes). |
|  | You may print each character of the string in uppercase or lowercase (for |
|  | example, the strings "yEs", "yes", "Yes" and "YES" will all be treated as |
|  | identical). |
|  | **Constraints** |
|  | 1≤T≤10^4 |
|  | 1≤A,B,C,D≤357 |
|  | A+B+C+D=360 |
|  | **Sample Input: Sample Ouput:** |
|  | 3 NO |
|  | 10 20 30 300 YES |
|  | 10 20 170 160 NO |
|  | 179 1 179 1 |
| 87. | RK bought N items from a shop. Although it is hard to carry all these items in |
|  | hand, so Chef has to buy some polybags to store these items. |
|  | 1 polybag can contain at most 10 items. What is the minimum number of |
|  | polybags needed by RK? |
|  | **Input Format** |
|  | The first line will contain an integer T - number of test cases. Then the test |
|  | cases follow. |
|  | The first and only line of each test case contains an integer N - the number |
|  | of items bought by RK. |
|  | **Output Format** |
|  | For each test case, output the minimum number of polybags required. |
|  | **Constraints** |
|  | 1≤T≤1000 |
|  | 1≤N≤1000 |
|  | **Sample Input: Sample Output:** |
|  | 3 2 |
|  | 20 3 |
|  | 24 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 99 | | | | | |
| 88. | Given n (n is even), determine the number an n×n chessboard.  **Input Format**  The only line of the input contains a single integer n.  **Output Format**  Output the number of black cells in an n×n chessboard.  **Constraints**  2≤n≤100 n is even  **Sample Input: Sample Output:**  8 32 | | of | black | cells | in |
| 89. | Ram has fallen in love with Sita, and wants to buy N gifts for her. On | | | | | |
|  | reaching the gift shop, Ram got to know the following two things: | | | | | |
|  | The cost of each gift is 1 coin. | | | | | |
|  | On the purchase of every 4th gift, Ram gets the 5th gift free of cost. | | | | | |
|  | What is the minimum number of coins that Ram will require in order to | | | | | |
|  | come out of the shop carrying N gifts? | | | | | |
|  | **Input Format** | | | | | |
|  | The first line of input will contain an integer T — the number of test cases. | | | | | |
|  | The description of T test cases follows. | | | | | |
|  | The first and only line of each test case contains an integer N, the number of | | | | | |
|  | gifts in the shop. | | | | | |
|  | **Output Format** | | | | | |
|  | For each test case, output on a new line the minimum number of coins that | | | | | |
|  | Chef will require to obtain all N gifts. | | | | | |
|  | **Constraints** | | | | | |
|  | 1≤T≤1000 | | | | | |
|  | 1≤N≤10^9 | | | | | |
|  | **Sample Input:** | **Sample Output:** |  |  |  |  |
|  | 2 | 4 |  |  |  |  |
|  | 5 | 4 |  |  |  |  |
|  | 4 | | | | | |
| 90. | In Chefland, denominations less than rupees 10 have stopped and now rupees 10 is the smallest denomination.  Suppose KK goes to buy some item with cost not a multiple of 10, then, he will be charged the cost that is the nearest multiple of 10.  If the cost is equally distant from two nearest multiples of 1010, then the cost is rounded up.  For example, 35,38,40,44 are all rounded to 40.  KK purchased an item having cost X (X≤100) and gave a bill of rupees 100. How much amount will he get back?  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases. | | | | | |

|  |  |
| --- | --- |
|  | Each test case consists of a single integer X, the cost of the item. |
| **Output Format** |
| For each test case, output the amount returned to Chef. |
| **Constraints** |
| 1≤T≤100 |
| 1≤X≤100 |
| **Sample Input: Sample Output:** |
| 4 60 |
| 35 50 |
| 54 20 |
| 80 90 |
| 12 |
| 91. | Chef has 3 numbers A,B and C. |
|  | Chef wonders if it is possible to choose exactly two numbers out of the three |
|  | numbers such that their sum is odd. |
|  | **Input Format** |
|  | The first line of input will contain a single integer T, denoting the number of |
|  | test cases. |
|  | Each test case consists of three integers A,B,C. |
|  | **Output Format** |
|  | For each test case, output YES if you can choose exactly two numbers with |
|  | odd sum, NO otherwise. |
|  | The output is case-insensitive. Thus, the strings YES, yes, yeS, and Yes are all |
|  | considered the same. |
|  | **Constraints** |
|  | 1≤T≤100 |
|  | 1≤A,B,C≤10 |
|  | **Sample Input: Sample Output:** |
|  | 4 YES |
|  | 1 2 3 NO |
|  | 8 4 6 NO |
|  | 3 3 9 YES |
|  | 7 8 6 |
| 92. | James has a square-shaped chart paper with the side length equal to N. He wants to cut out K×K squares from this chart paper.  Find the maximum number of K×K squares he can cut from the entire chart paper.  Note that, some part of the chart paper might not be a included in any K×K cutout square.  **Input Format**  The first line contains a single integer T — the number of test cases. Then the test cases follow.  The first and only line of each test case contains two space-separated  integers N and K — the side length of the entire chart paper and the side length of the cutout squares. |

|  |  |  |
| --- | --- | --- |
|  | **Output Format** |  |
| For each test case, output on a new line the maximum | number |
| of K×K squares James can cut from the entire chart paper. |  |
| **Constraints** |  |
| 1≤T≤1000 |  |
| 1≤K≤N≤1000 |  |
| **Sample Input: Sample Output:** |  |
| 3 25 |  |
| 5 1 1 |  |
| 2 2 4 |  |
| 5 2 |  |
| 93. | Roy is confused whether to go out and eat at the restaurant or order food | |
|  | online. | |
|  | The online order costs N rupees while the cost of eating at the restaurant | |
|  | is M rupees. | |
|  | However, Roy has a discount coupon with which he can avail flat 10% off on | |
|  | his online order. | |
|  | Find the cheaper option for Roy to eat, i.e., whether to order food online or | |
|  | eat at the restaurant. | |
|  | **Input Format** | |
|  | The first line of input will contain a single integer T, denoting the number of | |
|  | test cases. | |
|  | Each test case consists of two space-separated integers N and M, the cost of | |
|  | ordering online and eating at the restaurant respectively. | |
|  | **Output Format** | |
|  | For each test case, output on a new line: | |
|  | ONLINE, if Roy gets a better deal in online ordering, | |
|  | DINING if Roy gets a better deal in eating at the restaurant, | |
|  | EITHER if both deals cost the same. | |
|  | You may print each character in uppercase or lowercase, For example, the | |
|  | strings Online, online, ONLINE, and onLiNe are all considered identical. | |
|  | **Constraints** | |
|  | 1≤T≤1000 | |
|  | 1≤N,M≤1000 | |
|  | **Sample Input: Sample Output:** | |
|  | 4 ONLINE | |
|  | 500 500 DINING | |
|  | 500 400 DINING | |
|  | 25 22 EITHER | |
|  | 100 90 | |
| 94. | There are 2 stores in Chefland and both sell the same product. The first store sells the product for 100 rupees whereas the second store sells it for 200 rupees.  It is the holiday season and both stores have announced a special discount.  The first store is providing a discount of A percent on its product and the | |

|  |  |
| --- | --- |
|  | second store is providing a discount of B percent on its product.  Tony is wondering which store is selling the product at a cheaper price after the discount has been applied. Can you help him identify the better deal?  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  Each test case consists of a single line of input containing two space- separated integers A and B denoting the discount provided by the first and second store respectively.  **Output Format**  For each test case, output FIRST if the first store is cheaper, SECOND if the second store is cheaper, and BOTH if both the stores are selling the product for the same price after discount.  The checker is case-insensitive so answers like FiRsT, first, and FIRST would be considered the same.  **Constraints**  1≤T≤1000  1≤A,B≤100  **Sample Input: Sample output:**   1. FIRST 2. 20 SECOND   10 60 FIRST  7 7 BOTH  10 55 |
| 95. | Hackerman wants to know who is the better player between Bob and Alice with the help of a game.  The game proceeds as follows:  First, Alice throws a die and gets the number A Then, Bob throws a die and gets the number B  Alice wins the game if the sum on the dice is a [prime number](https://en.wikipedia.org/wiki/Prime_number); and Bob wins otherwise.  Given A and B, determine who wins the game.  **Input Format**  The first line of input will contain a single integer T, denoting the number of test cases.  The first and only line of each test case contains two space-separated integers A and B.  **Output Format**  For each test case, output on a new line the winner of the game: Alice or Bob.  Each letter of the output may be printed in either uppercase or lowercase, i.e, Alice, ALICE, AlIce and aLIcE will all be considered equivalent.  **Constraints**  1≤T≤36 |

|  |  |
| --- | --- |
|  | 1≤A≤6  1≤B≤6  **Sample Input: Sample Output:**  3 Alice  2 1 Alice  1 1 Bob  2 2 |
| 96. | Arun has two variables X and Y. He wants to find out whether the variables |
|  | satisfy the equation: |
|  | X^2+4.Y^2=4.X^2.Y |
|  | **Input Format** |
|  | The first line of input will contain a single integer T, denoting the number of |
|  | test cases. |
|  | Each test case consists of two integers X and Y, as mentioned in statement. |
|  | **Output Format** |
|  | For each test case, output YES if the variables X and Y satisfy the given |
|  | equation, NO otherwise. |
|  | You may print each character in uppercase or lowercase. For |
|  | example, Yes, YES, yes, and YeS are all considered the same. |
|  | **Constraints** |
|  | 1≤T≤1000 |
|  | 1≤X≤10^9 |
|  | 1≤Y≤10^18 |
|  | **Sample Input: Sample Output:** |
|  | 5 YES |
|  | 2 2 NO |
|  | 4 4 NO |
|  | 3 6 YES |
|  | 8 32 YES |
|  | 200000000 20000000000000000 |
| 97. | Rushitote went to a programming contest to distribute apples and oranges to the contestants.  He has N apples and M oranges, which need to be divided equally amongst the contestants. Find the maximum possible number of contestants such that:  Every contestant gets an equal number of apples; and Every contestant gets an equal number of oranges.  Note that every fruit with Rushitote must be distributed, there cannot be any left over.  For example, 2 apples and 4 oranges can be distributed equally to two contestants, where each one receives 1 apple and 2 oranges.  However, 2 apples and 5 oranges can only be distributed equally to one contestant.  **Input Format** |

|  |  |
| --- | --- |
|  | The first line of input will contain a single integer T, denoting the number of |
| test cases. |
| The first and only line of each test case contains two space-separated |
| integers N and M — the number of apples and oranges, respectively. |
| **Output Format** |
| For each test case, output on a new line the answer: the maximum number |
| of contestants such that everyone receives an equal number of apples and |
| an equal number of oranges. |
| **Constraints** |
| 1≤T≤1000 |
| 1≤N,M≤10^9 |
| **Sample Input: Sample Output:** |
| 3 1 |
| 1 5 2 |
| 2 4 2 |
| 4 6 |
| 98. | Luigi has an array A of N positive integers. He wants to make all elements of |
|  | the array equal. |
|  | In one move, he can: |
|  | Choose an index i (1≤i≤N) and divide the element Ai by any one of |
|  | its [divisors](https://en.wikipedia.org/wiki/Divisor). |
|  | In other words, he can choose a positive integer X such that X∣Ai and |
|  | set Ai :=XAi . |
|  | Find the minimum number of moves required to make all the elements of |
|  | the array equal. |
|  | **Input Format** |
|  | The first line of input will contain a single integer T, denoting the number of |
|  | test cases. |
|  | Each test case consists of two lines of input. |
|  | The first line of each test case contains N, the size of array A. |
|  | The second line of each test case contains N space-separated integers, the |
|  | elements of array A. |
|  | **Output Format** |
|  | For each test case, output on a new line, the minimum number of moves |
|  | required to make all elements of the array equal. |
|  | **Constraints** |
|  | 1≤T≤1000 |
|  | 1≤N≤3000 |
|  | 1≤Ai ≤10^9 |
|  | **Sample Input: Sample Output:** |
|  | 4 1 |
|  | 2 0 |
|  | 11 22 2 |
|  | 5 4 |
|  | 38 38 38 38 38 |

|  |  |
| --- | --- |
|  | 4 |
| 4 4 16 8 |
| 4 |
| 11 13 17 19 |
| 99. | Raghu has an array A of length N. |
|  | An index i is called strong if we can change the [gcd](https://en.wikipedia.org/wiki/Greatest_common_divisor) of the whole array just by |
|  | changing the value of Ai . |
|  | Determine the number of strong indices in the array. |
|  | **Input Format** |
|  | First line will contain T, number of test cases. Then the test cases follow. |
|  | First line of each test case contains an integer N denoting the size of the |
|  | array A. |
|  | Second line contains N space separated integers A1 ,A2 ,…,AN - |
|  | denoting the array A. |
|  | **Output Format** |
|  | For each test case, output the number of strong indices in the array. |
|  | **Constraints** |
|  | 1≤T≤5⋅ 10^4 |
|  | 2≤N≤3⋅ 10^5 |
|  | 1≤Ai ≤10^9 |
|  | Sum of N over all test cases do not exceed 3⋅ 105. |
|  | **Sample Input: Sample Output:** |
|  | 3 3 |
|  | 3 0 |
|  | 5 10 20 4 |
|  | 4 |
|  | 3 5 7 11 |
|  | 4 |
|  | 2 2 2 2 |
| 100. | Given an integer N, help Chef in finding an N- digit odd positive integerodd positive integer X such that X is divisible by 3 but not by 9.  Note:Note: There should not be any leading zeroes in X. In other words, 003 is not a valid 3-digit odd positive integer.  **Input Format**  The first line of input contains a single integer T, denoting the number of testcases. The description of the T testcases follows.  The first and only line of each test case contains a single integer N, denoting the number of digits in X.  **Output Format**  For each testcase, output a single line containing an N-digit odd positive integer X in decimal number system, such that X is divisible by 3 but not by 9.  **Constraints**  1≤T≤500  1≤N≤10^4 |

|  |  |
| --- | --- |
|  | The sum of N over all test cases does not exceed 105  **Sample Input: Sample Output:**  3 3  1 15  2 123  3 |

|  |  |
| --- | --- |
| **S.NO** | **Description** |
| 101. | The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)  P A H N A P L S I I G Y I R  And then read line by line: "PAHNAPLSIIGYIR"  Write the code that will take a string and make this conversion given a number of rows:  string convert(string s, int numRows);  **Example 1:**  Input: s = "PAYPALISHIRING", numRows = 3 Output: "PAHNAPLSIIGYIR"  **Example 2:**  Input: s = "PAYPALISHIRING", numRows = 4 Output: "PINALSIGYAHRPI"  **Explanation:**  P I N A L S I G Y A H R  P I  **Constraints:**  1 <= s.length <= 1000  s consists of English letters (lower-case and upper-case), ',' and '.'. 1 <= numRows <= 1000 |
| 102. | You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water.  Return the maximum amount of water a container can store. Notice that you may not slant the container.  **Example 1:**    **Input:** height = [1,8,6,2,5,4,8,3,7] |

|  |  |
| --- | --- |
|  | **Output:** 49  **Explanation:** The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.  **Constraints:**  n == height.length 2 <= n <= 10^5  0 <= height[i] <= 10^4 |
| 103. | Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.  Notice that the solution set must not contain duplicate triplets.  **Example 1:**  **Input**: nums = [-1,0,1,2,-1,-4]  **Output:** [[-1,-1,2],[-1,0,1]]  **Explanation:**  nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.  nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.  nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0. The distinct triplets are [-1,0,1] and [-1,-1,2].  Notice that the order of the output and the order of the triplets does not matter.  **Constraints:**  3 <= nums.length <= 3000  -10^5 <= nums[i] <= 10^5 |
| 104. | Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.  Return the sum of the three integers.  You may assume that each input would have exactly one solution.  **Example 1:**  **Input:** nums = [-1,2,1,-4], target = 1  **Output:** 2  **Explanation:** The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).  **Constraints:**  3 <= nums.length <= 500  -1000 <= nums[i] <= 1000  -10^4 <= target <= 10^4 |
| 105. | Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.  A mapping of digits to letters (just like on the telephone buttons) is given  below. Note that 1 does not map to any letters. |

|  |  |
| --- | --- |
|  | **Example 1:**  **Input:** digits = "23"  **Output:** ["ad","ae","af","bd","be","bf","cd","ce","cf"]  **Constraints:**  0 <= digits.length <= 4  digits[i] is a digit in the range ['2', '9']. |
| 106. | Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that:  0 <= a, b, c, d < n  a, b, c, and d are distinct.  nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order.  **Example 1:**  **Input:** nums = [1,0,-1,0,-2,2], target = 0  **Output:** [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]  **Constraints:**  1 <= nums.length <= 200  -10^9 <= nums[i] <= 10^9  -10^9 <= target <= 10^9 |
| 107. | Given the head of a linked list, remove the nth node from the end of the list and return its head.  **Example 1:**    **Input:** head = [1,2,3,4,5], n = 2  **Output:** [1,2,3,5] |

|  |  |
| --- | --- |
|  | **Constraints:**  The number of nodes in the list is sz. 1 <= sz <= 30  0 <= Node.val <= 100 1 <= n <= sz |
| 108. | Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.  **Example 1:**  Input: n = 3  Output: ["((()))","(()())","(())()","()(())","()()()"]  **Example 2:** Input: n = 1 Output: ["()"]  **Constraints:**  1 <= n <= 8 |
| 109. | There is an integer array nums sorted in ascending order (with distinct values).  Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0- indexed). For example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].  Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.  You must write an algorithm with O(log n) runtime complexity.  **Example 1:**  **Input:** nums = [4,5,6,7,0,1,2], target = 0  **Output:** 4  **Example 2:**  Input: nums = [4,5,6,7,0,1,2], target = 3 Output: -1  **Constraints:**  1 <= nums.length <= 5000  -10^4 <= nums[i] <= 10^4  All values of nums are unique.  nums is an ascending array that is possibly rotated.  -10^4 <= target <= 10^4 |

|  |  |
| --- | --- |
| 110. | Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.  If target is not found in the array, return [-1, -1].  You must write an algorithm with O(log n) runtime complexity.  **Example 1:**  Input: nums = [5,7,7,8,8,10], target = 8 Output: [3,4]  **Example 2:**  Input: nums = [5,7,7,8,8,10], target = 6 Output: [-1,-1]  **Constraints:**  0 <= nums.length <= 10^5  -10^9 <= nums[i] <= 10^9  nums is a non-decreasing array.  -10^9 <= target <= 10^9 |
| 111. | The count-and-say sequence is a sequence of digit strings defined by the recursive formula:  countAndSay(1) = "1"  countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string. To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.  For example, the saying and conversion for digit string "3322251":    Given a positive integer n, return the nth term of the count-and- say sequence.  **Example 1:**  Input: n = 1 Output: "1"  Explanation: This is the base case.  **Example 2:** Input: n = 4 Output: "1211"  **Constraints:** |

|  |  |
| --- | --- |
|  | 1 <= n <= 30 |
| 112. | Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.  The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the  frequency  of at least one of the chosen numbers is different.  The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.  **Example 1:**  Input: candidates = [2,3,6,7], target = 7 Output: [[2,2,3],[7]]  **Explanation:**  2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times.  7 is a candidate, and 7 = 7.  These are the only two combinations.  **Constraints:**   1. <= candidates.length <= 30 2. <= candidates[i] <= 40   All elements of candidates are distinct. 1 <= target <= 40 |
| 113. | Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target.  Each number in candidates may only be used once in the combination. Note: The solution set must not contain duplicate combinations.  **Example 1:**  Input: candidates = [10,1,2,7,6,1,5], target = 8  **Output:**  [ [1,1,6],  [1,2,5],  [1,7],  [2,6]  ]  **Constraints:**  1 <= candidates.length <= 100  1 <= candidates[i] <= 50  1 <= target <= 30 |

|  |  |
| --- | --- |
|  |  |
| 114. | Given an m x n matrix, return all elements of the matrix in spiral order.  **Example 1:**    Input: matrix = [[1,2,3],[4,5,6],[7,8,9]] Output: [1,2,3,6,9,8,7,4,5]  **Constraints:**  m == matrix.length n == matrix[i].length 1 <= m, n <= 10  -100 <= matrix[i][j] <= 100 |
| 115. | You are given a 0-indexed array of integers nums of length n. You are initially positioned at nums[0].  Each element nums[i] represents the maximum length of a forward jump from index i. In other words, if you are at nums[i], you can jump to any nums[i + j] where:   1. <= j <= nums[i] and i + j < n   Return the minimum number of jumps to reach nums[n - 1]. The test cases are generated such that you can reach nums[n - 1].  **Example 1:**  Input: nums = [2,3,1,1,4] Output: 2  **Explanation:** The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.  **Constraints:**   1. <= nums.length <= 10^4 0 <= nums[i] <= 1000   It's guaranteed that you can reach nums[n - 1]. |
| 116. | Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: intervals = [[1,3],[2,6],[8,10],[15,18]]  Output: [[1,6],[8,10],[15,18]]  **Explanation:** Since intervals [1,3] and [2,6] overlap, merge them into [1,6].  **Example 2:**  Input: intervals = [[1,4],[4,5]] Output: [[1,5]]  Explanation: Intervals [1,4] and [4,5] are considered overlapping.  **Constraints:**  1 <= intervals.length <= 10^4 intervals[i].length == 2  0 <= starti <= endi <= 10^4 |
| 117. | 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]  **Constraints:**  The number of nodes in the list is in the range [0, 500].  -100 <= Node.val <= 100 0 <= k <= 2 \* 10^9 |
| 118. | 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 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.  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 \* 109.  **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:**  1 <= m, n <= 100 |
| 119. | 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 to 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   **Constraints:**  m == obstacleGrid.length n == obstacleGrid[i].length 1 <= m, n <= 100  obstacleGrid[i][j] is 0 or 1. |
| 120. | Given an array nums with n objects colored red, white, or blue, sort them [in-](https://en.wikipedia.org/wiki/In-place_algorithm) |

|  |  |
| --- | --- |
|  | [place](https://en.wikipedia.org/wiki/In-place_algorithm) so that objects of the same color are adjacent, with the colors in the order red, white, and blue.  We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.  You must solve this problem without using the library's sort function.  **Example 1:**  Input: nums = [2,0,2,1,1,0]Output: [0,0,1,1,2,2]  Example 2:  Input: nums = [2,0,1]Output: [0,1,2]  **Constraints:**  n == nums.length 1 <= n <= 300  nums[i] is either 0, 1, or 2. |
| 121. | You are given an m x n integer matrix matrix with the following two properties:  Each row is sorted in non-decreasing order.  The first integer of each row is greater than the last integer of the previous row.  Given an integer target, return true if target is in matrix or false otherwise. You must write a solution in O(log(m \* n)) time complexity.  **Example 1:**    Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3  Output: true  **Constraints:**  m == matrix.length n == matrix[i].length 1 <= m, n <= 100  -10^4 <= matrix[i][j], target <= 10^4 |
| 122. | Given the head of a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.  You should preserve the original relative order of the nodes in each of the two partitions.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: head = [1,4,3,2,5,2], x = 3  Output: [1,2,2,4,3,5]  **Constraints:**  The number of nodes in the list is in the range [0, 200].  -100 <= Node.val <= 100  -200 <= x <= 200 |
| 123. | Given an integer array nums that may contain duplicates, return all possible subsets  (the power set).  The solution set must not contain duplicate subsets. Return the solution in any order.  **Example 1:**  Input: nums = [1,2,2]  Output: [[],[1],[1,2],[1,2,2],[2],[2,2]]  **Example 2:**  Input: nums = [0] Output: [[],[0]]  **Constraints:**  1 <= nums.length <= 10  -10 <= nums[i] <= 10 |
| 124. | Given the head of a singly linked list and two integers left and right where left <= right, reverse the nodes of the list from position left to position right, and return the reversed list.  **Example 1:**    Input: head = [1,2,3,4,5], left = 2, right = 4 Output: [1,4,3,2,5]  **Example 2:**  Input: head = [5], left = 1, right = 1 Output: [5]  **Constraints:**  The number of nodes in the list is n. |

|  |  |
| --- | --- |
|  | 1 <= n <= 500  -500 <= Node.val <= 500 1 <= left <= right <= n |
| 125. | Given a triangle array, return the minimum path sum from top to bottom. For each step, you may move to an adjacent number of the row below. More formally, if you are on index i on the current row, you may move to either index i or index i + 1 on the next row.  **Example 1:**  Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]  Output: 11  Explanation: The triangle looks like:  2  3 4  6 5 7  4 1 8 3  The minimum path sum from top to bottom is 2 + 3 + 5 + 1 = 11  **Constraints:**  1 <= triangle.length <= 200  triangle[0].length == 1  triangle[i].length == triangle[i - 1].length + 1  -10^4 <= triangle[i][j] <= 10^4 |
| 126. | Given an integer array nums where every element appears three times except for one, which appears exactly once. Find the single element and return it.  You must implement a solution with a linear runtime complexity and use only constant extra space.  **Example 1:**  Input: nums = [2,2,3,2] Output: 3  **Example 2:**  Input: nums = [0,1,0,1,0,1,99]  Output: 99  **Constraints:**  1 <= nums.length <= 3 \* 104  -231 <= nums[i] <= 231 - 1  Each element in nums appears exactly three times except for one element which appears once. |
| 127. | Given the head of a linked list, return the list after sorting it in ascending order. |

|  |  |
| --- | --- |
|  | **Example 1:**    Input: head = [4,2,1,3] Output: [1,2,3,4]  **Constraints:**  The number of nodes in the list is in the range [0, 5 \* 10^4].  -10^5 <= Node.val <= 10^5 |
| 128. | Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.  Note that the same word in the dictionary may be reused multiple times in the segmentation.  **Example 1:**  Input: s = "applepenapple", wordDict = ["apple","pen"] Output: true  Explanation: Return true because "applepenapple" can be segmented as "apple pen apple".  **Example 2:**  Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"] Output: false  **Constraints:**  1 <= s.length <= 300  1 <= wordDict.length <= 1000  1 <= wordDict[i].length <= 20  s and wordDict[i] consist of only lowercase English letters. All the strings of wordDict are unique. |
| 129. | Given a string s, partition s such that every substring  of the partition is a palindrome  . Return all possible palindrome partitioning of s.  **Example 1:**  Input: s = "aab" |

|  |  |
| --- | --- |
|  | Output: [["a","a","b"],["aa","b"]]  **Example 2:**  Input: s = "a" Output: [["a"]]  **Constraints:**  1 <= s.length <= 16  s contains only lowercase English letters. |
| 130. | Given an input string s, reverse the order of the words.  A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space.  Return a string of the words in reverse order concatenated by a single space. Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.  **Example 1:**  Input: s = "the sky is blue" Output: "blue is sky the" **Example 2:**  Input: s = " hello world " Output: "world hello"  **Constraints:**  1 <= s.length <= 104  s contains English letters (upper-case and lower-case), digits, and spaces ' '. There is at least one word in s. |
| 131. | Given a 1-indexed array of integers numbers that is already sorted in non- decreasing order, find two numbers such that they add up to a specific target number. Let these two numbers be numbers[index1] and numbers[index2] where 1 <= index1 < index2  < numbers.length.  Return the indices of the two numbers, index1 and index2, added by one as an integer array [index1, index2] of length 2.  The tests are generated such that there is exactly one solution. You may not use the same element twice.  Your solution must use only constant extra space.  **Example 1:**  Input: numbers = [2,7,11,15], target = 9 Output: [1,2]  Explanation: The sum of 2 and 7 is 9. Therefore, index1 = 1, index2 = 2. We return [1, 2]. |

|  |  |
| --- | --- |
|  | **Constraints:**  2 <= numbers.length <= 3 \* 10^4  -1000 <= numbers[i] <= 1000  numbers is sorted in non-decreasing order.  -1000 <= target <= 1000  The tests are generated such that there is exactly one solution. |
| 132. | Given an integer array nums, rotate the array to the right by k steps, where k is non-negative.  **Example 1:**  Input: nums = [1,2,3,4,5,6,7], k = 3  Output: [5,6,7,1,2,3,4]  **Explanation:**  rotate 1 steps to the right: [7,1,2,3,4,5,6] rotate 2 steps to the right: [6,7,1,2,3,4,5] rotate 3 steps to the right: [5,6,7,1,2,3,4]  **Constraints:**  1 <= nums.length <= 10^5  -231 <= nums[i] <= 231 - 1 0 <= k <= 10^5 |
| 133. | Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return 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: grid = [ ["1","1","1","1","0"],  ["1","1","0","1","0"],  ["1","1","0","0","0"],  ["0","0","0","0","0"]  ]  Output: 1  **Constraints:**  m == grid.length n == grid[i].length 1 <= m, n <= 300  grid[i][j] is '0' or '1'. |
| 134. | You are a professional robber planning to rob houses along a street. Each  house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security |

|  |  |
| --- | --- |
|  | systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.  Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.  **Example 1:**  Input: nums = [1,2,3,1]Output: 4Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).  Total amount you can rob = 1 + 3 = 4.  **Example 2:**  Input: nums = [2,7,9,3,1]Output: 12Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).  Total amount you can rob = 2 + 9 + 1 = 12.  **Constraints:**  1 <= nums.length <= 100  0 <= nums[i] <= 400 |
| 135. | Given an integer n, return the number of prime numbers that are strictly less than n.  **Example 1:**  Input: n = 10 Output: 4  Explanation: There are 4 prime numbers less than 10, they are 2, 3, 5, 7.  **Constraints:**  0 <= n <= 5 \* 10^6 |
| 136. | There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.  For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.  Return true if you can finish all courses. Otherwise, return false.  **Example 1:**  Input: numCourses = 2, prerequisites = [[1,0]] Output: true  Explanation: There are a total of 2 courses to take.  To take course 1 you should have finished course 0. So it is possible.  **Example 2:**  Input: numCourses = 2, prerequisites = [[1,0],[0,1] Output: false |

|  |  |
| --- | --- |
|  | Explanation: There are a total of 2 courses to take.  To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible.  **Constraints:**  1 <= numCourses <= 2000  0 <= prerequisites.length <= 5000  prerequisites[i].length == 2 0 <= ai, bi < numCourses  All the pairs prerequisites[i] are unique. |
| 137. | There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.  For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.  Return the ordering of courses you should take to finish all courses. If there are many valid answers, return any of them. If it is impossible to finish all courses, return an empty array.  **Example 1:**  Input: numCourses = 2, prerequisites = [[1,0]] Output: [0,1]  Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is [0,1].  **Example 2:**  Input: numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]] Output: [0,2,1,3]  Explanation: There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.  So one correct course order is [0,1,2,3]. Another correct ordering is [0,2,1,3].  **Constraints:**  1 <= numCourses <= 2000  0 <= prerequisites.length <= numCourses \* (numCourses - 1) prerequisites[i].length == 2  0 <= ai, bi < numCourses ai != bi  All the pairs [ai, bi] are distinct. |
| 138. | You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have a security system connected, and it  will automatically contact the police if two adjacent houses were broken into |

|  |  |
| --- | --- |
|  | on the same night.  Given an integer array nums representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.  **Example 1:**  Input: nums = [2,3,2] Output: 3  Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent houses.  **Example 2:**  Input: nums = [1,2,3,1] Output: 4  Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3). Total amount you can rob = 1 + 3 = 4.  **Constraints:**  1 <= nums.length <= 100  0 <= nums[i] <= 1000 |
| 139. | Find all valid combinations of k numbers that sum up to n such that the following conditions are true:  Only numbers 1 through 9 are used. Each number is used at most once.  Return a list of all possible valid combinations. The list must not contain the same combination twice, and the combinations may be returned in any order.  **Example 1:**  Input: k = 3, n = 7 Output: [[1,2,4]] Explanation:  1 + 2 + 4 = 7  There are no other valid combinations.  **Example 2:**  Input: k = 3, n = 9  Output: [[1,2,6],[1,3,5],[2,3,4]]  Explanation:  1 + 2 + 6 = 9  1 + 3 + 5 = 9  2 + 3 + 4 = 9  There are no other valid combinations.  **Constraints:**  2 <= k <= 9  1 <= n <= 60 |

|  |  |
| --- | --- |
| 140. | Given an m x n binary matrix filled with 0's and 1's, find the largest square |
|  | containing only 1's and return its area. |
|  | **Example 1:** |
|  |  |
|  | Input: matrix = |
|  | [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0"," |
|  | 1","0"]] |
|  | Output: 4 |
|  | **Example 2:** |
|  |  |
|  | Input: matrix = [["0","1"],["1","0"]] |
|  | Output: 1 |
|  | **Constraints:** |
|  | m == matrix.length |
|  | n == matrix[i].length |
|  | 1 <= m, n <= 300 |
|  | matrix[i][j] is '0' or '1'. |
| 141. | An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.  Given an integer n, return the nth ugly number.  **Example 1:**  Input: n = 10 Output: 12  Explanation: [1, 2, 3, 4, 5, 6, 8, 9, 10, 12] is the sequence of the first 10 ugly numbers.  **Constraints:**  1 <= n <= 1690 |
| 142. | Given an integer array nums, return an array answer such that answer[i] is equal to the product of all the elements of nums except nums[i].  The product of any prefix or suffix of nums is guaranteed to fit in a 32- bit integer. |

|  |  |
| --- | --- |
|  | You must write an algorithm that runs in O(n) time and without using the division operation.  **Example 1:**  Input: nums = [1,2,3,4] Output: [24,12,8,6]  **Example 2:**  Input: nums = [-1,1,0,-3,3]  Output: [0,0,9,0,0]  **Constraints:**  2 <= nums.length <= 10^5  -30 <= nums[i] <= 30  The product of any prefix or suffix of nums is guaranteed to fit in a 32- bit integer. |
| 143. | Given an integer array of size n, find all elements that appear more than n/3 times.  **Example 1:**  Input: nums = [3,2,3] Output: [3]  **Example 2:**  Input: nums = [1] Output: [1]  **Example 3:**  Input: nums = [1,2] Output: [1,2]  **Constraints:**  1 <= nums.length <= 5 \* 10^4  -10^9 <= nums[i] <= 10^9 |
| 144. | Given a string s which represents an expression, evaluate this expression and return its value.  The integer division should truncate toward zero.  You may assume that the given expression is always valid. All intermediate results will be in the range of [-231, 231 - 1].  Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().  **Example 1:**  Input: s = "3+2\*2" Output: 7  **Example 2:**  Input: s = " 3/2 " Output: 1 |

|  |  |
| --- | --- |
|  | **Example 3:**  Input: s = " 3+5 / 2 " Output: 5 **Constraints:**  1 <= s.length <= 3 \* 10^5  s consists of integers and operators ('+', '-', '\*', '/') separated by some number of spaces.  s represents a valid expression.  All the integers in the expression are non-negative integers in the range [0, 231 - 1].  The answer is guaranteed to fit in a 32-bit integer. |
| 145. | Given an integer array nums, find a subarray  that has the largest product, and return the product.  The test cases are generated so that the answer will fit in a 32-bit integer.  **Example 1:**  Input: nums = [2,3,-2,4] Output: 6  Explanation: [2,3] has the largest product 6.  **Example 2:**  Input: nums = [-2,0,-1] Output: 0  Explanation: The result cannot be 2, because [-2,-1] is not a subarray.  **Constraints:**  1 <= nums.length <= 2 \* 10^4  -10 <= nums[i] <= 10  The product of any prefix or suffix of nums is guaranteed to fit in a 32- bit integer. |
| 146. | Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.  You must write an algorithm that runs in O(n) time.  **Example 1:**  Input: nums = [100,4,200,1,3,2]  Output: 4  Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.  **Example 2:**  Input: nums = [0,3,7,2,5,8,4,6,0,1]  Output: 9  **Constraints:**  0 <= nums.length <= 10^5 |

|  |  |
| --- | --- |
|  | -10^9 <= nums[i] <= 10^9 |
| 147. | Given an integer n, return the least number of perfect square numbers that sum to n.  A perfect square is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, 1, 4, 9, and 16 are perfect squares while 3 and 11 are not.  **Example 1:**  Input: n = 12 Output: 3  Explanation: 12 = 4 + 4 + 4.  **Example 2:**  Input: n = 13 Output: 2  Explanation: 13 = 4 + 9.  **Constraints:**  1 <= n <= 104 |
| 148. | An additive number is a string whose digits can form an additive sequence.  A valid additive sequence should contain at least three numbers. Except for the first two numbers, each subsequent number in the sequence must be the sum of the preceding two.  Given a string containing only digits, return true if it is an additive number or false otherwise.  Note: Numbers in the additive sequence cannot have leading zeros, so sequence 1, 2, 03 or 1, 02, 3 is invalid.  **Example 1:**  Input: "112358"  Output: true  **Explanation:**  The digits can form an additive sequence: 1, 1, 2, 3, 5, 8.  1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, 3 + 5 = 8  **Example 2:**  Input: "199100199"  Output: true  **Explanation:**  The additive sequence is: 1, 99, 100, 199.  1 + 99 = 100, 99 + 100 = 199  **Constraints:**  1 <= num.length <= 35 num consists only of digits. |
| 149. | A super ugly number is a positive integer whose prime factors are in the |

|  |  |
| --- | --- |
|  | array primes.  Given an integer n and an array of integers primes, return the nth super ugly number.  The nth super ugly number is guaranteed to fit in a 32-bit signed integer.  **Example 1:**  Input: n = 12, primes = [2,7,13,19] Output: 32  Explanation: [1,2,4,7,8,13,14,16,19,26,28,32] is the sequence of the first 12 super ugly numbers given primes = [2,7,13,19].  **Constraints:**  1 <= n <= 10^5  1 <= primes.length <= 100  2 <= primes[i] <= 1000  primes[i] is guaranteed to be a prime number.  All the values of primes are unique and sorted in ascending order. |
| 150. | You are given an integer array coins representing coins of different denominations and an integer amount representing a total amount of money.  Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.  You may assume that you have an infinite number of each kind of coin.  **Example 1:**  Input: coins = [1,2,5], amount = 11 Output: 3  Explanation: 11 = 5 + 5 + 1  **Example 2:**  Input: coins = [2], amount = 3 Output: -1  **Constraints:**  1 <= coins.length <= 12  1 <= coins[i] <= 231 - 1 0 <= amount <= 10^4 |
| 151. | Given an integer array nums, find the Subarray with the largest sum, and return its sum.  **Example 1:**  Input: nums = [-2,1,-3,4,-1,2,1,-5,4]  Output: 6  Explanation: The subarray [4,-1,2,1] has the largest sum 6.  **Example 2:** |

|  |  |
| --- | --- |
|  | Input: nums = [5,4,-1,7,8] Output: 23  Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.  **Constraints:**  1 <= nums.length <= 10^5  -10^4 <= nums[i] <= 10^4 |
| 152. | Given strings s1, s2, and s3, find whether s3 is formed by an interleaving of s1 and s2.  An interleaving of two strings s and t is a configuration where s and t are divided into n and m Substrings respectively, such that:  s = s1 + s2 + ... + sn t = t1 + t2 + ... + tm  |n - m| <= 1  The interleaving is s1 + t1 + s2 + t2 + s3 + t3 + ... or t1 + s1 + t2 + s2 + t3 + s3  + ...  Note: a + b is the concatenation of strings a and b.  **Example 1:**    Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac" Output: true  Explanation: One way to obtain s3 is:  Split s1 into s1 = "aa" + "bc" + "c", and s2 into s2 = "dbbc" + "a".  Interleaving the two splits, we get "aa" + "dbbc" + "bc" + "a" + "c" = "aadbbcbcac".  Since s3 can be obtained by interleaving s1 and s2, we return true. |
| 153. | You are given a perfect binary tree where all leaves are on the same level, and every parent has two children. The binary tree has the following definition:  struct Node { int val; Node \*left; Node \*right; Node \*next;  }  Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.  Initially, all next pointers are set to NULL.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: root = [1,2,3,4,5,6,7]  Output: [1,#,2,3,#,4,5,6,7,#]  Explanation: Given the above perfect binary tree (Figure A), your function should populate each next pointer to point to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.  **Example 2:**  Input: root = [] Output: [] **Constraints:**  The number of nodes in the tree is in the range [0, 2^12 - 1].  -1000 <= Node.val <= 1000 |
| 154. | You are given an integer array prices where prices[i] is the price of a given stock on the ith day.  On each day, you may decide to buy and/or sell the stock. You can only hold at most one share of the stock at any time. However, you can buy it then immediately sell it on the same day.  Find and return the maximum profit you can achieve.  **Example 1:**  Input: prices = [7,1,5,3,6,4] Output: 7  Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1  = 4.  Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3. Total profit is 4 + 3 = 7.  **Example 2:**  Input: prices = [1,2,3,4,5] Output: 4  Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1  = 4.  Total profit is 4. |
| 155. | There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].  You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from the ith station to its next (i + 1)th station. You begin the journey with an empty tank at one of the gas stations.  Given two integer arrays gas and cost, return the starting gas station's index |

|  |  |
| --- | --- |
|  | if you can travel around the circuit once in the clockwise direction, otherwise return -1. If there exists a solution, it is guaranteed to be unique  **Example 1:**  Input: gas = [1,2,3,4,5], cost = [3,4,5,1,2]  Output: 3 Explanation:  Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4 Travel to station 4. Your tank = 4 - 1 + 5 = 8  Travel to station 0. Your tank = 8 - 2 + 1 = 7  Travel to station 1. Your tank = 7 - 3 + 2 = 6  Travel to station 2. Your tank = 6 - 4 + 3 = 5  Travel to station 3. The cost is 5. Your gas is just enough to travel back to station 3.  Therefore, return 3 as the starting index.  **Constraints:**  n == gas.length == cost.length 1 <= n <= 105  0 <= gas[i], cost[i] <= 10^4 |
| 156. | You are given an array of strings tokens that represents an arithmetic expression in a [Reverse Polish Notation](http://en.wikipedia.org/wiki/Reverse_Polish_notation).  Evaluate the expression. Return an integer that represents the value of the expression.  Note that:  The valid operators are '+', '-', '\*', and '/'.  Each operand may be an integer or another expression.  The division between two integers always truncates toward zero. There will not be any division by zero.  The input represents a valid arithmetic expression in a reverse polish notation.  The answer and all the intermediate calculations can be represented in a 32- bit integer.  **Example 1:**  Input: tokens = ["2","1","+","3","\*"] Output: 9  Explanation: ((2 + 1) \* 3) = 9  **Example 2:**  Input: tokens = ["4","13","5","/","+"] Output: 6  Explanation: (4 + (13 / 5)) = 6 |
| 157. | Given an integer n, return the number of trailing zeroes in n!. Note that n! = n \* (n - 1) \* (n - 2) \* ... \* 3 \* 2 \* 1. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: n = 3 Output: 0  Explanation: 3! = 6, no trailing zero.  **Example 2:**  Input: n = 5 Output: 1  Explanation: 5! = 120, one trailing zero.  **Constraints:**  0 <= n <= 10^4 |
| 158. | You are given an array prices where prices[i] is the price of a given stock on the ith day.  Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times) with the following restrictions:  After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).  Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).  **Example 1:**  Input: prices = [1,2,3,0,2] Output: 3  Explanation: transactions = [buy, sell, cooldown, buy, sell]  **Example 2:**  Input: prices = [1] Output: 0  **Constraints:**  1 <= prices.length <= 5000  0 <= prices[i] <= 1000 |
| 159. | Given an integer array nums, reorder it such that nums[0] < nums[1] > nums[2] < nums[3]....  You may assume the input array always has a valid answer.  **Example 1:**  Input: nums = [1,5,1,1,6,4]  Output: [1,6,1,5,1,4]  Explanation: [1,4,1,5,1,6] is also accepted.  **Example 2:**  Input: nums = [1,3,2,2,3,1]  Output: [2,3,1,3,1,2] |

|  |  |
| --- | --- |
|  | **Constraints:**  1 <= nums.length <= 5 \* 10^4 0 <= nums[i] <= 5000  It is guaranteed that there will be an answer for the given input nums. |
| 160. | Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.  The first node is considered odd, and the second node is even, and so on. Note that the relative order inside both the even and odd groups should remain as it was in the input.  You must solve the problem in O(1) extra space complexity and O(n) time complexity.  **Example 1:**    Input: head = [1,2,3,4,5]  Output: [1,3,5,2,4]  **Constraints:**  The number of nodes in the linked list is in the range [0, 104].  -106 <= Node.val <= 10^6 |
| 161. | Given an integer array nums, return true if there exists a triple of indices (i, j, k) such that i < j < k and nums[i] < nums[j] < nums[k]. If no such indices exists, return false.  **Example 1:**  Input: nums = [1,2,3,4,5] Output: true  Explanation: Any triplet where i < j < k is valid.  **Example 2:**  Input: nums = [5,4,3,2,1] Output: false  Explanation: No triplet exists.  **Constraints:**  1 <= nums.length <= 5 \* 10^5  -231 <= nums[i] <= 231 - 1 |
| 162. | The thief has found himself a new place for his thievery again. There is only one entrance to this area, called root.  Besides the root, each house has one and only one parent house. After a |

|  |  |
| --- | --- |
|  | tour, the smart thief realized that all houses in this place form a binary tree. It will automatically contact the police if two directly-linked houses were broken into on the same night.  Given the root of the binary tree, return the maximum amount of money the thief can rob without alerting the police.  **Example 1:**    Input: root = [3,2,3,null,3,null,1] Output: 7  Explanation: Maximum amount of money the thief can rob = 3 + 3 + 1 = 7.  **Constraints:**  The number of nodes in the tree is in the range [1, 10^4]. 0 <= Node.val <= 10^4 |
| 163. | Given an integer n, break it into the sum of k positive integers, where k >= 2, and maximize the product of those integers.  Return the maximum product you can get.  **Example 1:**  Input: n = 2 Output: 1  Explanation: 2 = 1 + 1, 1 × 1 = 1.  **Example 2:**  Input: n = 10 Output: 36  Explanation: 10 = 3 + 3 + 4, 3 × 3 × 4 = 36.  **Constraints:**  2 <= n <= 58 |
| 164. | Given an integer array nums and an integer k, return the k most frequent elements. You may return the answer in any order.  **Example 1:**  Input: nums = [1,1,1,2,2,3], k = 2 Output: [1,2] |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: nums = [1], k = 1 Output: [1]  **Constraints:**  1 <= nums.length <= 10^5  -10^4 <= nums[i] <= 10^4  k is in the range [1, the number of unique elements in the array]. It is guaranteed that the answer is unique. |
| 165. | Your task is to calculate ab mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.  **Example 1:**  Input: a = 2, b = [3] Output: 8  **Example 2:**  Input: a = 2, b = [1,0] Output: 1024  **Example 3:**  Input: a = 1, b = [4,3,3,8,5,2]  Output: 1  **Constraints:**  1 <= a <= 231 - 1  1 <= b.length <= 2000  0 <= b[i] <= 9  b does not contain leading zeros. |
| 166. | You are given two integer arrays nums1 and nums2 sorted in non- decreasing order and an integer k.  Define a pair (u, v) which consists of one element from the first array and one element from the second array.  Return the k pairs (u1, v1), (u2, v2), ..., (uk, vk) with the smallest sums.  **Example 1:**  Input: nums1 = [1,7,11], nums2 = [2,4,6], k = 3 Output: [[1,2],[1,4],[1,6]]  Explanation: The first 3 pairs are returned from the sequence: [1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],[11,4],[11,6]  **Example 2:**  Input: nums1 = [1,1,2], nums2 = [1,2,3], k = 2 Output: [[1,1],[1,1]]  Explanation: The first 2 pairs are returned from the sequence: [1,1],[1,1],[1,2],[2,1],[1,2],[2,2],[1,3],[1,3],[2,3] |

|  |  |
| --- | --- |
|  | **Constraints:**  1 <= nums1.length, nums2.length <= 10^5  -10^9 <= nums1[i], nums2[i] <= 10^9  nums1 and nums2 both are sorted in non-decreasing order. 1 <= k <= 10^4 |
| 167. | A wiggle sequence is a sequence where the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with one element and a sequence with two non-equal elements are trivially wiggle sequences.  For example, [1, 7, 4, 9, 2, 5] is a wiggle sequence because the differences (6,  -3, 5, -7, 3) alternate between positive and negative.  In contrast, [1, 4, 7, 2, 5] and [1, 7, 4, 5, 5] are not wiggle sequences. The first is not because its first two differences are positive, and the second is not because its last difference is zero.  A subsequence is obtained by deleting some elements (possibly zero) from the original sequence, leaving the remaining elements in their original order. Given an integer array nums, return the length of the longest wiggle subsequence of nums.  **Example 1:**  Input: nums = [1,7,4,9,2,5] Output: 6  Explanation: The entire sequence is a wiggle sequence with differences (6, - 3, 5, -7, 3).  **Example 2:**  Input: nums = [1,17,5,10,13,15,10,5,16,8]  Output: 7  Explanation: There are several subsequences that achieve this length. One is [1, 17, 10, 13, 10, 16, 8] with differences (16, -7, 3, -3, 6, -8).  **Constraints:**  1 <= nums.length <= 1000  0 <= nums[i] <= 1000 |
| 168. | Given an array of distinct integers nums and a target integer target, return the number of possible combinations that add up to target.  The test cases are generated so that the answer can fit in a 32-bit integer.  **Example 1:**  Input: nums = [1,2,3], target = 4 Output: 7  Explanation:  The possible combination ways are: (1, 1, 1, 1) |

|  |  |
| --- | --- |
|  | (1, 1, 2)  (1, 2, 1)  (1, 3)  (2, 1, 1)  (2, 2)  (3, 1)  Note that different sequences are counted as different combinations.  **Example 2:**  Input: nums = [9], target = 3 Output: 0  **Constraints:**  1 <= nums.length <= 200  1 <= nums[i] <= 1000  All the elements of nums are unique. 1 <= target <= 1000 |
| 169. | You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm on arr:  Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.  Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.  Keep repeating the steps again, alternating left to right and right to left, until a single number remains.  Given the integer n, return the last number that remains in arr.  **Example 1:**  Input: n = 9 Output: 6 Explanation:  arr = [1, 2, 3, 4, 5, 6, 7, 8, 9]  arr = [2, 4, 6, 8]  arr = [2, 6]  arr = [6]  **Example 2:** Input: n = 1 Output: 1  **Constraints:**  1 <= n <= 10^9 |
| 170. | Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k.  if no such substring exists, return 0. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: s = "aaabb", k = 3 Output: 3  Explanation: The longest substring is "aaa", as 'a' is repeated 3 times.  **Example 2:**  Input: s = "ababbc", k = 2 Output: 5  Explanation: The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.  **Constraints:**  1 <= s.length <= 10^4  s consists of only lowercase English letters. 1 <= k <= 10^5 |
| 171. | Given a positive integer n, you can apply one of the following operations: If n is even, replace n with n / 2.  If n is odd, replace n with either n + 1 or n - 1.  Return the minimum number of operations needed for n to become 1.  **Example 1:**  Input: n = 8 Output: 3  Explanation: 8 -> 4 -> 2 -> 1  **Example 2:**  Input: n = 7 Output: 4  Explanation: 7 -> 8 -> 4 -> 2 -> 1  or 7 -> 6 -> 3 -> 2 -> 1  **Example 3:** Input: n = 4 Output: 2  **Constraints:**  1 <= n <= 2^31 - 1 |
| 172. | Given an integer n, return the nth digit of the infinite integer sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...].  **Example 1:**  Input: n = 3 Output: 3  **Example 2:**  Input: n = 11 Output: 0  Explanation: The 11th digit of the sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...  is a 0, which is part of the number 10. |

|  |  |
| --- | --- |
|  | **Constraints:**  1 <= n <= 2^31 - 1 |
| 173. | Given string num representing a non-negative integer num, and an integer k, return the smallest possible integer after removing k digits from num.  **Example 1:**  Input: num = "1432219", k = 3 Output: "1219"  Explanation: Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.  **Example 2:**  Input: num = "10200", k = 1 Output: "200"  Explanation: Remove the leading 1 and the number is 200. Note that the output must not contain leading zeroes.  **Constraints:**  1 <= k <= num.length <= 10^5 num consists of only digits.  num does not have any leading zeros except for the zero itself. |
| 174. | An integer array is called arithmetic if it consists of at least three elements and if the difference between any two consecutive elements is the same.  For example, [1,3,5,7,9], [7,7,7,7], and [3,-1,-5,-9] are arithmetic sequences. Given an integer array nums, return the number of arithmetic subarrays of nums.  A subarray is a contiguous subsequence of the array.  **Example 1:**  Input: nums = [1,2,3,4] Output: 3  Explanation: We have 3 arithmetic slices in nums: [1, 2, 3], [2, 3, 4] and  [1,2,3,4] itself.  **Example 2:**  Input: nums = [1] Output: 0  **Constraints:**  1 <= nums.length <= 5000  -1000 <= nums[i] <= 1000 |
| 175. | Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: nums = [1,5,11,5] Output: true  Explanation: The array can be partitioned as [1, 5, 5] and [11].  **Example 2:**  Input: nums = [1,2,3,5] Output: false  Explanation: The array cannot be partitioned into equal sum subsets.  **Constraints:**  1 <= nums.length <= 200  1 <= nums[i] <= 100 |
| 176. | You are given a string s and an integer k. You can choose any character of the string and change it to any other uppercase English character. You can perform this operation at most k times.  Return the length of the longest substring containing the same letter you can get after performing the above operations.  **Example 1:**  Input: s = "ABAB", k = 2 Output: 4  Explanation: Replace the two 'A's with two 'B's or vice versa.  **Example 2:**  Input: s = "AABABBA", k = 1  Output: 4  Explanation: Replace the one 'A' in the middle with 'B' and form "AABBBBA". The substring "BBBB" has the longest repeating letters, which is 4.  There may exists other ways to achieve this answer too.  **Constraints:**  1 <= s.length <= 10^5  s consists of only uppercase English letters. 0 <= k <= s.length |
| 177. | Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the answer in any order.  An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.  **Example 1:**  Input: s = "cbaebabacd", p = "abc" Output: [0,6]  Explanation:  The substring with start index = 0 is "cba", which is an anagram of "abc". The substring with start index = 6 is "bac", which is an anagram of "abc". **Example 2:** |

|  |  |
| --- | --- |
|  | Input: s = "abab", p = "ab" Output: [0,1,2] Explanation:  The substring with start index = 0 is "ab", which is an anagram of "ab". The substring with start index = 1 is "ba", which is an anagram of "ab". The substring with start index = 2 is "ab", which is an anagram of "ab".  **Constraints:**  1 <= s.length, p.length <= 3 \* 10^4  s and p consist of lowercase English letters. |
| 178. | Given an array of characters chars, compress it using the following algorithm:  Begin with an empty string s. For each group of consecutive repeating characters in chars:  If the group's length is 1, append the character to s.  Otherwise, append the character followed by the group's length.  The compressed string s should not be returned separately, but instead, be stored in the input character array chars. Note that group lengths that are 10 or longer will be split into multiple characters in chars.  After you are done modifying the input array, return the new length of the array.  You must write an algorithm that uses only constant extra space.  **Example 1:**  Input: chars = ["a","a","b","b","c","c","c"]  Output: Return 6, and the first 6 characters of the input array should be: ["a","2","b","2","c","3"]  Explanation: The groups are "aa", "bb", and "ccc". This compresses to "a2b2c3".  **Example 2:**  Input: chars = ["a"]  Output: Return 1, and the first character of the input array should be: ["a"] Explanation: The only group is "a", which remains uncompressed since it's a single character.  **Constraints:**  1 <= chars.length <= 2000  chars[i] is a lowercase English letter, uppercase English letter, digit, or symbol. |
| 179. | You are given two non-empty linked lists representing two non-negative integers. The most significant digit comes first and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.  You may assume the two numbers do not contain any leading zero, except |

|  |  |
| --- | --- |
|  | the number 0 itself.  **Example 1:**    Input: l1 = [7,2,4,3], l2 = [5,6,4]  Output: [7,8,0,7]  **Example 2:**  Input: l1 = [2,4,3], l2 = [5,6,4]  Output: [8,0,7]  **Constraints:**  The number of nodes in each linked list is in the range [1, 100]. 0 <= Node.val <= 9  It is guaranteed that the list represents a number that does not have leading zeros. |
| 180. | Given four integer arrays nums1, nums2, nums3, and nums4 all of length n, return the number of tuples (i, j, k, l) such that:  0 <= i, j, k, l < n  nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0  **Example 1:**  Input: nums1 = [1,2], nums2 = [-2,-1], nums3 = [-1,2], nums4 = [0,2]  Output: 2 Explanation:  The two tuples are:  1. (0, 0, 0, 1) -> nums1[0] + nums2[0] + nums3[0] + nums4[1] = 1 + (-2) + (-1)  + 2 = 0  2. (1, 1, 0, 0) -> nums1[1] + nums2[1] + nums3[0] + nums4[0] = 2 + (-1) + (-1)  + 0 = 0  **Example 2:**  Input: nums1 = [0], nums2 = [0], nums3 = [0], nums4 = [0]  Output: 1  **Constraints:**  n == nums1.length n == nums2.length n == nums3.length  n == nums4.length |

|  |  |
| --- | --- |
|  | 1 <= n <= 200  -2^28 <= nums1[i], nums2[i], nums3[i], nums4[i] <= 2^28 |
| 181. | Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.  Return the sorted string. If there are multiple answers, return any of them.  **Example 1:**  Input: s = "tree" Output: "eert"  Explanation: 'e' appears twice while 'r' and 't' both appear once.  So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.  **Example 2:**  Input: s = "cccaaa" Output: "aaaccc"  Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.  Note that "cacaca" is incorrect, as the same characters must be together.  **Constraints:**  1 <= s.length <= 5 \* 10^5  s consists of uppercase and lowercase English letters and digits. |
| 182. | Given an array of n integers nums, a 132 pattern is a subsequence of three integers nums[i], nums[j] and nums[k] such that i < j < k and nums[i] < nums[k] < nums[j].  Return true if there is a 132 pattern in nums, otherwise, return false.  **Example 1:**  Input: nums = [1,2,3,4] Output: false  Explanation: There is no 132 pattern in the sequence.  **Example 2:**  Input: nums = [3,1,4,2] Output: true  Explanation: There is a 132 pattern in the sequence: [1, 4, 2].  **Example 3:**  Input: nums = [-1,3,2,0  ]Output: true  Explanation: There are three 132 patterns in the sequence: [-1, 3, 2], [-1, 3,  0] and [-1, 2, 0].  **Constraints:**  n == nums.length 1 <= n <= 2 \* 10^5  -10^9 <= nums[i] <= 10^9 |

|  |  |
| --- | --- |
|  |  |
| 183. | Given a string queryIP, return "IPv4" if IP is a valid IPv4 address, "IPv6" if IP is a valid IPv6 address or "Neither" if IP is not a correct IP of any type.  A valid IPv4 address is an IP in the form "x1.x2.x3.x4" where 0 <= xi <=  255 and xi cannot contain leading zeros. For example, "192.168.1.1" and "192.168.1.0" are valid IPv4 addresses while "192.168.01.1", "192.168.1.00", and "[192.168@1.1](mailto:192.168@1.1)" are invalid IPv4 addresses.  A valid IPv6 address is an IP in the form "x1:x2:x3:x4:x5:x6:x7:x8" where:   * 1 <= xi.length <= 4 * xi is a hexadecimal string which may contain digits, lowercase English letter ('a' to 'f') and upper-case English letters ('A' to 'F'). * Leading zeros are allowed in xi.   For example, "2001:0db8:85a3:0000:0000:8a2e:0370:7334" and "2001:db8:85a3:0:0:8A2E:0370:7334" are valid IPv6 addresses, while "2001:0db8:85a3::8A2E:037j:7334" and  "02001:0db8:85a3:0000:0000:8a2e:0370:7334" are invalid IPv6 addresses.  **Example 1:**  Input: queryIP = "172.16.254.1" Output: "IPv4"  Explanation: This is a valid IPv4 address, return "IPv4".  **Example 2:**  Input: queryIP = "2001:0db8:85a3:0:0:8A2E:0370:7334"  Output: "IPv6"  Explanation: This is a valid IPv6 address, return "IPv6".  **Example 3:**  Input: queryIP = "256.256.256.256"  Output: "Neither"  Explanation: This is neither a IPv4 address nor a IPv6 address.  **Constraints:**  queryIP consists only of English letters, digits and the characters '.' and ':'. |
| 184. | We define the string base to be the infinite wraparound string of "abcdefghijklmnopqrstuvwxyz", so base will look like this: "...zabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd ".  Given a string s, return the number of unique non-empty substrings of s are present in base.  **Example 1:**  Input: s = "a" Output: 1  Explanation: Only the substring "a" of s is in base.  **Example 2:** |

|  |  |
| --- | --- |
|  | Input: s = "cac" Output: 2  Explanation: There are two substrings ("a", "c") of s in base.  **Example 3:**  Input: s = "zab" Output: 6  Explanation: There are six substrings ("z", "a", "b", "za", "ab", and "zab") of s in base.  **Constraints:**  1 <= s.length <= 10^5  s consists of lowercase English letters. |
| 185. | Winter is coming! During the contest, your first job is to design a standard heater with a fixed warm radius to warm all the houses.  Every house can be warmed, as long as the house is within the heater's warm radius range.  Given the positions of houses and heaters on a horizontal line, return the minimum radius standard of heaters so that those heaters could cover all houses.  Notice that all the heaters follow your radius standard, and the warm radius will the same.  **Example 1:**  Input: houses = [1,2,3], heaters = [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: houses = [1,2,3,4], heaters = [1,4] Output: 1  Explanation: The two heaters were placed at positions 1 and 4. We need to use a radius 1 standard, then all the houses can be warmed.  **Example 3:**  Input: houses = [1,5], heaters = [2] Output: 3  **Constraints:**  1 <= houses.length, heaters.length <= 3 \* 10^4 1 <= houses[i], heaters[i] <= 10^9 |
| 186. | A magical string s consists of only '1' and '2' and obeys the following rules: The string s is magical because concatenating the number of contiguous occurrences of characters '1' and '2' generates the string s itself.  The first few elements of s is s = "1221121221221121122……". If we group the consecutive 1's and 2's in s, it will be "1 22 11 2 1 22 1 22 11 2 11 |

|  |  |
| --- | --- |
|  | 22 " and the occurrences of 1's or 2's in each group are "1 2 2 1 1 2 1 2 2  1 2 2 ". You can see that the occurrence sequence is s itself.  Given an integer n, return the number of 1's in the first n number in the magical string s.  **Example 1:**  Input: n = 6 Output: 3  Explanation: The first 6 elements of magical string s is "122112" and it contains three 1's, so return 3.  **Example 2:** Input: n = 1 Output: 1  **Constraints:**  1 <= n <= 10^5 |
| 187. | You are given an integer array nums and an integer target.  You want to build an expression out of nums by adding one of the symbols '+' and '-' before each integer in nums and then concatenate all the integers.  For example, if nums = [2, 1], you can add a '+' before 2 and a '- ' before 1 and concatenate them to build the expression "+2-1".  Return the number of different expressions that you can build, which evaluates to target.  **Example 1:**  Input: nums = [1,1,1,1,1], target = 3 Output: 5  Explanation: There are 5 ways to assign symbols to make the sum of nums be target 3.  -1 + 1 + 1 + 1 + 1 = 3  +1 - 1 + 1 + 1 + 1 = 3  +1 + 1 - 1 + 1 + 1 = 3  +1 + 1 + 1 - 1 + 1 = 3  +1 + 1 + 1 + 1 - 1 = 3  **Example 2:**  Input: nums = [1], target = 1 Output: 1  **Constraints:**  1 <= nums.length <= 20  0 <= nums[i] <= 1000  0 <= sum(nums[i]) <= 1000  -1000 <= target <= 1000 |

|  |  |
| --- | --- |
| 188. | Given an integer array nums and an integer k, return true if nums has a good subarray or false otherwise.  A good subarray is a subarray where:  its length is at least two, and  the sum of the elements of the subarray is a multiple of k. Note that:  A subarray is a contiguous part of the array.  An integer x is a multiple of k if there exists an integer n such that x = n \*  k. 0 is always a multiple of k.  **Example 1:**  Input: nums = [23,2,4,6,7], k = 6 Output: true  Explanation: [2, 4] is a continuous subarray of size 2 whose elements sum up to 6.  **Example 2:**  Input: nums = [23,2,6,4,7], k = 6 Output: true  Explanation: [23, 2, 6, 4, 7] is an continuous subarray of size 5 whose elements sum up to 42.  42 is a multiple of 6 because 42 = 7 \* 6 and 7 is an integer.  **Example 3:**  Input: nums = [23,2,6,4,7], k = 13 Output: false  **Constraints:**  1 <= nums.length <= 10^5 0 <= nums[i] <= 10^9  0 <= sum(nums[i]) <= 2^31 - 1 1 <= k <= 2^31 - 1 |
| 189. | Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0 and 1.  **Example 1:**  Input: nums = [0,1] Output: 2  Explanation: [0, 1] is the longest contiguous subarray with an equal number of 0 and 1.  **Example 2:**  Input: nums = [0,1,0] Output: 2  Explanation: [0, 1] (or [1, 0]) is a longest contiguous subarray with equal  number of 0 and 1.  **Constraints:**  1 <= nums.length <= 10^5 |

|  |  |
| --- | --- |
|  | nums[i] is either 0 or 1. |
| 190. | Given an array of integers nums and an integer k, return the number of unique k-diff pairs in the array.  A k-diff pair is an integer pair (nums[i], nums[j]), where the following are true:   * 0 <= i, j < nums.length * i != j * |nums[i] - nums[j]| == k   Notice that |val| denotes the absolute value of val.  **Example 1:**  Input: nums = [3,1,4,1,5], k = 2 Output: 2  Explanation: There are two 2-diff pairs in the array, (1, 3) and (3, 5). Although we have two 1s in the input, we should only return the number of unique pairs.  **Example 2:**  Input: nums = [1,2,3,4,5], k = 1 Output: 4  Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and  (4, 5).  **Example 3:**  Input: nums = [1,3,1,5,4], k = 0 Output: 1  Explanation: There is one 0-diff pair in the array, (1, 1).  **Constraints:**  1 <= nums.length <= 10^4  -10^7 <= nums[i] <= 10^7 0 <= k <= 10^7 |
| 191. | A [complex number](https://en.wikipedia.org/wiki/Complex_number) can be represented as a string on the form "real+imaginaryi" where:   * real is the real part and is an integer in the range [-100, 100]. * imaginary is the imaginary part and is an integer in the range [-100, 100]. i2 == -1.   Given two complex numbers num1 and num2 as strings, return a string of the complex number that represents their multiplications.  **Example 1:**  Input: num1 = "1+1i", num2 = "1+1i" Output: "0+2i"  Explanation: (1 + i) \* (1 + i) = 1 + i2 + 2 \* i = 2i, and you need convert it to the form of 0+2i.  **Example 2:**  Input: num1 = "1+-1i", num2 = "1+-1i" |

|  |  |
| --- | --- |
|  | Output: "0+-2i"  Explanation: (1 - i) \* (1 - i) = 1 + i2 - 2 \* i = -2i, and you need convert it to the form of 0+-2i.  **Constraints:**  num1 and num2 are valid complex numbers. |
| 192. | There is only one character 'A' on the screen of a notepad. You can perform one of two operations on this notepad for each step:  Copy All: You can copy all the characters present on the screen (a partial copy is not allowed).  Paste: You can paste the characters which are copied last time.  Given an integer n, return the minimum number of operations to get the character 'A' exactly n times on the screen.  **Example 1:**  Input: n = 3 Output: 3  Explanation: Initially, we have one character 'A'. In step 1, we use Copy All operation.  In step 2, we use Paste operation to get 'AA'. In step 3, we use Paste operation to get 'AAA'. **Example 2:**  Input: n = 1 Output: 0  **Constraints:**  1 <= n <= 1000 |
| 193. | Given a sorted integer array arr, two integers k and x, return the k closest integers to x in the array. The result should also be sorted in ascending order.  An integer a is closer to x than an integer b if:  |a - x| < |b - x|, or  |a - x| == |b - x| and a < b  **Example 1:**  Input: arr = [1,2,3,4,5], k = 4, x = 3 Output: [1,2,3,4]  **Example 2:**  Input: arr = [1,2,3,4,5], k = 4, x = -1 Output: [1,2,3,4]  **Constraints:**  1 <= k <= arr.length  1 <= arr.length <= 10^4  arr is sorted in ascending order.  -10^4 <= arr[i], x <= 10^4 |

|  |  |
| --- | --- |
|  |  |
| 194. | You have a data structure of employee information, including the employee's unique ID, importance value, and direct subordinates' IDs.  You are given an array of employees employees where:   * employees[i].id is the ID of the ith employee. * employees[i].importance is the importance value of the ith employee. * employees[i].subordinates is a list of the IDs of the direct subordinates of the ith employee.   Given an integer id that represents an employee's ID, return the total importance value of this employee and all their direct and indirect subordinates.  **Example 1:**    Input: employees = [[1,5,[2,3]],[2,3,[]],[3,3,[]]], id = 1  Output: 11  Explanation: Employee 1 has an importance value of 5 and has two direct subordinates: employee 2 and employee 3.  They both have an importance value of 3.  Thus, the total importance value of employee 1 is 5 + 3 + 3 = 11.  **Constraints:**   * 1 <= employees.length <= 2000 * 1 <= employees[i].id <= 2000 * All employees[i].id are unique. * -100 <= employees[i].importance <= 100 * One employee has at most one direct leader and may have several subordinates. * The IDs in employees[i].subordinates are valid IDs. |
| 195. | Given two strings s1 and s2, return the lowest ASCII sum of deleted characters to make two strings equal.  **Example 1:**  Input: s1 = "sea", s2 = "eat" Output: 231  Explanation: Deleting "s" from "sea" adds the ASCII value of "s" (115) to the sum.  Deleting "t" from "eat" adds 116 to the sum. |

|  |  |
| --- | --- |
|  | At the end, both strings are equal, and 115 + 116 = 231 is the minimum sum possible to achieve this.  **Example 2:**  Input: s1 = "delete", s2 = "leet" Output: 403  Explanation: Deleting "dee" from "delete" to turn the string into "let", adds 100[d] + 101[e] + 101[e] to the sum.  Deleting "e" from "leet" adds 101[e] to the sum.  At the end, both strings are equal to "let", and the answer is 100+101+101+101 = 403.  If instead we turned both strings into "lee" or "eet", we would get answers of 433 or 417, which are higher.  **Constraints:**  1 <= s1.length, s2.length <= 1000  s1 and s2 consist of lowercase English letters. |
| 196. | Given an array of integers nums and an integer k, return the number of contiguous subarrays where the product of all the elements in the subarray is strictly less than k.  **Example 1:**  Input: nums = [10,5,2,6], k = 100 Output: 8  Explanation: The 8 subarrays that have product less than 100 are: [10], [5], [2], [6], [10, 5], [5, 2], [2, 6], [5, 2, 6]  Note that [10, 5, 2] is not included as the product of 100 is not strictly less than k.  **Example 2:**  Input: nums = [1,2,3], k = 0 Output: 0  **Constraints:**  1 <= nums.length <= 3 \* 10^4 1 <= nums[i] <= 1000  0 <= k <= 10^6 |
| 197. | We are given an array asteroids of integers representing asteroids in a row. For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.  Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.  **Example 1:**  Input: asteroids = [5,10,-5] |

|  |  |
| --- | --- |
|  | Output: [5,10]  Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.  **Example 2:**  Input: asteroids = [8,-8] Output: []  Explanation: The 8 and -8 collide exploding each other.  **Example 3:**  Input: asteroids = [10,2,-5] Output: [10]  Explanation: The 2 and -5 collide resulting in -5. The 10 and -5 collide  resulting in 10.  **Constraints:**  2 <= asteroids.length <= 10^4  -1000 <= asteroids[i] <= 1000  asteroids[i] != 0 |
| 198. | You are standing at position 0 on an infinite number line. There is a destination at position target.  You can make some number of moves numMoves so that: On each move, you can either go left or right.  During the ith move (starting from i == 1 to i == numMoves), you take i steps in the chosen direction.  Given the integer target, return the minimum number of moves required (i.e., the minimum numMoves) to reach the destination.  **Example 1:**  Input: target = 2 Output: 3 Explanation:  On the 1st move, we step from 0 to 1 (1 step). On the 2nd move, we step from 1 to -1 (2 steps). On the 3rd move, we step from -1 to 2 (3 steps). **Example 2:**  Input: target = 3 Output: 2 Explanation:  On the 1st move, we step from 0 to 1 (1 step). On the 2nd move, we step from 1 to 3 (2 steps).  **Constraints:**  -10^9 <= target <= 10^9 target != 0 |
| 199. | Given a string s, rearrange the characters of s so that any two adjacent characters are not the same.  Return any possible rearrangement of s or return "" if not possible. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: s = "aab" Output: "aba" **Example 2:**  Input: s = "aaab" Output: ""  **Constraints:**  1 <= s.length <= 500  s consists of lowercase English letters. |
| 200. | Given a string s, you can transform every letter individually to be lowercase or uppercase to create another string.  Return a list of all possible strings we could create. Return the output in any order.  **Example 1:**  Input: s = "a1b2"  Output: ["a1b2","a1B2","A1b2","A1B2"]  **Example 2:**  Input: s = "3z4" Output: ["3z4","3Z4"]  **Constraints:**  1 <= s.length <= 12  s consists of lowercase English letters, uppercase English letters, and digits. |

|  |  |
| --- | --- |
| **S.NO** | **Description** |
| 201. | Roy wants to change his profile picture on Facebook. Now Facebook has |
|  | some restriction over the dimension of picture that we can upload. |
|  | Minimum dimension of the picture can be L x L, where L is the length of the |
|  | side of square. |
|  | Now Roy has N photos of various dimensions. |
|  | Dimension of a photo is denoted as W x H |
|  | where W - width of the photo and H - Height of the photo |
|  | When any photo is uploaded following events may occur: |
|  | [1] If any of the width or height is less than L, user is prompted to upload |
|  | another one. Print "UPLOAD ANOTHER" in this case. |
|  | [2] If width and height, both are large enough and |
|  | (a) if the photo is already square then it is accepted. Print "ACCEPTED" in |
|  | this case. |
|  | (b) else user is prompted to crop it. Print "CROP IT" in this case. |
|  | (quotes are only for clarification) |
|  | Given L, N, W and H as input, print appropriate text as output. |
|  | **Input:** |
|  | First line contains L. |
|  | Second line contains N, number of photos. |
|  | Following N lines each contains two space separated integers W and H. |
|  | **Output:** |
|  | Print appropriate text for each photo in a new line. |
|  | **Constraints:** |
|  | 1 <= L,W,H <= 10000 |
|  | 1 <= N <= 1000 |
|  | **Sample Input Sample Output** |
|  | 180 CROP IT |
|  | 3 UPLOAD ANOTHER |
|  | 640 480 ACCEPTED |
|  | 120 300 |
|  | 180 180 |
| 202. | We need the ability to divide an unknown integer into a given number of |

|  |  |
| --- | --- |
|  | even parts - or at least as even as they can be. The sum of the parts should be the original value, but each part should be an integer, and they should be as close as possible.  Complete the function so that it returns an array of integers representing the parts. If the input number is too small to split it into requested amount of parts, the additional parts should have value 0. Ignoring the order of the parts, there is only one valid solution for each input to your function!  **Example:**  splitInteger(20, 6) // returns [3, 3, 3, 3, 4, 4]  **Inputs**  The input to your function will always be valid for this kata. |
| 203. | Given the total number of persons n and a number k which indicates that k- 1 persons are skipped and kth person is killed in circle in a fixed direction.  After each operation, the count will start from k+1th person. The task is to choose the safe place in the circle so that when you perform these operations starting from 1st place in the circle, you are the last one remaining and survive.  **Example 1:**  Input:  n = 3 k = 2  Output: 3  Explanation: There are 3 persons so skipping 1 person i.e 1st person 2nd person will be killed. Thus the safe position is 3.  Example 2: Input:  n = 5 k = 3  Output: 4  Explanation: There are 5 persons so skipping 2 person i.e 3rd person will be killed. Thus the safe position is 4. |
| 204. | Write a program to find the transpose of a square matrix of size N\*N. Transpose of a matrix is obtained by changing rows to columns and columns to rows.  **Example 1: Input:**  N = 4  mat[][] = {{1, 1, 1, 1},  {2, 2, 2, 2}  {3, 3, 3, 3}  {4, 4, 4, 4}}  **Output:**  {{1, 2, 3, 4},  {1, 2, 3, 4}  {1, 2, 3, 4} |

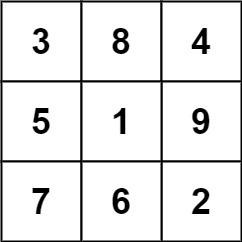
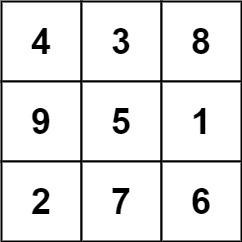
|  |  |
| --- | --- |
|  | {1, 2, 3, 4}}  **Example 2: Input:**  N = 2  mat[][] = {{1, 2},  {-9, -2}}  **Output:**  {{1, -9},  {2, -2}} |
| 205. | Given two four digit prime numbers, suppose 1033 and 8179, we need to find the shortest path from 1033 to 8179 by altering only single digit at a time such that every number that we get after changing a digit is prime. For example a solution is 1033, 1733, 3733, 3739, 3779, 8779, 8179  Input : 1033 8179  Output :6  Input : 1373 8017  Output : 7  Input : 1033 1033  Output : 0 |
| 206. | Given an array arr of n integers, write a function that returns true if there is a triplet (a, b, c) from the array (where a, b, and c are on different indexes) that satisfies a2 + b2 = c2, otherwise return false.  **Example 1:**  Input:  N = 5  Arr[] = {3, 2, 4, 6, 5}  Output: Yes  Explanation: a=3, b=4, and c=5 forms a pythagorean triplet.  **Example 2:**  Input:  N = 3  Arr[] = {3, 8, 5}  Output: No |
| 207. | Your task is to convert a number into a string that contains sounds corresponding to certain potential factors. A factor is a number that evenly divides into another number, leaving no remainder. The simplest way to test if one number is a factor of another is to use the [modulo operation](https://en.wikipedia.org/wiki/Modulo_operation).  The rules of Sound are that if a given number: |

|  |  |
| --- | --- |
|  | * has 3 as a factor, add 'Hehe' to the result. * has 5 as a factor, add 'hhee' to the result. * has 7 as a factor, add 'eehh' to the result. * have any of 3, 5, or 7 as a factor, the result should be the digits of the number.   **Examples**   * 28 has 7 as a factor, but not 3 or 5, so the result would be "eehh". * 30 has both 3 and 5 as factors, but not 7, so the result would be "hehehhee". * 34 is not factored by 3, 5, or 7, so the result would be "34". |
| 208. | We stack glasses in a pyramid, where the first row has 1 glass, the second row has 2 glasses, and so on until the 100th row. Each glass holds one cup of champagne.  Then, some champagne is poured into the first glass at the top. When the topmost glass is full, any excess liquid poured will fall equally to the glass immediately to the left and right of it. When those glasses become full, any excess champagne will fall equally to the left and right of those glasses, and so on. (A glass at the bottom row has its excess champagne fall on the floor.)  For example, after one cup of champagne is poured, the top most glass is full. After two cups of champagne are poured, the two glasses on the second row are half full. After three cups of champagne are poured, those two cups become full - there are 3 full glasses total now. After four cups of champagne are poured, the third row has the middle glass half full, and the two outside glasses are a quarter full, as pictured below.    Now after pouring some non-negative integer cups of champagne, return how full the jth glass in the ith row is (both i and j are 0-indexed.)  **Example 1:**  Input: poured = 1, query\_row = 1, query\_glass = 1 Output: 0.00000  Explanation: We poured 1 cup of champange to the top glass of the tower (which is indexed as (0, 0)). There will be no excess liquid so all the glasses under the top glass will remain empty.  **Example 2:**  Input: poured = 2, query\_row = 1, query\_glass = 1  Output: 0.50000 |

|  |  |
| --- | --- |
|  | Explanation: We poured 2 cups of champange to the top glass of the tower (which is indexed as (0, 0)). There is one cup of excess liquid. The glass indexed as (1, 0) and the glass indexed as (1, 1) will share the excess liquid equally, and each will get half cup of champange.  **Example 3:**  Input: poured = 100000009, query\_row = 33, query\_glass = 17 Output: 1.00000  **Constraints:**  0 <= poured <= 10^9  0 <= query\_glass <= query\_row < 100 |
| 209. | There are two types of soup: type A and type B. Initially, we have n ml of each type of soup. There are four kinds of operations:  Serve 100 ml of soup A and 0 ml of soup B, Serve 75 ml of soup A and 25 ml of soup B, Serve 50 ml of soup A and 50 ml of soup B, and Serve 25 ml of soup A and 75 ml of soup B.  When we serve some soup, we give it to someone, and we no longer have it. Each turn, we will choose from the four operations with an equal probability 0.25. If the remaining volume of soup is not enough to complete the operation, we will serve as much as possible. We stop once we no longer have some quantity of both types of soup.  Note that we do not have an operation where all 100 ml's of soup B are used first.  Return the probability that soup A will be empty first, plus half the probability that A and B become empty at the same time. Answers within 10-5 of the actual answer will be accepted.  **Example 1:**  Input: n = 50 Output: 0.62500  Explanation: If we choose the first two operations, A will become empty first.  For the third operation, A and B will become empty at the same time. For the fourth operation, B will become empty first.  So the total probability of A becoming empty first plus half the probability that A and B become empty at the same time, is 0.25 \* (1 + 1 + 0.5 + 0) = 0.625.  **Example 2:** Input: n = 100 Output: 0.71875  **Constraints:**  0 <= n <= 10^9 |

|  |  |
| --- | --- |
| 210. | You are given an integer array nums and an integer k. You can partition the array into at most k non-empty adjacent subarrays. The score of a partition is the sum of the averages of each subarray.  Note that the partition must use every integer in nums, and that the score is not necessarily an integer.  Return the maximum score you can achieve of all the possible partitions. Answers within 10-6 of the actual answer will be accepted.  **Example 1:**  Input: nums = [9,1,2,3,9], k = 3 Output: 20.00000 Explanation:  The best choice is to partition nums into [9], [1, 2, 3], [9]. The answer is 9 +  (1 + 2 + 3) / 3 + 9 = 20.  We could have also partitioned nums into [9, 1], [2], [3, 9], for example. That partition would lead to a score of 5 + 2 + 6 = 13, which is worse.  **Example 2:**  Input: nums = [1,2,3,4,5,6,7], k = 4 Output: 20.50000  **Constraints:**  1 <= nums.length <= 100 1 <= nums[i] <= 10^4  1 <= k <= nums.length |
| 211. | A 3 x 3 magic square is a 3 x 3 grid filled with distinct numbers from 1 to 9 such that each row, column, and both diagonals all have the same sum.  Given a row x col grid of integers, how many 3 x 3 "magic square" subgrids are there? (Each subgrid is contiguous).  Example 1:    Input: grid = [[4,3,8,4],[9,5,1,9],[2,7,6,2]]  Output: 1 Explanation: |

|  |  |
| --- | --- |
|  | The following subgrid is a 3 x 3 magic square:  while this one is not:  In total, there is only one magic square inside the given grid.  **Example 2:** Input: grid = [[8]] Output: 0  **Constraints:**  row == grid.length col == grid[i].length 1 <= row, col <= 10  0 <= grid[i][j] <= 15 |
| 212. | You are given a string of digits num, such as "123456579". We can split it into a Fibonacci-like sequence [123, 456, 579].  Formally, a Fibonacci-like sequence is a list f of non-negative integers such that:  0 <= f[i] < 231, (that is, each integer fits in a 32-bit signed integer type), f.length >= 3, and  f[i] + f[i + 1] == f[i + 2] for all 0 <= i < f.length - 2.  Note that when splitting the string into pieces, each piece must not have extra leading zeroes, except if the piece is the number 0 itself.  Return any Fibonacci-like sequence split from num, or return [] if it cannot be done.  **Example 1:**  Input: num = "1101111" Output: [11,0,11,11]  Explanation: The output [110, 1, 111] would also be accepted.  **Example 2:**  Input: num = "11235813 0"Output: []  Explanation: The task is impossible.  **Example 3:**  Input: num = "0123" Output: []  Explanation: Leading zeroes are not allowed, so "01", "2", "3" is not valid. |



|  |  |
| --- | --- |
|  | **Constraints:**  1 <= num.length <= 200 num contains only digits. |
| 213. | Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size groupSize, and consists of groupSize consecutive cards.  Given an integer array hand where hand[i] is the value written on the ith card and an integer groupSize, return true if she can rearrange the cards, or false otherwise.  **Example 1:**  Input: hand = [1,2,3,6,2,3,4,7,8], groupSize = 3 Output: true  Explanation: Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]  **Example 2:**  Input: hand = [1,2,3,4,5], groupSize = 4 Output: false  Explanation: Alice's hand can not be rearranged into groups of 4.  **Constraints:**  1 <= hand.length <= 10^4 0 <= hand[i] <= 10^9  1 <= groupSize <= hand.length |
| 214. | You are given a string s of lowercase English letters and an integer array shifts of the same length.  Call the shift() of a letter, the next letter in the alphabet, (wrapping around so that 'z' becomes 'a').  For example, shift('a') = 'b', shift('t') = 'u', and shift('z') = 'a'.  Now for each shifts[i] = x, we want to shift the first i + 1 letters of s, x times. Return the final string after all such shifts to s are applied.  **Example 1:**  Input: s = "abc", shifts = [3,5,9] Output: "rpl"  Explanation: We start with "abc".  After shifting the first 1 letters of s by 3, we have "dbc". After shifting the first 2 letters of s by 5, we have "igc".  After shifting the first 3 letters of s by 9, we have "rpl", the answer.  **Example 2:**  Input: s = "aaa", shifts = [1,2,3] Output: "gfd"  **Constraints:**  1 <= s.length <= 10^5  s consists of lowercase English letters. shifts.length == s.length |

|  |  |
| --- | --- |
|  | 0 <= shifts[i] <= 10^9 |
| 215. | A car fleet is some non-empty set of cars driving at the same position and same speed. Note that a single car is also a car fleet.  If a car catches up to a car fleet right at the destination point, it will still be considered as one car fleet.  Return the number of car fleets that will arrive at the destination.  **Example 1:**  Input: target = 12, position = [10,8,0,5,3], speed = [2,4,1,1,3] Output: 3  Explanation:  The cars starting at 10 (speed 2) and 8 (speed 4) become a fleet, meeting each other at 12.  The car starting at 0 does not catch up to any other car, so it is a fleet by itself.  The cars starting at 5 (speed 1) and 3 (speed 3) become a fleet, meeting each other at 6. The fleet moves at speed 1 until it reaches target.  Note that no other cars meet these fleets before the destination, so the answer is 3.  **Example 2:**  Input: target = 10, position = [3], speed = [3] Output: 1  Explanation: There is only one car, hence there is only one fleet.  **Constraints:**  n == position.length == speed.length 1 <= n <= 10^5  0 < target <= 10^6  0 <= position[i] < target  All the values of position are unique. 0 < speed[i] <= 10^6 |
| 216. | There is a special square room with mirrors on each of the four walls. Except for the southwest corner, there are receptors on each of the remaining corners, numbered 0, 1, and 2.  The square room has walls of length p and a laser ray from the southwest corner first meets the east wall at a distance q from the 0th receptor.  Given the two integers p and q, return the number of the receptor that the ray meets first.  The test cases are guaranteed so that the ray will meet a receptor eventually.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: p = 2, q = 1 Output: 2  Explanation: The ray meets receptor 2 the first time it gets reflected back to the left wall.  **Example 2:** Input: p = 3, q = 1 Output: 1  **Constraints:**  1 <= q <= p <= 1000 |
| 217. | Given an integer n, return the smallest prime palindrome greater than or equal to n.  An integer is prime if it has exactly two divisors: 1 and itself. Note that 1 is not a prime number.  For example, 2, 3, 5, 7, 11, and 13 are all primes.  An integer is a palindrome if it reads the same from left to right as it does from right to left.  For example, 101 and 12321 are palindromes.  The test cases are generated so that the answer always exists and is in the range [2, 2 \* 108].  **Example 1:** Input: n = 6 Output: 7  **Example 2:** Input: n = 8 Output: 11  **Example 3:** Input: n = 13 Output: 101  **Constraints:**  1 <= n <= 10^8 |
| 218. | Given the root of a binary tree, the depth of each node is the shortest distance to the root.  Return the smallest subtree such that it contains all the deepest nodes in the original tree.  A node is called the deepest if it has the largest depth possible among any |

|  |  |
| --- | --- |
|  | node in the entire tree.  The subtree of a node is a tree consisting of that node, plus the set of all descendants of that node.  **Example 1:**    Input: root = [3,5,1,6,2,0,8,null,null,7,4] Output: [2,7,4]  Explanation: We return the node with value 2, colored in yellow in the diagram.  The nodes coloured in blue are the deepest nodes of the tree.  Notice that nodes 5, 3 and 2 contain the deepest nodes in the tree but node 2 is the smallest subtree among them, so we return it.  **Example 2:**  Input: root = [1] Output: [1]  Explanation: The root is the deepest node in the tree.  **Constraints:**  The number of nodes in the tree will be in the range [1, 500]. 0 <= Node.val <= 500  The values of the nodes in the tree are unique. |
| 219. | Given a list of strings words and a string pattern, return a list of words[i] that match pattern. You may return the answer in any order.  A word matches the pattern if there exists a permutation of letters p so that after replacing every letter x in the pattern with p(x), we get the desired word.  Recall that a permutation of letters is a bijection from letters to letters: every letter maps to another letter, and no two letters map to the same letter.  **Example 1:**  Input: words = ["abc","deq","mee","aqq","dkd","ccc"], pattern = "abb" Output: ["mee","aqq"]  Explanation: "mee" matches the pattern because there is a permutation {a -  > m, b -> e, ...}.  "ccc" does not match the pattern because {a -> c, b -> c, ...} is not a  permutation, since a and b map to the same letter. |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: words = ["a","b","c"], pattern = "a"  Output: ["a","b","c"]  **Constraints:**  1 <= pattern.length <= 20  1 <= words.length <= 50 words[i].length == pattern.length  pattern and words[i] are lowercase English letters. |
| 220. | You are visiting a farm that has a single row of fruit trees arranged from left to right. The trees are represented by an integer array fruits where fruits[i] is the type of fruit the ith tree produces.  You want to collect as much fruit as possible. However, the owner has some strict rules that you must follow:  You only have two baskets, and each basket can only hold a single type of fruit. There is no limit on the amount of fruit each basket can hold.  Starting from any tree of your choice, you must pick exactly one fruit from every tree (including the start tree) while moving to the right. The picked fruits must fit in one of your baskets.  Once you reach a tree with fruit that cannot fit in your baskets, you must stop.  Given the integer array fruits, return the maximum number of fruits you can pick.  **Example 1:**  Input: fruits = [1,2,1] Output: 3  Explanation: We can pick from all 3 trees.  **Example 2:**  Input: fruits = [0,1,2,2] Output: 3  Explanation: We can pick from trees [1,2,2].  If we had started at the first tree, we would only pick from trees [0,1].  **Example 3:**  Input: fruits = [1,2,3,2,2] Output: 4  Explanation: We can pick from trees [2,3,2,2].  If we had started at the first tree, we would only pick from trees [1,2].  **Constraints:**  1 <= fruits.length <= 10^5 0 <= fruits[i] < fruits.length |
| 221. | You are given an n x n binary matrix grid where 1 represents land and 0 represents water.  An island is a 4-directionally connected group of 1's not connected to any |

|  |  |
| --- | --- |
|  | other 1's. There are exactly two islands in grid.  You may change 0's to 1's to connect the two islands to form one island. Return the smallest number of 0's you must flip to connect the two islands.  **Example 1:**  Input: grid = [[0,1],[1,0]] Output: 1  **Example 2:**  Input: grid = [[0,1,0],[0,0,0],[0,0,1]]  Output: 2  **Example 3:**  Input: grid = [[1,1,1,1,1],[1,0,0,0,1],[1,0,1,0,1],[1,0,0,0,1],[1,1,1,1,1]]  Output: 1  **Constraints:**  n == grid.length == grid[i].length 2 <= n <= 100  grid[i][j] is either 0 or 1.  There are exactly two islands in grid. |
| 222. | You have an initial power of power, an initial score of 0, and a bag of tokens where tokens[i] is the value of the ith token (0-indexed).  Your goal is to maximize your total score by potentially playing each token in one of two ways:  If your current power is at least tokens[i], you may play the ith token face up, losing tokens[i] power and gaining 1 score.  If your current score is at least 1, you may play the ith token face down, gaining tokens[i] power and losing 1 score.  Each token may be played at most once and in any order. You do not have to play all the tokens.  Return the largest possible score you can achieve after playing any number of tokens.  **Example 1:**  Input: tokens = [100], power = 50 Output: 0  Explanation: Playing the only token in the bag is impossible because you either have too little power or too little score.  **Example 2:**  Input: tokens = [100,200], power = 150 Output: 1  Explanation: Play the 0th token (100) face up, your power becomes 50 and score becomes 1.  There is no need to play the 1st token since you cannot play it face up to add to your score.  **Example 3:**  Input: tokens = [100,200,300,400], power = 200 |

|  |  |
| --- | --- |
|  | Output: 2  Explanation: Play the tokens in this order to get a score of 2:   1. Play the 0th token (100) face up, your power becomes 100 and score becomes 1. 2. Play the 3rd token (400) face down, your power becomes 500 and score becomes 0. 3. Play the 1st token (200) face up, your power becomes 300 and score becomes 1. 4. Play the 2nd token (300) face up, your power becomes 0 and score becomes 2.   **Constraints:**  0 <= tokens.length <= 1000 0 <= tokens[i], power < 10^4 |
| 223. | There are 8 prison cells in a row and each cell is either occupied or vacant. Each day, whether the cell is occupied or vacant changes according to the following rules:  If a cell has two adjacent neighbors that are both occupied or both vacant, then the cell becomes occupied.  Otherwise, it becomes vacant.  Note that because the prison is a row, the first and the last cells in the row can't have two adjacent neighbors.  You are given an integer array cells where cells[i] == 1 if the ith cell is occupied and cells[i] == 0 if the ith cell is vacant, and you are given an integer n.  Return the state of the prison after n days (i.e., n such changes described above).  **Example 1:**  Input: cells = [0,1,0,1,1,0,0,1], n = 7  Output: [0,0,1,1,0,0,0,0]  Explanation: The following table summarizes the state of the prison on each day:  Day 0: [0, 1, 0, 1, 1, 0, 0, 1]  Day 1: [0, 1, 1, 0, 0, 0, 0, 0]  Day 2: [0, 0, 0, 0, 1, 1, 1, 0]  Day 3: [0, 1, 1, 0, 0, 1, 0, 0]  Day 4: [0, 0, 0, 0, 0, 1, 0, 0]  Day 5: [0, 1, 1, 1, 0, 1, 0, 0]  Day 6: [0, 0, 1, 0, 1, 1, 0, 0]  Day 7: [0, 0, 1, 1, 0, 0, 0, 0]  **Example 2:**  Input: cells = [1,0,0,1,0,0,1,0], n = 1000000000 Output: [0,0,1,1,1,1,1,0]  **Constraints:**  cells.length == 8 |

|  |  |
| --- | --- |
|  | cells[i] is either 0 or 1. 1 <= n <= 10^9 |
| 224. | You have planned some train traveling one year in advance. The days of the year in which you will travel are given as an integer array days. Each day is an integer from 1 to 365.  Train tickets are sold in three different ways:  a 1-day pass is sold for costs[0] dollars,  a 7-day pass is sold for costs[1] dollars, and a 30-day pass is sold for costs[2] dollars.  The passes allow that many days of consecutive travel.  For example, if we get a 7-day pass on day 2, then we can travel for 7 days: 2, 3, 4, 5, 6, 7, and 8.  Return the minimum number of dollars you need to travel every day in the given list of days.  **Example 1:**  Input: days = [1,4,6,7,8,20], costs = [2,7,15] Output: 11  Explanation: For example, here is one way to buy passes that lets you travel your travel plan:  On day 1, you bought a 1-day pass for costs[0] = $2, which covered day 1. On day 3, you bought a 7-day pass for costs[1] = $7, which covered days 3, 4, ..., 9.  On day 20, you bought a 1-day pass for costs[0] = $2, which covered day 20. In total, you spent $11 and covered all the days of your travel.  **Example 2:**  Input: days = [1,2,3,4,5,6,7,8,9,10,30,31], costs = [2,7,15]  Output: 17  Explanation: For example, here is one way to buy passes that lets you travel your travel plan:  On day 1, you bought a 30-day pass for costs[2] = $15 which covered days 1, 2, ..., 30.  On day 31, you bought a 1-day pass for costs[0] = $2 which covered day 31. In total, you spent $17 and covered all the days of your travel.  **Constraints:**  1 <= days.length <= 365  1 <= days[i] <= 365  days is in strictly increasing order. costs.length == 3  1 <= costs[i] <= 1000 |
| 225. | Given two integers a and b, return any string s such that:  s has length a + b and contains exactly a 'a' letters, and exactly b 'b' letters, The substring 'aaa' does not occur in s, and  The substring 'bbb' does not occur in s. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: a = 1, b = 2 Output: "abb"  Explanation: "abb", "bab" and "bba" are all correct answers.  **Example 2:**  Input: a = 4, b = 1 Output: "aabaa"  **Constraints:**  0 <= a, b <= 100  It is guaranteed such an s exists for the given a and b. |
| 226. | You are given an m x n grid where each cell can have one of three values: 0 representing an empty cell,  1 representing a fresh orange, or 2 representing a rotten orange.  Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.  Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.  **Example 1:**    Input: grid = [[2,1,1],[1,1,0],[0,1,1]]  Output: 4  **Example 2:**  Input: grid = [[2,1,1],[0,1,1],[1,0,1]]  Output: -1  Explanation: The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens 4-directionally.  **Example 3:**  Input: grid = [[0,2]] Output: 0  Explanation: Since there are already no fresh oranges at minute 0, the answer is just 0.  **Constraints:**  m == grid.length n == grid[i].length 1 <= m, n <= 10  grid[i][j] is 0, 1, or 2. |
| 227. | You are given an array of strings equations that represent relationships |

|  |  |
| --- | --- |
|  | between variables where each string equations[i] is of length 4 and takes one of two different forms: "xi==yi" or "xi!=yi".Here, xi and yi are lowercase letters (not necessarily different) that represent one-letter variable names. Return true if it is possible to assign integers to variable names so as to satisfy all the given equations, or false otherwise.  **Example 1:**  Input: equations = ["a==b","b!=a"] Output: false  Explanation: If we assign say, a = 1 and b = 1, then the first equation is satisfied, but not the second.  There is no way to assign the variables to satisfy both equations.  **Example 2:**  Input: equations = ["b==a","a==b"] Output: true  Explanation: We could assign a = 1 and b = 1 to satisfy both equations.  **Constraints:**  1 <= equations.length <= 500  equations[i].length == 4 equations[i][0] is a lowercase letter. equations[i][1] is either '=' or '!'. equations[i][2] is '='.  equations[i][3] is a lowercase letter. |
| 228. | You are given an m x n binary matrix grid, where 0 represents a sea cell and 1 represents a land cell.  A move consists of walking from one land cell to another adjacent (4- directionally) land cell or walking off the boundary of the grid.  Return the number of land cells in grid for which we cannot walk off the boundary of the grid in any number of moves.  Example 1:    Input: grid = [[0,0,0,0],[1,0,1,0],[0,1,1,0],[0,0,0,0]]  Output: 3  Explanation: There are three 1s that are enclosed by 0s, and one 1 that is not enclosed because its on the boundary.  **Example 2:** |

|  |  |
| --- | --- |
|  | Input: grid = [[0,1,1,0],[0,0,1,0],[0,0,1,0],[0,0,0,0]]  Output: 0  Explanation: All 1s are either on the boundary or can reach the boundary.  **Constraints:**  m == grid.length n == grid[i].length 1 <= m, n <= 500  grid[i][j] is either 0 or 1. |
| 229. | You are given an m x n integer matrix grid, and three integers row, col, and color. Each value in the grid represents the color of the grid square at that location.  Two squares are called adjacent if they are next to each other in any of the 4 directions.  Two squares belong to the same connected component if they have the same color and they are adjacent.  The border of a connected component is all the squares in the connected component that are either adjacent to (at least) a square not in the component, or on the boundary of the grid (the first or last row or column). You should color the border of the connected component that contains the square grid[row][col] with color.  Return the final grid.  **Example 1:**  Input: grid = [[1,1],[1,2]], row = 0, col = 0, color = 3  Output: [[3,3],[3,2]]  **Example 2:**  Input: grid = [[1,2,2],[2,3,2]], row = 0, col = 1, color = 3  Output: [[1,3,3],[2,3,3]]  **Example 3:**  Input: grid = [[1,1,1],[1,1,1],[1,1,1]], row = 1, col = 1, color = 2  Output: [[2,2,2],[2,1,2],[2,2,2]]  **Constraints:**  m == grid.length n == grid[i].length  1 <= m, n <= 50 |

|  |  |
| --- | --- |
|  | 1 <= grid[i][j], color <= 1000 0 <= row < m  0 <= col < n |
| 230. | You are given two integer arrays nums1 and nums2. We write the integers of nums1 and nums2 (in the order they are given) on two separate horizontal lines.  We may draw connecting lines: a straight line connecting two numbers nums1[i] and nums2[j] such that:  nums1[i] == nums2[j], and  the line we draw does not intersect any other connecting (non-horizontal) line.  Note that a connecting line cannot intersect even at the endpoints (i.e., each number can only belong to one connecting line).  Return the maximum number of connecting lines we can draw in this way.  **Example 1:**    Input: nums1 = [1,4,2], nums2 = [1,2,4] Output: 2  Explanation: We can draw 2 uncrossed lines as in the diagram.  We cannot draw 3 uncrossed lines, because the line from nums1[1] = 4 to nums2[2] = 4 will intersect the line from nums1[2]=2 to nums2[1]=2.  **Example 2:**  Input: nums1 = [2,5,1,2,5], nums2 = [10,5,2,1,5,2]  Output: 3  **Example 3:**  Input: nums1 = [1,3,7,1,7,5], nums2 = [1,9,2,5,1]  Output: 2  **Constraints:**  1 <= nums1.length, nums2.length <= 500  1 <= nums1[i], nums2[j] <= 2000 |
| 231. | You are given an array of integers stones where stones[i] is the weight of the ith stone.  We are playing a game with the stones. On each turn, we choose any two stones and smash them together. Suppose the stones have weights x and y with x <= y. The result of this smash is:  If x == y, both stones are destroyed, and  If x != y, the stone of weight x is destroyed, and the stone of weight y has |

|  |  |
| --- | --- |
|  | new weight y - x.  At the end of the game, there is at most one stone left.  Return the smallest possible weight of the left stone. If there are no stones left, return 0.  **Example 1:**  Input: stones = [2,7,4,1,8,1] Output: 1  Explanation:  We can combine 2 and 4 to get 2, so the array converts to [2,7,1,8,1] then, we can combine 7 and 8 to get 1, so the array converts to [2,1,1,1] then, we can combine 2 and 1 to get 1, so the array converts to [1,1,1] then,  we can combine 1 and 1 to get 0, so the array converts to [1], then that's the optimal value.  **Example 2:**  Input: stones = [31,26,33,21,40] Output: 5  **Constraints:**  1 <= stones.length <= 30  1 <= stones[i] <= 100 |
| 232. | In a warehouse, there is a row of barcodes, where the ith barcode is barcodes[i].  Rearrange the barcodes so that no two adjacent barcodes are equal. You may return any answer, and it is guaranteed an answer exists.  **Example 1:**  Input: barcodes = [1,1,1,2,2,2] Output: [2,1,2,1,2,1]  **Example 2:**  Input: barcodes = [1,1,1,1,2,2,3,3]  Output: [1,3,1,3,1,2,1,2]  **Constraints:**  1 <= barcodes.length <= 10000  1 <= barcodes[i] <= 10000 |
| 233. | Given two numbers arr1 and arr2 in base -2, return the result of adding them together.  Each number is given in array format: as an array of 0s and 1s, from most significant bit to least significant bit. For example, arr = [1,1,0,1] represents the number (-2)^3 + (-2)^2 + (-2)^0 = -3. A number arr in array, format is also guaranteed to have no leading zeros: either arr == [0] or arr[0] == 1.  Return the result of adding arr1 and arr2 in the same format: as an array of 0s and 1s with no leading zeros. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: arr1 = [1,1,1,1,1], arr2 = [1,0,1] Output: [1,0,0,0,0]  Explanation: arr1 represents 11, arr2 represents 5, the output represents 16.  **Example 2:**  Input: arr1 = [0], arr2 = [0] Output: [0]  **Example 3:**  Input: arr1 = [0], arr2 = [1] Output: [1]  **Constraints:**  1 <= arr1.length, arr2.length <= 1000 arr1[i] and arr2[i] are 0 or 1  arr1 and arr2 have no leading zeros |
| 234. | There is a car with capacity empty seats. The vehicle only drives east (i.e., it cannot turn around and drive west).  You are given the integer capacity and an array trips where trips[i] = [numPassengersi, fromi, toi] indicates that the ith trip has numPassengersi passengers and the locations to pick them up and drop them off are fromi and toi respectively. The locations are given as the number of kilometers due east from the car's initial location.  Return true if it is possible to pick up and drop off all passengers for all the given trips, or false otherwise.  **Example 1:**  Input: trips = [[2,1,5],[3,3,7]], capacity = 4 Output: false  **Example 2:**  Input: trips = [[2,1,5],[3,3,7]], capacity = 5 Output: true  **Constraints:**  1 <= trips.length <= 1000  trips[i].length == 3  1 <= numPassengersi <= 100 0 <= fromi < toi <= 1000  1 <= capacity <= 10^5 |
| 235. | There are n flights that are labeled from 1 to n.  You are given an array of flight bookings bookings, where bookings[i] = [firsti, lasti, seatsi] represents a booking for flights firsti through lasti (inclusive) with seatsi seats reserved for each flight in the range.  Return an array answer of length n, where answer[i] is the total number of |

|  |  |
| --- | --- |
|  | seats reserved for flight i.  **Example 1:**  Input: bookings = [[1,2,10],[2,3,20],[2,5,25]], n = 5 Output: [10,55,45,25,25]  Explanation:  Flight labels: 1 2 3 4 5  Booking 1 reserved: 10 10  Booking 2 reserved: 20 20  Booking 3 reserved: 25 25 25 25  Total seats: 10 55 45 25 25 Hence, answer = [10,55,45,25,25] **Example 2:**  Input: bookings = [[1,2,10],[2,2,15]], n = 2 Output: [10,25]  Explanation:  Flight labels: 1 2  Booking 1 reserved: 10 10  Booking 2 reserved: 15  Total seats: 10 25 Hence, answer = [10,25]  **Constraints:**  1 <= n <= 2 \* 10^4  1 <= bookings.length <= 2 \* 10^4 bookings[i].length == 3  1 <= firsti <= lasti <= n 1 <= seatsi <= 10^4 |
| 236. | There are two kinds of threads: oxygen and hydrogen. Your goal is to group these threads to form water molecules.  There is a barrier where each thread has to wait until a complete molecule can be formed. Hydrogen and oxygen threads will be given releaseHydrogen and releaseOxygen methods respectively, which will allow them to pass the barrier. These threads should pass the barrier in groups of three, and they must immediately bond with each other to form a water molecule. You must guarantee that all the threads from one molecule bond before any other threads from the next molecule do.  In other words:  If an oxygen thread arrives at the barrier when no hydrogen threads are present, it must wait for two hydrogen threads.  If a hydrogen thread arrives at the barrier when no other threads are present, it must wait for an oxygen thread and another hydrogen thread.  We do not have to worry about matching the threads up explicitly; the threads do not necessarily know which other threads they are paired up with. The key is that threads pass the barriers in complete sets; thus, if we  examine the sequence of threads that bind and divide them into groups of |

|  |  |
| --- | --- |
|  | three, each group should contain one oxygen and two hydrogen threads. Write synchronization code for oxygen and hydrogen molecules that enforces these constraints.  **Example 1:**  Input: water = "HOH" Output: "HHO"  Explanation: "HOH" and "OHH" are also valid answers.  **Example 2:**  Input: water = "OOHHHH" Output: "HHOHHO"  Explanation: "HOHHHO", "OHHHHO", "HHOHOH", "HOHHOH", "OHHHOH",  "HHOOHH", "HOHOHH" and "OHHOHH" are also valid answers.  **Constraints:**  3 \* n == water.length 1 <= n <= 20  water[i] is either 'H' or 'O'.  There will be exactly 2 \* n 'H' in water. There will be exactly n 'O' in water. |
| 237. | We are given hours, a list of the number of hours worked per day for a given employee.  A day is considered to be a tiring day if and only if the number of hours worked is (strictly) greater than 8.  A well-performing interval is an interval of days for which the number of tiring days is strictly larger than the number of non-tiring days.  Return the length of the longest well-performing interval.  **Example 1:**  Input: hours = [9,9,6,0,6,6,9] Output: 3  Explanation: The longest well-performing interval is [9,9,6].  **Example 2:**  Input: hours = [6,6,6] Output: 0  **Constraints:**  1 <= hours.length <= 10^4 0 <= hours[i] <= 16 |
| 238. | Given two arrays of integers with equal lengths, return the maximum value of:  |arr1[i] - arr1[j]| + |arr2[i] - arr2[j]| + |i - j|  where the maximum is taken over all 0 <= i, j < arr1.length.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: arr1 = [1,2,3,4], arr2 = [-1,4,5,6]  Output: 13  **Example 2:**  Input: arr1 = [1,-2,-5,0,10], arr2 = [0,-2,-1,-7,-4]  Output: 20  **Constraints:**  2 <= arr1.length == arr2.length <= 40000  -10^6 <= arr1[i], arr2[i] <= 10^6 |
| 239. | On an alphabet board, we start at position (0, 0), corresponding to character board[0][0].  Here, board = ["abcde", "fghij", "klmno", "pqrst", "uvwxy", "z"], as shown in the diagram below.    We may make the following moves:  'U' moves our position up one row, if the position exists on the board; 'D' moves our position down one row, if the position exists on the board;  'L' moves our position left one column, if the position exists on the board; 'R' moves our position right one column, if the position exists on the board; '!' adds the character board[r][c] at our current position (r, c) to the answer.  (Here, the only positions that exist on the board are positions with letters on them.)  Return a sequence of moves that makes our answer equal to target in the minimum number of moves. You may return any path that does so.  **Example 1:**  Input: target = "leet"Output: "DDR!UURRR!!DDD!"  **Example 2:**  Input: target = "code"Output: "RR!DDRR!UUL!R!"  **Constraints:**  1 <= target.length <= 100  target consists only of English lowercase letters. |
| 240. | A transaction is possibly invalid if:  the amount exceeds $1000, or;  if it occurs within (and including) 60 minutes of another transaction with the same name in a different city. |

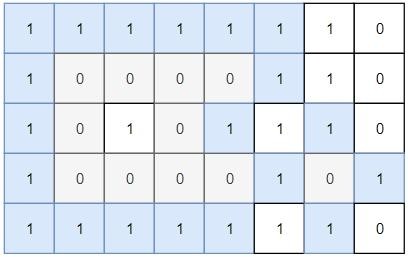
|  |  |
| --- | --- |
|  | You are given an array of strings transaction where transactions[i] consists of comma-separated values representing the name, time (in minutes), amount, and city of the transaction.  Return a list of transactions that are possibly invalid. You may return the answer in any order.  **Example 1:**  Input: transactions = ["alice,20,800,mtv","alice,50,100,beijing"] Output: ["alice,20,800,mtv","alice,50,100,beijing"]  Explanation: The first transaction is invalid because the second transaction occurs within a difference of 60 minutes, have the same name and is in a different city. Similarly the second one is invalid too.  **Example 2:**  Input: transactions = ["alice,20,800,mtv","alice,50,1200,mtv"] Output: ["alice,50,1200,mtv"]  **Example 3:**  Input: transactions = ["alice,20,800,mtv","bob,50,1200,mtv"] Output: ["bob,50,1200,mtv"]  **Constraints:**  transactions.length <= 1000  Each transactions[i] takes the form "{name},{time},{amount},{city}"  Each {name} and {city} consist of lowercase English letters, and have lengths between 1 and 10.  Each {time} consist of digits, and represent an integer between 0 and 1000. Each {amount} consist of digits, and represent an integer between 0 and 2000. |
| 241. | You are given a string s and array queries where queries[i] = [lefti, righti, ki]. We may rearrange the substring s[lefti...righti] for each query and then choose up to ki of them to replace with any lowercase English letter.  If the substring is possible to be a palindrome string after the operations above, the result of the query is true. Otherwise, the result is false.  Return a boolean array answer where answer[i] is the result of the ith query queries[i].  Note that each letter is counted individually for replacement, so if, for example s[lefti...righti] = "aaa", and ki = 2, we can only replace two of the letters. Also, note that no query modifies the initial string s.  **Example :**  Input: s = "abcda", queries = [[3,3,0],[1,2,0],[0,3,1],[0,3,2],[0,4,1]]  Output: [true,false,false,true,true] Explanation:  queries[0]: substring = "d", is palidrome. queries[1]: substring = "bc", is not palidrome.  queries[2]: substring = "abcd", is not palidrome after replacing only 1  character. |

|  |  |
| --- | --- |
|  | queries[3]: substring = "abcd", could be changed to "abba" which is palidrome. Also this can be changed to "baab" first rearrange it "bacd" then replace "cd" with "ab".  queries[4]: substring = "abcda", could be changed to "abcba" which is palidrome.  **Example 2:**  Input: s = "lyb", queries = [[0,1,0],[2,2,1]] Output: [false,true]  **Constraints:**  1 <= s.length, queries.length <= 10^5 0 <= lefti <= righti < s.length  0 <= ki <= s.length  s consists of lowercase English letters. |
| 242. | Given an integer array arr and an integer k, modify the array by repeating it k times.  For example, if arr = [1, 2] and k = 3 then the modified array will be [1, 2, 1, 2, 1, 2].  Return the maximum sub-array sum in the modified array. Note that the length of the sub-array can be 0 and its sum in that case is 0.  As the answer can be very large, return the answer modulo 109 + 7.  **Example 1:**  Input: arr = [1,2], k = 3 Output: 9  **Example 2:**  Input: arr = [1,-2,1], k = 5 Output: 2  **Example 3:**  Input: arr = [-1,-2], k = 7 Output: 0  **Constraints:**  1 <= arr.length <= 10^5 1 <= k <= 10^5  -10^4 <= arr[i] <= 10^4 |
| 243. | You are given a string s, and an array of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices(0-indexed) of the string. You can swap the characters at any pair of indices in the given pairs any number of times.  Return the lexicographically smallest string that s can be changed to after using the swaps.  **Example 1:**  Input: s = "dcab", pairs = [[0,3],[1,2]] |

|  |  |
| --- | --- |
|  | Output: "bacd" Explaination:  Swap s[0] and s[3], s = "bcad" Swap s[1] and s[2], s = "bacd" **Example 2:**  Input: s = "dcab", pairs = [[0,3],[1,2],[0,2]] Output: "abcd"  Explaination:  Swap s[0] and s[3], s = "bcad" Swap s[0] and s[2], s = "acbd" Swap s[1] and s[2], s = "abcd" **Example 3:**  Input: s = "cba", pairs = [[0,1],[1,2]] Output: "abc"  Explaination:  Swap s[0] and s[1], s = "bca"  Swap s[1] and s[2], s = "bac"  Swap s[0] and s[1], s = "abc"  **Constraints:**  1 <= s.length <= 10^5  0 <= pairs.length <= 10^5  0 <= pairs[i][0], pairs[i][1] < s.length  s only contains lower case English letters. |
| 244. | Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers.  Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After an individual philosopher finishes eating, they need to put down both forks so that the forks become available to others. A philosopher can take the fork on their right or the one on their left as they become available, but cannot start eating before getting both forks.  Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite demand are assumed.  Design a discipline of behaviour (a concurrent algorithm) such that no philosopher will starve; i.e., each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think. |

|  |  |
| --- | --- |
|  | The problem statement and the image above are taken from [wikipedia.org](https://en.wikipedia.org/wiki/Dining_philosophers_problem)  The philosophers' ids are numbered from 0 to 4 in a clockwise order. Implement the function void wantsToEat(philosopher, pickLeftFork, pickRightFork, eat, putLeftFork, putRightFork) where:  philosopher is the id of the philosopher who wants to eat.  pickLeftFork and pickRightFork are functions you can call to pick the corresponding forks of that philosopher.  eat is a function you can call to let the philosopher eat once he has picked both forks.  putLeftFork and putRightFork are functions you can call to put down the corresponding forks of that philosopher.  The philosophers are assumed to be thinking as long as they are not asking to eat (the function is not being called with their number).  Five threads, each representing a philosopher, will simultaneously use one object of your class to simulate the process. The function may be called for the same philosopher more than once, even before the last call ends.  **Example 1:**  Input: n = 1 Output:  [[4,2,1],[4,1,1],[0,1,1],[2,2,1],[2,1,1],[2,0,3],[2,1,2],[2,2,2],[4,0,3],[4,1,2],[0,2,  1],[4,2,2],[3,2,1],[3,1,1],[0,0,3],[0,1,2],[0,2,2],[1,2,1],[1,1,1],[3,0,3],[3,1,2],[3,  2,2],[1,0,3],[1,1,2],[1,2,2]]  Explanation:  n is the number of times each philosopher will call the function.  The output array describes the calls you made to the functions controlling the forks and the eat function, its format is:  output[i] = [a, b, c] (three integers)   * a is the id of a philosopher. * b specifies the fork: {1 : left, 2 : right}. * c specifies the operation: {1 : pick, 2 : put, 3 : eat}.   **Constraints:**  1 <= n <= 60 |

|  |  |
| --- | --- |
| 245. | You are given a string s of length n containing only four kinds of characters: 'Q', 'W', 'E', and 'R'.  A string is said to be balanced if each of its characters appears n / 4 times where n is the length of the string.  Return the minimum length of the substring that can be replaced with any other string of the same length to make s balanced. If s is already balanced, return 0.  **Example 1:**  Input: s = "QWER" Output: 0  Explanation: s is already balanced.  **Example 2:**  Input: s = "QQWE" Output: 1  Explanation: We need to replace a 'Q' to 'R', so that "RQWE" (or "QRWE") is balanced.  **Example 3:**  Input: s = "QQQW" Output: 2  Explanation: We can replace the first "QQ" to "ER".  **Constraints:**  n == s.length  4 <= n <= 10^5  n is a multiple of 4.  s contains only 'Q', 'W', 'E', and 'R'. |
| 246. | Given a 2D grid consists of 0s (land) and 1s (water). An island is a maximal 4-directionally connected group of 0s and a closed island is an island totally (all left, top, right, bottom) surrounded by 1s.  Return the number of closed islands. Example 1:  Input: grid = [[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],[1,0,1,0,1,1,1,0],[1,0,0,0,0,1,0,1],[1,1,1,1,1,  1,1,0]]  Output: 2 Explanation:  Islands in gray are closed because they are completely surrounded by water  (group of 1s). |



|  |  |
| --- | --- |
| 247. | An integer has sequential digits if and only if each digit in the number is one more than the previous digit.  Return a sorted list of all the integers in the range [low, high] inclusive that have sequential digits.  **Example 1:**  Input: low = 100, high = 300 Output: [123,234]  **Example 2:**  Input: low = 1000, high = 13000  Output: [1234,2345,3456,4567,5678,6789,12345]  **Constraints:**  10 <= low <= high <= 10^9 |
| 248. | Given an array of non-negative integers arr, you are initially positioned at start index of the array. When you are at index i, you can jump to i + arr[i] or i - arr[i], check if you can reach any index with value 0.  Notice that you can not jump outside of the array at any time.  **Example 1:**  Input: arr = [4,2,3,0,3,1,2], start = 5 Output: true  Explanation:  All possible ways to reach at index 3 with value 0 are:  index 5 -> index 4 -> index 1 -> index 3  index 5 -> index 6 -> index 4 -> index 1 -> index 3  **Example 2:**  Input: arr = [4,2,3,0,3,1,2], start = 0  Output: true Explanation: One possible way to reach at index 3 with value 0 is:  index 0 -> index 4 -> index 1 -> index 3  **Constraints:**  1 <= arr.length <= 5 \* 10^4 |
| 249. | Given a m x n matrix mat and an integer k, return a matrix answer where each answer[i][j] is the sum of all elements mat[r][c] for:  i - k <= r <= i + k,  j - k <= c <= j + k, and  (r, c) is a valid position in the matrix.  **Example 1:**  Input: mat = [[1,2,3],[4,5,6],[7,8,9]], k = 1  Output: [[12,21,16],[27,45,33],[24,39,28]]  **Example 2:**  Input: mat = [[1,2,3],[4,5,6],[7,8,9]], k = 2 |

|  |  |
| --- | --- |
|  | Output: [[45,45,45],[45,45,45],[45,45,45]]  **Constraints:**  m == mat.length n == mat[i].length  1 <= m, n, k <= 100  1 <= mat[i][j] <= 100 |
| 250. | You are given the root of a binary tree.  A ZigZag path for a binary tree is defined as follow:  Choose any node in the binary tree and a direction (right or left).  If the current direction is right, move to the right child of the current node; otherwise, move to the left child.  Change the direction from right to left or from left to right.  Repeat the second and third steps until you can't move in the tree.  Zigzag length is defined as the number of nodes visited - 1. (A single node has a length of 0).  Return the longest ZigZag path contained in that tree.  **Example 1:**    Input: root = [1,null,1,1,1,null,null,1,1,null,1,null,null,null,1] Output: 3  Explanation: Longest ZigZag path in blue nodes (right -> left -> right).  **Constraints:**  The number of nodes in the tree is in the range [1, 5 \* 10^4]. 1 <= Node.val <= 100 |
| 251. | Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]  Output: 6  Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.  **Example 2:**  Input: height = [4,2,0,3,2,5] Output: 9  **Constraints:**  n == height.length 1 <= n <= 2 \* 10^4  0 <= height[i] <= 10^5 |
| 252. | You are given a string s and an array of strings words. All the strings of words are of the same length.  A concatenated substring in s is a substring that contains all the strings of any permutation of words concatenated.  For example, if words = ["ab","cd","ef"], then "abcdef", "abefcd", "cdabef", "cdefab", "efabcd", and "efcdab" are all concatenated strings. "acdbef" is not a concatenated substring because it is not the concatenation of any permutation of words.  Return the starting indices of all the concatenated substrings in s. You can return the answer in any order.  **Example 1:**  Input: s = "barfoothefoobarman", words = ["foo","bar"] Output: [0,9]  Explanation: Since words.length == 2 and words[i].length == 3, the concatenated substring has to be of length 6.  The substring starting at 0 is "barfoo". It is the concatenation of ["bar","foo"] which is a permutation of words.  The substring starting at 9 is "foobar". It is the concatenation of ["foo","bar"] which is a permutation of words.  The output order does not matter. Returning [9,0] is fine too.  **Constraints:**  1 <= s.length <= 10^4  1 <= words.length <= 5000  1 <= words[i].length <= 30 |

|  |  |
| --- | --- |
|  | s and words[i] consist of lowercase English letters. |
| 253. | A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:  Every adjacent pair of words differs by a single letter.  Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.  sk == endWord  Given two words, beginWord and endWord, and a dictionary wordList, return all the shortest transformation sequences from beginWord to endWord, or an empty list if 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: There are 2 shortest transformation sequences:  "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: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.  **Constraints:**  1 <= beginWord.length <= 5 endWord.length == beginWord.length 1 <= wordList.length <= 500 wordList[i].length == beginWord.length  beginWord, endWord, and wordList[i] consist of lowercase English letters. beginWord != endWord  All the words in wordList are unique.  The sum of all shortest transformation sequences does not exceed 10^5. |
| 254. | There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.  You are giving candies to these children subjected to the following requirements:  Each child must have at least one candy.  Children with a higher rating get more candies than their neighbors.  Return the minimum number of candies you need to have to distribute the candies to the children. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: ratings = [1,0,2] Output: 5  Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.  **Example 2:**  Input: ratings = [1,2,2] Output: 4  Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.  The third child gets 1 candy because it satisfies the above two conditions.  **Constraints:**  n == ratings.length 1 <= n <= 2 \* 10^4  0 <= ratings[i] <= 2 \* 10^4 |
| 255. | Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,4,4,5,6,7] might become:  [4,5,6,7,0,1,4] if it was rotated 4 times. [0,1,4,4,5,6,7] if it was rotated 7 times.  Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].  Given the sorted rotated array nums that may contain duplicates, return the minimum element of this array.  You must decrease the overall operation steps as much as possible.  **Example 1:**  Input: nums = [1,3,5] Output: 1  **Example 2:**  Input: nums = [2,2,2,0,1] Output: 0  **Constraints:**  n == nums.length 1 <= n <= 5000  -5000 <= nums[i] <= 5000  nums is sorted and rotated between 1 and n times. |
| 256. | The demons had captured the princess and imprisoned her in the bottom- right corner of a dungeon. The dungeon consists of m x n rooms laid out in a 2D grid. Our valiant knight was initially positioned in the top-left room and must fight his way through dungeon to rescue the princess.  The knight has an initial health point represented by a positive integer. If at |

|  |  |
| --- | --- |
|  | any point his health point drops to 0 or below, he dies immediately.  Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).  To reach the princess as quickly as possible, the knight decides to move only rightward or downward in each step.  Return the knight's minimum initial health so that he can rescue the princess.  Note that any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.  **Example 1:**    Input: dungeon = [[-2,-3,3],[-5,-10,1],[10,30,-5]]  Output: 7  Explanation: The initial health of the knight must be at least 7 if he follows the optimal path: RIGHT-> RIGHT -> DOWN -> DOWN.  **Example 2:**  Input: dungeon = [[0]] Output: 1  **Constraints:**  m == dungeon.length n == dungeon[i].length 1 <= m, n <= 200  -1000 <= dungeon[i][j] <= 1000 |
| 257. | Given an m x n board of characters and a list of strings words, return all words on the board.  Each word must be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once in a word.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words =  ["oath","pea","eat","rain"] Output: ["eat","oath"] **Example 2:**    Input: board = [["a","b"],["c","d"]], words = ["abcb"] Output: []  **Constraints:**  m == board.length n == board[i].length 1 <= m, n <= 12  board[i][j] is a lowercase English letter. 1 <= words.length <= 3 \* 104  1 <= words[i].length <= 10  words[i] consists of lowercase English letters. All the strings of words are unique. |
| 258. | You are given an integer array nums and two integers indexDiff and valueDiff.  Find a pair of indices (i, j) such that: i != j,  abs(i - j) <= indexDiff.  abs(nums[i] - nums[j]) <= valueDiff, and  Return true if such pair exists or false otherwise.  **Example 1:**  Input: nums = [1,2,3,1], indexDiff = 3, valueDiff = 0 Output: true  Explanation: We can choose (i, j) = (0, 3). |

|  |  |
| --- | --- |
|  | We satisfy the three conditions:  i != j --> 0 != 3  abs(i - j) <= indexDiff --> abs(0 - 3) <= 3  abs(nums[i] - nums[j]) <= valueDiff --> abs(1 - 1) <= 0  **Example 2:**  Input: nums = [1,5,9,1,5,9], indexDiff = 2, valueDiff = 3 Output: false  Explanation: After trying all the possible pairs (i, j), we cannot satisfy the three conditions, so we return false.  **Constraints:**  2 <= nums.length <= 10^5  -109 <= nums[i] <= 10^9  1 <= indexDiff <= nums.length 0 <= valueDiff <= 10^9 |
| 259. | Given an integer n, count the total number of digit 1 appearing in all non- negative integers less than or equal to n.  **Example 1:** Input: n = 13 Output: 6  **Example 2:** Input: n = 0 Output: 0  **Constraints:**  0 <= n <= 10^9 |
| 260. | Convert a non-negative integer num to its English words representation.  **Example 1:**  Input: num = 123  Output: "One Hundred Twenty Three"  **Example 2:**  Input: num = 12345  Output: "Twelve Thousand Three Hundred Forty Five"  **Example 3:**  Input: num = 1234567  Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"  **Constraints:**  0 <= num <= 2^31 - 1 |
| 261. | Given a string s that contains parentheses and letters, remove the minimum  number of invalid parentheses to make the input string valid. |

|  |  |
| --- | --- |
|  | Return a list of unique strings that are valid with the minimum number of removals. You may return the answer in any order.  **Example 1:**  Input: s = "()())()" Output: ["(())()","()()()"]  **Example 2:**  Input: s = "(a)())()" Output: ["(a())()","(a)()()"]  **Example 3:**  Input: s = ")("  Output: [""]  **Constraints:**  1 <= s.length <= 25  s consists of lowercase English letters and parentheses '(' and ')'. There will be at most 20 parentheses in s. |
| 262. | You are given n balloons, indexed from 0 to n - 1. Each balloon is painted with a number on it represented by an array nums. You are asked to burst all the balloons.  If you burst the ith balloon, you will get nums[i - 1] \* nums[i] \* nums[i + 1] coins. If i - 1 or i + 1 goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it.  Return the maximum coins you can collect by bursting the balloons wisely.  **Example 1:**  Input: nums = [3,1,5,8] Output: 167 Explanation:  nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []  coins = 3\*1\*5 + 3\*5\*8 + 1\*3\*8 + 1\*8\*1 = 167  **Example 2:**  Input: nums = [1,5] Output: 10  **Constraints:**  n == nums.length 1 <= n <= 300  0 <= nums[i] <= 100 |
| 263. | Given an integer array nums and two integers lower and upper, return the number of range sums that lie in [lower, upper] inclusive.  Range sum S(i, j) is defined as the sum of the elements in nums between indices i and j inclusive, where i <= j.  **Example 1:** |

|  |  |
| --- | --- |
|  | Input: nums = [-2,5,-1], lower = -2, upper = 2 Output: 3  Explanation: The three ranges are: [0,0], [2,2], and [0,2] and their respective sums are: -2, -1, 2.  **Example 2:**  Input: nums = [0], lower = 0, upper = 0 Output: 1  **Constraints:**  1 <= nums.length <= 10^5  -2^31 <= nums[i] <= 2^31 - 1  -10^5 <= lower <= upper <= 10^5  The answer is guaranteed to fit in a 32-bit integer. |
| 264. | Given a sorted integer array nums and an integer n, add/patch elements to the array such that any number in the range [1, n] inclusive can be formed by the sum of some elements in the array.  Return the minimum number of patches required.  **Example 1:**  Input: nums = [1,3], n = 6 Output: 1  Explanation:  Combinations of nums are [1], [3], [1,3], which form possible sums of: 1, 3,  4.  Now if we add/patch 2 to nums, the combinations are: [1], [2], [3], [1,3],  [2,3], [1,2,3].  Possible sums are 1, 2, 3, 4, 5, 6, which now covers the range [1, 6]. So we only need 1 patch.  **Example 2:**  Input: nums = [1,5,10], n = 2 0Output: 2  Explanation: The two patches can be [2, 4].  **Constraints:**  1 <= nums.length <= 1000 1 <= nums[i] <= 10^4  nums is sorted in ascending order. 1 <= n <= 2^31 - 1 |
| 265. | You are given a 2D array of integers envelopes where envelopes[i] = [wi, hi] represents the width and the height of an envelope.  One envelope can fit into another if and only if both the width and height of one envelope are greater than the other envelope's width and height.  Return the maximum number of envelopes you can Russian doll (i.e., put one inside the other).  Note: You cannot rotate an envelope. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: envelopes = [[5,4],[6,4],[6,7],[2,3]] Output: 3  Explanation: The maximum number of envelopes you can Russian doll is 3 ([2,3] => [5,4] => [6,7]).  **Example 2:**  Input: envelopes = [[1,1],[1,1],[1,1]] Output: 1  **Constraints:**  1 <= envelopes.length <= 10^5 envelopes[i].length == 2  1 <= wi, hi <= 10^5 |
| 266. | A frog is crossing a river. The river is divided into some number of units, and at each unit, there may or may not exist a stone. The frog can jump on a stone, but it must not jump into the water.  Given a list of stones positions (in units) in sorted ascending order, determine if the frog can cross the river by landing on the last stone. Initially, the frog is on the first stone and assumes the first jump must be 1 unit.  If the frog's last jump was k units, its next jump must be either k - 1, k, or k + 1 units. The frog can only jump in the forward direction.  **Example 1:**  Input: stones = [0,1,3,5,6,8,12,17] Output: true  Explanation: The frog can jump to the last stone by jumping 1 unit to the 2nd stone, then 2 units to the 3rd stone, then 2 units to the 4th stone, then 3 units to the 6th stone, 4 units to the 7th stone, and 5 units to the 8th stone.  **Example 2:**  Input: stones = [0,1,2,3,4,8,9,11] Output: false  Explanation: There is no way to jump to the last stone as the gap between the 5th and 6th stone is too large.  **Constraints:**  2 <= stones.length <= 2000 0 <= stones[i] <= 2^31 - 1 stones[0] == 0  stones is sorted in a strictly increasing order. |
| 267. | Given an m x n integer matrix heightMap representing the height of each  unit cell in a 2D elevation map, return the volume of water it can trap after raining. |

|  |  |
| --- | --- |
|  | **Example 1:**    Input: heightMap = [[1,4,3,1,3,2],[3,2,1,3,2,4],[2,3,3,2,3,1]]  Output: 4  Explanation: After the rain, water is trapped between the blocks. We have two small ponds 1 and 3 units trapped.  The total volume of water trapped is 4.  **Example 2:**    Input: heightMap = [[3,3,3,3,3],[3,2,2,2,3],[3,2,1,2,3],[3,2,2,2,3],[3,3,3,3,3]]  Output: 10  **Constraints:**  m == heightMap.length n == heightMap[i].length 1 <= m, n <= 200  0 <= heightMap[i][j] <= 2 \* 10^4 |
| 268. | A password is considered strong if the below conditions are all met: It has at least 6 characters and at most 20 characters.  It contains at least one lowercase letter, at least one uppercase letter, and at least one digit.  It does not contain three repeating characters in a row (i.e., "Baaabb0" is weak, but "Baaba0" is strong).  Given a string password, return the minimum number of steps required to  make password strong. if password is already strong, return 0. |

|  |  |
| --- | --- |
|  | In one step, you can:  Insert one character to password, Delete one character from password, or  Replace one character of password with another character.  **Example 1:**  Input: password = "a" Output: 5  **Example 2:**  Input: password = "aA1" Output: 3  **Example 3:**  Input: password = "1337C0d3" Output: 0  **Constraints:**  1 <= password.length <= 50  password consists of letters, digits, dot '.' or exclamation mark '!'. |
| 269. | There are buckets buckets of liquid, where exactly one of the buckets is poisonous. To figure out which one is poisonous, you feed some number of (poor) pigs the liquid to see whether they will die or not. Unfortunately, you only have minutesToTest minutes to determine which bucket is poisonous. You can feed the pigs according to these steps:  Choose some live pigs to feed.  For each pig, choose which buckets to feed it. The pig will consume all the chosen buckets simultaneously and will take no time. Each pig can feed from any number of buckets, and each bucket can be fed from by any number of pigs.  Wait for minutesToDie minutes. You may not feed any other pigs during this time.  After minutesToDie minutes have passed, any pigs that have been fed the poisonous bucket will die, and all others will survive.  Repeat this process until you run out of time.  Given buckets, minutesToDie, and minutesToTest, return the minimum number of pigs needed to figure out which bucket is poisonous within the allotted time.  **Example 1:**  Input: buckets = 4, minutesToDie = 15, minutesToTest = 15 Output: 2  Explanation: We can determine the poisonous bucket as follows:  At time 0, feed the first pig buckets 1 and 2, and feed the second pig buckets 2 and 3.  At time 15, there are 4 possible outcomes:   * If only the first pig dies, then bucket 1 must be poisonous. * If only the second pig dies, then bucket 3 must be poisonous. |

|  |  |
| --- | --- |
|  | * If both pigs die, then bucket 2 must be poisonous. * If neither pig dies, then bucket 4 must be poisonous.   **Constraints:**  1 <= buckets <= 1000  1 <= minutesToDie <= minutesToTest <= 100 |
| 270. | Given an integer array nums, return the number of all the arithmetic subsequences of nums.  A sequence of numbers is called arithmetic if it consists of at least three elements and if the difference between any two consecutive elements is the same.  For example, [1, 3, 5, 7, 9], [7, 7, 7, 7], and [3, -1, -5, -9] are arithmetic sequences.  For example, [1, 1, 2, 5, 7] is not an arithmetic sequence.  A subsequence of an array is a sequence that can be formed by removing some elements (possibly none) of the array.  For example, [2,5,10] is a subsequence of [1,2,1,2,4,1,5,10].  The test cases are generated so that the answer fits in 32-bit integer.  **Example 1:**  Input: nums = [2,4,6,8,10] Output: 7  Explanation: All arithmetic subsequence slices are:  [2,4,6]  [4,6,8]  [6,8,10]  [2,4,6,8]  [4,6,8,10]  [2,4,6,8,10]  [2,6,10]  Example 2:  Input: nums = [7,7,7,7,7]Output: 16Explanation: Any subsequence of this array is arithmetic.  **Constraints:**  1 <= nums.length <= 1000  -2^31 <= nums[i] <= 2^31 - 1 |
| 271. | Given two integers n and k, return the kth lexicographically smallest integer in the range [1, n].  **Example 1:**  Input: n = 13, k = 2 Output: 10  Explanation: The lexicographical order is [1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9], so the second smallest number is 10. |

|  |  |
| --- | --- |
|  | **Example 2:** Input: n = 1, k = 1 Output: 1  **Constraints:**  1 <= k <= n <= 10^9 |
| 272. | We define str = [s, n] as the string str which consists of the string s concatenated n times.  For example, str == ["abc", 3] =="abcabcabc".  We define that string s1 can be obtained from string s2 if we can remove some characters from s2 such that it becomes s1.  For example, s1 = "abc" can be obtained from s2 = "abdbec" based on our definition by removing the bolded underlined characters.  You are given two strings s1 and s2 and two integers n1 and n2. You have the two strings str1 = [s1, n1] and str2 = [s2, n2].  Return the maximum integer m such that str = [str2, m] can be obtained from str1.  **Example 1:**  Input: s1 = "acb", n1 = 4, s2 = "ab", n2 = 2 Output: 2  **Example 2:**  Input: s1 = "acb", n1 = 1, s2 = "acb", n2 = 1 Output: 1  **Constraints:**  1 <= s1.length, s2.length <= 100  s1 and s2 consist of lowercase English letters. 1 <= n1, n2 <= 10^6 |
| 273. | The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle values.  For examples, if arr = [2,3,4], the median is 3.  For examples, if arr = [1,2,3,4], the median is (2 + 3) / 2 = 2.5.  You are given an integer array nums and an integer k. There is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.  Return the median array for each window in the original array. Answers within 10-5 of the actual value will be accepted.  **Example 1:**  Input: nums = [1,3,-1,-3,5,3,6,7], k = 3  Output: [1.00000,-1.00000,-1.00000,3.00000,5.00000,6.00000]  Explanation: |

|  |  |
| --- | --- |
|  | Window position Median    [1 3 -1] -3 5 3 6 7 1  1 [3 -1 -3] 5 3 6 7 -1  1 3 [-1 -3 5] 3 6 7 -1  1 3 -1 [-3 5 3] 6 7 3  1 3 -1 -3 [5 3 6] 7 5  1 3 -1 -3 5 [3 6 7] 6  **Example 2:**  Input: nums = [1,2,3,4,2,3,1,4,2], k = 3  Output: [2.00000,3.00000,3.00000,3.00000,2.00000,3.00000,2.00000]  **Constraints:**  1 <= k <= nums.length <= 10^5  -2^31 <= nums[i] <= 2^31 - 1 |
| 274. | Suppose CrackCode will start its IPO soon. In order to sell a good price of its shares to Venture Capital, CrackCode would like to work on some projects to increase its capital before the IPO. Since it has limited resources, it can only finish at most k distinct projects before the IPO. Help CrackCode design the best way to maximize its total capital after finishing at most k distinct projects.  You are given n projects where the ith project has a pure profit profits[i] and a minimum capital of capital[i] is needed to start it.  Initially, you have w capital. When you finish a project, you will obtain its pure profit and the profit will be added to your total capital.  Pick a list of at most k distinct projects from given projects to maximize your final capital, and return the final maximized capital.  The answer is guaranteed to fit in a 32-bit signed integer.  **Example 1:**  Input: k = 2, w = 0, profits = [1,2,3], capital = [0,1,1]  Output: 4  Explanation: Since your initial capital is 0, you can only start the project indexed 0.  After finishing it you will obtain profit 1 and your capital becomes 1.  With capital 1, you can either start the project indexed 1 or the project indexed 2.  Since you can choose at most 2 projects, you need to finish the project indexed 2 to get the maximum capital.  Therefore, output the final maximized capital, which is 0 + 1 + 3 = 4.  **Example 2:**  Input: k = 3, w = 0, profits = [1,2,3], capital = [0,1,2]  Output: 6  **Constraints:**  1 <= k <= 10^5 |

|  |  |
| --- | --- |
|  | 0 <= w <= 10^9  n == profits.length n == capital.length 1 <= n <= 10^5  0 <= profits[i] <= 10^4 0 <= capital[i] <= 10^9 |
| 275. | In the video game Fallout 4, the quest "Road to Freedom" requires players to reach a metal dial called the "Freedom Trail Ring" and use the dial to spell a specific keyword to open the door.  Given a string ring that represents the code engraved on the outer ring and another string key that represents the keyword that needs to be spelled, return the minimum number of steps to spell all the characters in the keyword.  Initially, the first character of the ring is aligned at the "12:00" direction. You should spell all the characters in key one by one by rotating ring clockwise or anticlockwise to make each character of the string key aligned at the "12:00" direction and then by pressing the center button.  At the stage of rotating the ring to spell the key character key[i]:  You can rotate the ring clockwise or anticlockwise by one place, which counts as one step. The final purpose of the rotation is to align one of ring's characters at the "12:00" direction, where this character must equal key[i]. If the character key[i] has been aligned at the "12:00" direction, press the center button to spell, which also counts as one step. After the pressing, you could begin to spell the next character in the key (next stage). Otherwise, you have finished all the spelling.  **Example 1:**    Input: ring = "godding", key = "gd" Output: 4  Explanation:  For the first key character 'g', since it is already in place, we just need 1 step to spell this character.  For the second key character 'd', we need to rotate the ring "godding" anticlockwise by two steps to make it become "ddinggo".  Also, we need 1 more step for spelling. So the final output is 4. |

|  |  |
| --- | --- |
|  | **Example 2:**  Input: ring = "godding", key = "godding" Output: 13  **Constraints:**  1 <= ring.length, key.length <= 100  ring and key consist of only lower case English letters.  It is guaranteed that key could always be spelled by rotating ring. |
| 276. | You are given several boxes with different colors represented by different positive numbers.  You may experience several rounds to remove boxes until there is no box left. Each time you can choose some continuous boxes with the same color (i.e., composed of k boxes, k >= 1), remove them and get k \* k points.  Return the maximum points you can get.  **Example 1:**  Input: boxes = [1,3,2,2,2,3,4,3,1]  Output: 23 Explanation:  [1, 3, 2, 2, 2, 3, 4, 3, 1]  ----> [1, 3, 3, 4, 3, 1] (3\*3=9 points)  ----> [1, 3, 3, 3, 1] (1\*1=1 points)  ----> [1, 1] (3\*3=9 points)  ----> [] (2\*2=4 points)  **Example 2:**  Input: boxes = [1,1,1] Output: 9  Example 3:  Input: boxes = [1] Output: 1  **Constraints:**  1 <= boxes.length <= 100  1 <= boxes[i] <= 100 |
| 277. | An attendance record for a student can be represented as a string where each character signifies whether the student was absent, late, or present on that day. The record only contains the following three characters:  'A': Absent.  'L': Late.  'P': Present.  Any student is eligible for an attendance award if they meet both of the following criteria:  The student was absent ('A') for strictly fewer than 2 days total. The student was never late ('L') for 3 or more consecutive days.  Given an integer n, return the number of possible attendance records of |

|  |  |
| --- | --- |
|  | length n that make a student eligible for an attendance award. The answer may be very large, so return it modulo 109 + 7.  **Example 1:**  Input: n = 2 Output: 8  Explanation: There are 8 records with length 2 that are eligible for an award: "PP", "AP", "PA", "LP", "PL", "AL", "LA", "LL"  Only "AA" is not eligible because there are 2 absences (there need to be fewer than 2).  **Example 2:** Input: n = 1 Output: 3  **Example 3:**  Input: n = 10101 Output: 183236316  **Constraints:**  1 <= n <= 10^5 |
| 278. | You are given an array trees where trees[i] = [xi, yi] represents the location of a tree in the garden.  Fence the entire garden using the minimum length of rope, as it is expensive. The garden is well-fenced only if all the trees are enclosed.  Return the coordinates of trees that are exactly located on the fence perimeter. You may return the answer in any order.  **Example 1:**    Input: trees = [[1,1],[2,2],[2,0],[2,4],[3,3],[4,2]]  Output: [[1,1],[2,0],[4,2],[3,3],[2,4]]  Explanation: All the trees will be on the perimeter of the fence except the tree at [2, 2], which will be inside the fence.  **Example 2:** |

|  |  |
| --- | --- |
|  | Input: trees = [[1,2],[2,2],[4,2]]  Output: [[4,2],[2,2],[1,2]]  Explanation: The fence forms a line that passes through all the trees.  **Constraints:**  1 <= trees.length <= 3000  trees[i].length == 2  0 <= xi, yi <= 100  All the given positions are unique. |
| 279. | For an integer array nums, an inverse pair is a pair of integers [i, j] where 0  <= i < j < nums.length and nums[i] > nums[j].  Given two integers n and k, return the number of different arrays consist of numbers from 1 to n such that there are exactly k inverse pairs. Since the answer can be huge, return it modulo 109 + 7.  **Example 1:**  Input: n = 3, k = 0 Output: 1  Explanation: Only the array [1,2,3] which consists of numbers from 1 to 3 has exactly 0 inverse pairs.  **Example 2:**  Input: n = 3, k = 1 Output: 2  Explanation: The array [1,3,2] and [2,1,3] have exactly 1 inverse pair.  **Constraints:**  1 <= n <= 1000  0 <= k <= 1000 |
| 280. | There are n different online courses numbered from 1 to n. You are given an array courses where courses[i] = [durationi, lastDayi] indicate that the ith course should be taken continuously for durationi days and must be finished before or on lastDayi.  You will start on the 1st day and you cannot take two or more courses simultaneously.  Return the maximum number of courses that you can take. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: courses = [[100,200],[200,1300],[1000,1250],[2000,3200]]  Output: 3 Explanation:  There are totally 4 courses, but you can take 3 courses at most:  First, take the 1st course, it costs 100 days so you will finish it on the 100th day, and ready to take the next course on the 101st day.  Second, take the 3rd course, it costs 1000 days so you will finish it on the 1100th day, and ready to take the next course on the 1101st day.  Third, take the 2nd course, it costs 200 days so you will finish it on the 1300th day.  The 4th course cannot be taken now, since you will finish it on the 3300th day, which exceeds the closed date.  **Example 2:**  Input: courses = [[1,2]] Output: 1  **Example 3:**  Input: courses = [[3,2],[4,3]] Output: 0  **Constraints:**  1 <= courses.length <= 10^4  1 <= durationi, lastDayi <= 10^4 |
| 281. | A message containing letters from A-Z can be encoded into numbers using the following mapping:  'A' -> "1"  'B' -> "2"  ...  'Z' -> "26"  To decode an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, "11106" can be mapped into:  "AAJF" with the grouping (1 1 10 6)  "KJF" with the grouping (11 10 6)  Note that the grouping (1 11 06) is invalid because "06" cannot be mapped into 'F' since "6" is different from "06".  In addition to the mapping above, an encoded message may contain the '\*' character, which can represent any digit from '1' to '9' ('0' is excluded). For example, the encoded message "1\*" may represent any of the encoded messages "11", "12", "13", "14", "15", "16", "17", "18", or "19". Decoding "1\*" is equivalent to decoding any of the encoded messages it can represent.  Given a string s consisting of digits and '\*' characters, return the number of ways to decode it.  Since the answer may be very large, return it modulo 109 + 7. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: s = "\*" Output: 9  Explanation: The encoded message can represent any of the encoded messages "1", "2", "3", "4", "5", "6", "7", "8", or "9".  Each of these can be decoded to the strings "A", "B", "C", "D", "E", "F", "G", "H", and "I" respectively.  Hence, there are a total of 9 ways to decode "\*".  **Example 2:**  Input: s = "1\*" Output: 18  Explanation: The encoded message can represent any of the encoded messages "11", "12", "13", "14", "15", "16", "17", "18", or "19".  Each of these encoded messages have 2 ways to be decoded (e.g. "11" can be decoded to "AA" or "K").  Hence, there are a total of 9 \* 2 = 18 ways to decode "1\*".  **Example 3:**  Input: s = "2\*" Output: 15  Explanation: The encoded message can represent any of the encoded messages "21", "22", "23", "24", "25", "26", "27", "28", or "29".  "21", "22", "23", "24", "25", and "26" have 2 ways of being decoded, but  "27", "28", and "29" only have 1 way.  Hence, there are a total of (6 \* 2) + (3 \* 1) = 12 + 3 = 15 ways to decode "2\*".  **Constraints:**  1 <= s.length <= 10^5 s[i] is a digit or '\*'. |
| 282. | You are given an integer array cards of length 4. You have four cards, each containing a number in the range [1, 9]. You should arrange the numbers on these cards in a mathematical expression using the operators ['+', '-', '\*', '/'] and the parentheses '(' and ')' to get the value 24.  You are restricted with the following rules:  The division operator '/' represents real division, not integer division. For example, 4 / (1 - 2 / 3) = 4 / (1 / 3) = 12.  Every operation done is between two numbers. In particular, we cannot use '-' as a unary operator.  For example, if cards = [1, 1, 1, 1], the expression "-1 - 1 - 1 - 1" is not allowed.  You cannot concatenate numbers together  For example, if cards = [1, 2, 1, 2], the expression "12 + 12" is not valid. Return true if you can get such expression that evaluates to 24, and false otherwise. |

|  |  |
| --- | --- |
|  | **Example 1:**  Input: cards = [4,1,8,7] Output: true  Explanation: (8-4) \* (7-1) = 24  **Example 2:**  Input: cards = [1,2,1,2] Output: false  **Constraints:**  cards.length == 4  1 <= cards[i] <= 9 |
| 283. | There are several squares being dropped onto the X-axis of a 2D plane.  You are given a 2D integer array positions where positions[i] = [lefti, sideLengthi] represents the ith square with a side length of sideLengthi that is dropped with its left edge aligned with X-coordinate lefti.  Each square is dropped one at a time from a height above any landed squares. It then falls downward (negative Y direction) until it either lands on the top side of another square or on the X-axis. A square brushing the left/right side of another square does not count as landing on it. Once it lands, it freezes in place and cannot be moved.  After each square is dropped, you must record the height of the current tallest stack of squares.  Return an integer array ans where ans[i] represents the height described above after dropping the ith square.  **Example 1:**    Input: positions = [[1,2],[2,3],[6,1]]Output: [2,5,5]Explanation: After the first drop, the tallest stack is square 1 with a height of 2.  After the second drop, the tallest stack is squares 1 and 2 with a height of 5. |

|  |  |
| --- | --- |
|  | After the third drop, the tallest stack is still squares 1 and 2 with a height of 5.  Thus, we return an answer of [2, 5, 5].  Example 2:  Input: positions = [[100,100],[200,100]]Output: [100,100]Explanation: After the first drop, the tallest stack is square 1 with a height of 100.  After the second drop, the tallest stack is either square 1 or square 2, both with heights of 100.  Thus, we return an answer of [100, 100].  Note that square 2 only brushes the right side of square 1, which does not count as landing on it.  **Constraints:**  1 <= positions.length <= 1000  1 <= lefti <= 108  1 <= sideLengthi <= 10^6 |
| 284. | Given a string formula representing a chemical formula, return the count of each atom.  The atomic element always starts with an uppercase character, then zero or more lowercase letters, representing the name.  One or more digits representing that element's count may follow if the count is greater than 1. If the count is 1, no digits will follow.  For example, "H2O" and "H2O2" are possible, but "H1O2" is impossible. Two formulas are concatenated together to produce another formula. For example, "H2O2He3Mg4" is also a formula.  A formula placed in parentheses, and a count (optionally added) is also a formula.  For example, "(H2O2)" and "(H2O2)3" are formulas.  Return the count of all elements as a string in the following form: the first name (in sorted order), followed by its count (if that count is more than 1), followed by the second name (in sorted order), followed by its count (if that count is more than 1), and so on.  The test cases are generated so that all the values in the output fit in a 32- bit integer.  **Example 1:**  Input: formula = "H2O" Output: "H2O"  Explanation: The count of elements are {'H': 2, 'O': 1}.  **Example 2:**  Input: formula = "Mg(OH)2" Output: "H2MgO2"  Explanation: The count of elements are {'H': 2, 'Mg': 1, 'O': 2}.  **Example 3:**  Input: formula = "K4(ON(SO3)2)2" Output: "K4N2O14S4" |

|  |  |
| --- | --- |
|  | Explanation: The count of elements are {'K': 4, 'N': 2, 'O': 14, 'S': 4}.  **Constraints:**  1 <= formula.length <= 1000  formula consists of English letters, digits, '(', and ')'. formula is always valid. |
| 285. | There is a safe protected by a password. The password is a sequence |
|  | of n digits where each digit can be in the range [0, k - 1]. |
|  | The safe has a peculiar way of checking the password. When you enter in a |
|  | sequence, it checks the most recent n digits that were entered each time |
|  | you type a digit. |
|  | For example, the correct password is "345" and you enter in "012345": |
|  | After typing 0, the most recent 3 digits is "0", which is incorrect. |
|  | After typing 1, the most recent 3 digits is "01", which is incorrect. |
|  | After typing 2, the most recent 3 digits is "012", which is incorrect. |
|  | After typing 3, the most recent 3 digits is "123", which is incorrect. |
|  | After typing 4, the most recent 3 digits is "234", which is incorrect. |
|  | After typing 5, the most recent 3 digits is "345", which is correct and the |
|  | safe unlocks. |
|  | Return any string of minimum length that will unlock the safe at some |
|  | point of entering it. |
|  | **Example 1:** |
|  | Input: n = 1, k = 2 |
|  | Output: "10" |
|  | Explanation: The password is a single digit, so enter each digit. "01" would |
|  | also unlock the safe. |
|  | **Example 2:** |
|  | Input: n = 2, k = 2 |
|  | Output: "01100" |
|  | Explanation: For each possible password: |
|  | - "00" is typed in starting from the 4th digit. |
|  | - "01" is typed in starting from the 1st digit. |
|  | - "10" is typed in starting from the 3rd digit. |
|  | - "11" is typed in starting from the 2nd digit. |
|  | Thus "01100" will unlock the safe. "10011", and "11001" would also unlock |
|  | the safe. |
|  | **Constraints:** |
|  | 1 <= n <= 4 |
|  | 1 <= k <= 10 |
|  | 1 <= kn <= 4096 |
| 286. | There are n couples sitting in 2n seats arranged in a row and want to hold hands.  The people and seats are represented by an integer |

|  |  |
| --- | --- |
|  | array row where row[i] is the ID of the person sitting in the ith seat. The couples are numbered in order, the first couple being (0, 1), the second couple being (2, 3), and so on with the last couple being (2n - 2, 2n - 1).  Return the minimum number of swaps so that every couple is sitting side by side. A swap consists of choosing any two people, then they stand up and switch seats.  **Example 1:**  Input: row = [0,2,1,3] Output: 1  Explanation: We only need to swap the second (row[1]) and third (row[2]) person.  **Example 2**:  Input: row = [3,2,0,1] Output: 0  Explanation: All couples are already seated side by side.  **Constraints:**  2n == row.length 2 <= n <= 30  n is even.  0 <= row[i] < 2n  All the elements of row are unique. |
| 287. | On an 2 x 3 board, there are five tiles labeled from 1 to 5, and an empty square represented by 0. A move consists of choosing 0 and a 4-directionally adjacent number and swapping it.  The state of the board is solved if and only if the board is [[1,2,3],[4,5,0]]. Given the puzzle board board, return the least number of moves required so that the state of the board is solved. If it is impossible for the state of the board to be solved, return -1.  **Example 1:**    Input: board = [[1,2,3],[4,0,5]] Output: 1  Explanation: Swap the 0 and the 5 in one move.  **Example 2:** |

|  |  |
| --- | --- |
|  | Input: board = [[1,2,3],[5,4,0]] Output: -1  Explanation: No number of moves will make the board solved.  **Example 3:**    Input: board = [[4,1,2],[5,0,3]] Output: 5  Explanation: 5 is the smallest number of moves that solves the board. An example path:  After move 0: [[4,1,2],[5,0,3]]  After move 1: [[4,1,2],[0,5,3]]  After move 2: [[0,1,2],[4,5,3]]  After move 3: [[1,0,2],[4,5,3]]  After move 4: [[1,2,0],[4,5,3]]  After move 5: [[1,2,3],[4,5,0]]  **Constraints:**  board.length == 2  board[i].length == 3  0 <= board[i][j] <= 5  Each value board[i][j] is unique. |
| 288. | You are given an integer array nums.  You should move each element of nums into one of the two arrays A and B such that A and B are non-empty, and average(A) == average(B).  Return true if it is possible to achieve that and false otherwise.  Note that for an array arr, average(arr) is the sum of all the elements of arr over the length of arr.  **Example 1:**  Input: nums = [1,2,3,4,5,6,7,8]  Output: true  Explanation: We can split the array into [1,4,5,8] and [2,3,6,7], and both of them have an average of 4.5.  **Example 2:**  Input: nums = [3,1] Output: false |

|  |  |
| --- | --- |
|  | **Constraints:**  1 <= nums.length <= 30 0 <= nums[i] <= 10^4 |
| 289. | Your car starts at position 0 and speed +1 on an infinite number line. Your car can go into negative positions. Your car drives automatically according to a sequence of instructions 'A' (accelerate) and 'R' (reverse):  When you get an instruction 'A', your car does the following: position += speed  speed \*= 2  When you get an instruction 'R', your car does the following:  If your speed is positive then speed = -1 otherwise speed = 1  Your position stays the same.  For example, after commands "AAR", your car goes to positions 0 --> 1 --> 3  --> 3, and your speed goes to 1 --> 2 --> 4 --> -1.  Given a target position target, return the length of the shortest sequence of instructions to get there.  **Example 1:**  Input: target = 3 Output: 2 Explanation:  The shortest instruction sequence is "AA". Your position goes from 0 --> 1 --> 3.  **Example 2:**  Input: target = 6 Output: 5 Explanation:  The shortest instruction sequence is "AAARA".  Your position goes from 0 --> 1 --> 3 --> 7 --> 7 --> 6.  **Constraints:**  1 <= target <= 10^4 |
| 290. | You are given an array routes representing bus routes where routes[i] is a bus route that the ith bus repeats forever.  For example, if routes[0] = [1, 5, 7], this means that the 0th bus travels in the sequence 1 -> 5 -> 7 -> 1 -> 5 -> 7 -> 1 -> ... forever.  You will start at the bus stop source (You are not on any bus initially), and you want to go to the bus stop target. You can travel between bus stops by buses only.  Return the least number of buses you must take to travel from source to target. Return -1 if it is not possible.  **Example 1:**  Input: routes = [[1,2,7],[3,6,7]], source = 1, target = 6 Output: 2 |

|  |  |
| --- | --- |
|  | Explanation: The best strategy is take the first bus to the bus stop 7, then take the second bus to the bus stop 6.  **Example 2:**  Input: routes = [[7,12],[4,5,15],[6],[15,19],[9,12,13]], source = 15, target = 12 Output: -1  **Constraints:**  1 <= routes.length <= 500.  1 <= routes[i].length <= 10^5  All the values of routes[i] are unique. sum(routes[i].length) <= 10^5  0 <= routes[i][j] < 10^6  0 <= source, target < 10^6 |
| 291. | There are n workers. You are given two integer arrays quality and wage where quality[i] is the quality of the ith worker and wage[i] is the minimum wage expectation for the ith worker.  We want to hire exactly k workers to form a paid group. To hire a group of k workers, we must pay them according to the following rules:  Every worker in the paid group should be paid in the ratio of their quality compared to other workers in the paid group.  Every worker in the paid group must be paid at least their minimum wage expectation.  Given the integer k, return the least amount of money needed to form a paid group satisfying the above conditions. Answers within 10-5 of the actual answer will be accepted.  **Example 1:**  Input: quality = [10,20,5], wage = [70,50,30], k = 2 Output: 105.00000  Explanation: We pay 70 to 0th worker and 35 to 2nd worker.  **Example 2:**  Input: quality = [3,1,10,10,1], wage = [4,8,2,2,7], k = 3 Output: 30.66667  Explanation: We pay 4 to 0th worker, 13.33333 to 2nd and 3rd workers separately.  **Constraints:**  n == quality.length == wage.length 1 <= k <= n <= 10^4  1 <= quality[i], wage[i] <= 10^4 |
| 292. | You are given k identical eggs and you have access to a building with n floors labeled from 1 to n.  You know that there exists a floor f where 0 <= f <= n such that any egg  dropped at a floor higher than f will break, and any egg dropped at or below floor f will not break. |

|  |  |
| --- | --- |
|  | Each move, you may take an unbroken egg and drop it from any floor x (where 1 <= x <= n). If the egg breaks, you can no longer use it. However, if the egg does not break, you may reuse it in future moves.  Return the minimum number of moves that you need to determine with certainty what the value of f is.  **Example 1:**  Input: k = 1, n = 2 Output: 2 Explanation:  Drop the egg from floor 1. If it breaks, we know that f = 0. Otherwise, drop the egg from floor 2. If it breaks, we know that f = 1. If it does not break, then we know f = 2.  Hence, we need at minimum 2 moves to determine with certainty what the value of f is.  **Example 2:** Input: k = 2, n = 6 Output: 3  **Example 3:**  Input: k = 3, n = 14 Output: 4  **Constraints:**  1 <= k <= 100  1 <= n <= 10^4 |
| 293. | A game on an undirected graph is played by two players, Mouse and Cat, who alternate turns.  The graph is given as follows: graph[a] is a list of all nodes b such that ab is an edge of the graph.  The mouse starts at node 1 and goes first, the cat starts at node 2 and goes second, and there is a hole at node 0.  During each player's turn, they must travel along one edge of the graph that meets where they are. For example, if the Mouse is at node 1, it must travel to any node in graph[1].  Additionally, it is not allowed for the Cat to travel to the Hole (node 0.) Then, the game can end in three ways:  If ever the Cat occupies the same node as the Mouse, the Cat wins. If ever the Mouse reaches the Hole, the Mouse wins.  If ever a position is repeated (i.e., the players are in the same position as a previous turn, and it is the same player's turn to move), the game is a draw. Given a graph, and assuming both players play optimally, return  1 if the mouse wins the game, 2 if the cat wins the game, or 0 if the game is a draw. |

|  |  |
| --- | --- |
|  | **Example 1:**    Input: graph = [[2,5],[3],[0,4,5],[1,4,5],[2,3],[0,2,3]]  Output: 0  **Example 2:**    Input: graph = [[1,3],[0],[3],[0,2]]  Output: 1  **Constraints:**  3 <= graph.length <= 50  1 <= graph[i].length < graph.length 0 <= graph[i][j] < graph.length graph[i][j] != i  graph[i] is unique.  The mouse and the cat can always move. |
| 294. | You are given a network of n nodes represented as an n x n adjacency matrix graph, where the ith node is directly connected to the jth node if graph[i][j] == 1.  Some nodes initial are initially infected by malware. Whenever two nodes are directly connected, and at least one of those two nodes is infected by malware, both nodes will be infected by malware. This spread of malware will continue until no more nodes can be infected in this manner.  Suppose M(initial) is the final number of nodes infected with malware in the entire network after the spread of malware stops.  We will remove exactly one node from initial, completely removing it and  any connections from this node to any other node. |

|  |  |
| --- | --- |
|  | Return the node that, if removed, would minimize M(initial). If multiple nodes could be removed to minimize M(initial), return such a node with the smallest index.  **Example 1:**  Input: graph = [[1,1,0],[1,1,0],[0,0,1]], initial = [0,1] Output: 0  **Example 2:**  Input: graph = [[1,1,0],[1,1,1],[0,1,1]], initial = [0,1] Output: 1  **Example 3:**  Input: graph = [[1,1,0,0],[1,1,1,0],[0,1,1,1],[0,0,1,1]], initial = [0,1]  Output: 1  **Constraints:**  n == graph.length  n == graph[i].length 2 <= n <= 300  graph[i][j] is 0 or 1. graph[i][j] == graph[j][i] graph[i][i] == 1  1 <= initial.length < n 0 <= initial[i] <= n - 1  All the integers in initial are unique. |
| 295. | You are given two strings stamp and target. Initially, there is a string s of length target.length with all s[i] == '?'.  In one turn, you can place stamp over s and replace every letter in the s with the corresponding letter from stamp.  For example, if stamp = "abc" and target = "abcba", then s is "?????" initially. In one turn you can:  place stamp at index 0 of s to obtain "abc??", place stamp at index 1 of s to obtain "?abc?", or place stamp at index 2 of s to obtain "??abc".  Note that stamp must be fully contained in the boundaries of s in order to stamp (i.e., you cannot place stamp at index 3 of s).  We want to convert s to target using at most 10 \* target.length turns. Return an array of the index of the left-most letter being stamped at each turn. If we cannot obtain target from s within 10 \* target.length turns, return an empty array.  **Example 1:**  Input: stamp = "abc", target = "ababc" Output: [0,2]  Explanation: Initially s = "?????".   * Place stamp at index 0 to get "abc??". * Place stamp at index 2 to get "ababc". |

|  |  |
| --- | --- |
|  | [1,0,2] would also be accepted as an answer, as well as some other answers.  **Example 2:**  Input: stamp = "abca", target = "aabcaca" Output: [3,0,1]  Explanation: Initially s = "???????".   * Place stamp at index 3 to get "???abca". * Place stamp at index 0 to get "abcabca". * Place stamp at index 1 to get "aabcaca".   **Constraints:**  1 <= stamp.length <= target.length <= 1000  stamp and target consist of lowercase English letters. |
| 296. | You are installing a billboard and want it to have the largest height. The billboard will have two steel supports, one on each side. Each steel support must be an equal height.  You are given a collection of rods that can be welded together. For example, if you have rods of lengths 1, 2, and 3, you can weld them together to make a support of length 6.  Return the largest possible height of your billboard installation. If you cannot support the billboard, return 0.  **Example 1:**  Input: rods = [1,2,3,6] Output: 6  Explanation: We have two disjoint subsets {1,2,3} and {6}, which have the same sum = 6.  **Example 2:**  Input: rods = [1,2,3,4,5,6] Output: 10  Explanation: We have two disjoint subsets {2,3,5} and {4,6}, which have the same sum = 10.  **Example 3:**  Input: rods = [1,2] Output: 0  Explanation: The billboard cannot be supported, so we return 0.  **Constraints:**  1 <= rods.length <= 20  1 <= rods[i] <= 1000  sum(rods[i]) <= 5000 |
| 297. | You are given an integer array arr. From some starting index, you can make a series of jumps. The (1st, 3rd, 5th, ...) jumps in the series are called odd- numbered jumps, and the (2nd, 4th, 6th, ...) jumps in the series are called even-numbered jumps. Note that the jumps are numbered, not the  indices. |

|  |  |
| --- | --- |
|  | You may jump forward from index i to index j (with i < j) in the following way:  During odd-numbered jumps (i.e., jumps 1, 3, 5, ...), you jump to the index j such that arr[i] <= arr[j] and arr[j] is the smallest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.  During even-numbered jumps (i.e., jumps 2, 4, 6, ...), you jump to the index j such that arr[i] >= arr[j] and arr[j] is the largest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.  It may be the case that for some index i, there are no legal jumps.  A starting index is good if, starting from that index, you can reach the end of the array (index arr.length - 1) by jumping some number of times (possibly 0 or more than once).  Return the number of good starting indices.  **Example 1:**  Input: arr = [10,13,12,14,15]  Output: 2 Explanation:  From starting index i = 0, we can make our 1st jump to i = 2 (since arr[2] is the smallest among arr[1], arr[2], arr[3], arr[4] that is greater or equal to arr[0]), then we cannot jump any more.  From starting index i = 1 and i = 2, we can make our 1st jump to i = 3, then we cannot jump any more.  From starting index i = 3, we can make our 1st jump to i = 4, so we have reached the end.  From starting index i = 4, we have reached the end already.  In total, there are 2 different starting indices i = 3 and i = 4, where we can reach the end with some number of  jumps.  **Constraints:**  1 <= arr.length <= 2 \* 10^4 0 <= arr[i] < 10^5 |
| 298. | Under the grammar given below, strings can represent a set of lowercase words. Let R(expr) denote the set of words the expression represents.  The grammar can best be understood through simple examples: Single letters represent a singleton set containing that word.  R("a") = {"a"}  R("w") = {"w"}  When we take a comma-delimited list of two or more expressions, we take the union of possibilities.  R("{a,b,c}") = {"a","b","c"}  R("{{a,b},{b,c}}") = {"a","b","c"} (notice the final set only contains each word at most once) |

|  |  |
| --- | --- |
|  | When we concatenate two expressions, we take the set of possible concatenations between two words where the first word comes from the first expression and the second word comes from the second expression.  R("{a,b}{c,d}") = {"ac","ad","bc","bd"}  R("a{b,c}{d,e}f{g,h}") = {"abdfg", "abdfh", "abefg", "abefh", "acdfg", "acdfh", "acefg", "acefh"}  Formally, the three rules for our grammar:  For every lowercase letter x, we have R(x) = {x}.  For expressions e1, e2, ... , ek with k >= 2, we have R({e1, e2, ...}) = R(e1) ∪  R(e2) ∪ ...  For expressions e1 and e2, we have R(e1 + e2) = {a + b for (a, b) in R(e1) × R(e2)}, where + denotes concatenation, and × denotes the cartesian product.  Given an expression representing a set of words under the given grammar, return the sorted list of words that the expression represents.  **Example 1:**  Input: expression = "{a,b}{c,{d,e}}" Output: ["ac","ad","ae","bc","bd","be"] **Example 2:**  Input: expression = "{{a,z},a{b,c},{ab,z}}"  Output: ["a","ab","ac","z"]  Explanation: Each distinct word is written only once in the final answer.  **Constraints:**  1 <= expression.length <= 60  expression[i] consists of '{', '}', ','or lowercase English letters.  The given expression represents a set of words based on the grammar given in the description. |
| 299. | In an n\*n grid, there is a snake that spans 2 cells and starts moving from the top left corner at (0, 0) and (0, 1). The grid has empty cells represented by zeros and blocked cells represented by ones. The snake wants to reach the lower right corner at (n-1, n-2) and (n-1, n-1).  In one move the snake can:  Move one cell to the right if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.  Move down one cell if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.  Rotate clockwise if it's in a horizontal position and the two cells under it are both empty. In that case the snake moves from (r, c) and (r, c+1) to (r,  c) and (r+1, c). |

|  |  |
| --- | --- |
|  | Rotate counterclockwise if it's in a vertical position and the two cells to its right are both empty. In that case the snake moves from (r, c) and (r+1, c) to (r, c) and (r, c+1).    Return the minimum number of moves to reach the target. If there is no way to reach the target, return -1.  **Example 1:**    Input: grid = [[0,0,0,0,0,1],  [1,1,0,0,1,0],  [0,0,0,0,1,1],  [0,0,1,0,1,0],  [0,1,1,0,0,0],  [0,1,1,0,0,0]]  Output: 11  Explanation:One possible solution is [right, right, rotate clockwise, right, down, down, down, down, rotate counterclockwise, right, down].  **Constraints:**  2 <= n <= 100  0 <= grid[i][j] <= 1  It is guaranteed that the snake starts at empty cells. |
| 300. | Given an array of integers arr and an integer d. In one step you can jump  from index i to index: |

|  |  |
| --- | --- |
|  | i + x where: i + x < arr.length and 0 < x <= d. i - x where: i - x >= 0 and 0 < x <= d.  In addition, you can only jump from index i to index j if arr[i] > arr[j] and arr[i] > arr[k] for all indices k between i and j (More formally min(i, j) < k < max(i, j)).  You can choose any index of the array and start jumping. Return the maximum number of indices you can visit.  Notice that you can not jump outside of the array at any time.  **Example 1:**    Input: arr = [6,4,14,6,8,13,9,7,10,6,12], d = 2  Output: 4  Explanation: You can start at index 10. You can jump 10 --> 8 --> 6 --> 7 as shown.  Note that if you start at index 6 you can only jump to index 7. You cannot jump to index 5 because 13 > 9. You cannot jump to index 4 because index 5 is between index 4 and 6 and 13 > 9.  Similarly You cannot jump from index 3 to index 2 or index 1.  **Example 2:**  Input: arr = [3,3,3,3,3], d = 3 Output: 1  Explanation: You can start at any index. You always cannot jump to any index.  **Example 3:**  Input: arr = [7,6,5,4,3,2,1], d = 1  Output: 7  Explanation: Start at index 0. You can visit all the indicies.  **Constraints:**  1 <= arr.length <= 1000 1 <= arr[i] <= 10^5  1 <= d <= arr.length |