Kth smallest element

Given an array **arr[]** and a number **K** where K is smaller than size of array, the task is to find the **K^{th} smallest** element in the given array. It is given that all array elements are distinct.

Expected Time Complexity: O(n)

Input:
The first line of input contains an integer **T,** denoting the number of testcases. Then T test cases follow. Each test case consists of three lines. First line of each testcase contains an integer **N** denoting size of the array. Second line contains N space separated integer denoting elements of the array. Third line of the test case contains an integer K.

Output:
Corresponding to each test case, print the k^{th} smallest element in a new line.

Constraints:
1 <= T <= 100
1 <= N <= 10^5
1 <= arr[i] <= 10^5
1 <= K <= N

Example:
Input:
2
6
7 10 4 3 20 15
3
5
7 10 4 20 15
4

Output:
7
15

Explanation:
Testcase 1: 3rd smallest element in the given array is 7.

Quickselect
if (k > 0 && k <= r - l + 1)
int p = partition ;
if (p == k-1) return
if (p > k-1) left subarray
if (p < k-1) right sub
k = k - pos + l - 1
left INT_MAX.

Find Transition Point

You are given a sorted array containing only numbers 0 and 1. Find the transition point efficiently. Transition point is a point where "0" ends and "1" begins.

Input:

You have to complete the method which takes 2 argument: the array arr[] and size of array N. You should not read any input from stdin/console. There are multiple test cases. For each test cases, this method will be called individually.

Output:

1. linearly traverse O(N)
2. Sorted, transition

Your function should return transition point.

Constraints:

$1 \le T \le 100$
$1 \le N \le 500000$
$0 \le C[i] \le 1$

Example:

Input
1
5
0 0 0 1 1

Output
3

Square root

Given an integer **x**. The task is to find the square root of x. If **x** is not a perfect square, then return floor(√x).

Input Format:
First line of input contains number of testcases T. For each testcase, the only line contains the number x.

Output Format:
For each testcase, print square root of given integer.

User Task:
The task is to complete the function **floorSqrt**() which should return the square root of given number x.

Constraints:
$1 \le T \le 1000$
$1 \le x \le 10^7$

Example:
Input:
2
5
4

Output:
2
2

Explanation:
Testcase 1: Since, 5 is not perfect square, so floor of square_root of 5 is 2.
Testcase 2: Since, 4 is a perfect square, so its square root is 2.

4 Element appearing once

Given an sorted array **A[i]** of **N** positive integers having all the numbers occuring exactly twice, except for one number which will occur only once. Find the number occuring only once.

Input
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. The first line of each test case contains a positive integer **N**, denoting the size of the array.
The second line of each test case contains a **N** positive integers, denoting the elements of the array.

Output
Print out the singly occuring number.

Constraints
1 <= **T** <= 100
0 < **N** <= 100
0 <= **A[i]** <= 100

Examples

Input
2
5
1 1 2 5 5
7
2 2 5 5 20 30 30

Output
2
20

Expected Complexity

Time : O(N)

5 Index Of an Extra Element

Given two sorted arrays. There is only 1 difference between the arrays. First array has one element extra added in between. Find the index of the extra element.

Input:
The first line of input contains an integer T, denoting the no of test cases. Then T test cases follow. The first line of each test case contains an integer N, denoting the number of elements in array. The second line of each test case contains N space separated values of the array A[].
The Third line of each test case contains N-1 space separated values of the array B[].

Output:
Return the index of the corresponding extra element in array A[].(starting index 0).

Constraints:
1<=T<=100
1<=N<=100
1<=Ai<=1000

Example:
Input:
2
7
2 4 6 8 9 10 12
2 4 6 8 10 12
6
3 5 7 9 11 13
3 5 7 11 13
Output:
4
3

Kth smallest element

Given an array **arr[]** and a number **K** where K is smaller than size of array, the task is to find the **K^{th} smallest** element in the given array. It is given that all array elements are distinct.

Expected Time Complexity: O(n)

Input:
The first line of input contains an integer **T,** denoting the number of testcases. Then T test cases follow. Each test case consists of three lines. First line of each testcase contains an integer **N** denoting size of the array. Second line contains N space separated integer denoting elements of the array. Third line of the test case contains an integer K.

Output:
Corresponding to each test case, print the k^{th} smallest element in a new line.

Constraints:
1 <= T <= 100
1 <= N <= 10^5
1 <= arr[i] <= 10^5
1 <= K <= N

Example:
Input:
2
6
7 10 4 3 20 15
3
5
7 10 4 20 15
4

Output:
7
15

Explanation:
Testcase 1: 3rd smallest element in the given array is 7.

Merge Sort for Linked List

Given Pointer/Reference to the head of the linked list, the task is to **Sort the given linked list using Merge Sort**.

Note: If the length of linked list is odd, then the extra node should go in the first list while splitting.

Input:
First line of input contains number of testcases T. For each testcase, first line of input contains number of nodes in the linked list and next line contains N elements of the linked list.

Output:
For each test, in a new line, print the sorted linked list.

Your Task:
For C++ and Python: The task is to complete the function **mergeSort**() which sort the linked list using merge sort function.
For Java: The task is to complete the function **mergeSort**() and return the node which can be used to print the sorted linked list.

Constraints:
1 <= T <= 100
1 <= N <= 10^5

Example:
Input:
2
5
3 5 2 4 1
3
9 15 0

Ouput:
1 2 3 4 5
0 9 15

Explanation:
Testcase 1: After sorting the given linked list, the resultant matrix will be 1->2->3->4->5.

Union of Two Linked Lists

Given two linked lists, your task is to complete the function **makeUnion(),** that returns the union of two linked lists. This union should include all the distinct elements only.

Input:
The function takes 2 arguments, reference pointer to the head of the first linked list (**head1**) and reference pointer to the head of the second linked list (**head2**).
There are multiple test cases and for each test case this function will be called separately.

Output:
The function should return reference pointer to the head of the new list that is formed by the union of the two the lists.
Note: The new list formed should be in non-decreasing order.

User Task:
The task is to complete the function **makeUnion**() which makes the union of the given two lists.

Constraints:
1<=T<=100
1<=N<=10^3

Example:
Input:
1
6
9 6 4 2 3 8
5
1 2 8 6 2

Output:
1 2 3 4 6 8 9

Explanation:
Testcase 1: Union of the given two lists have elements 1, 2, 3, 4, 6, 8 and 9 in the list.

9

Merge k Sorted Arrays

Given **K** sorted arrays arranged in form of a matrix. The task is to merge them. You need to complete **mergeKArrays**() function which takes 2 arguments, an arr[k][k] 2D Matrix containing k sorted arrays and an integer k denoting the number of sorted arrays. The function should return a pointer to the merged sorted arrays.

Input:
The first line of input contains the number of test cases, then T test cases follows. Each test case will contain an integer N denoting the number of sorted arrays. Then in the next line contains all the elements of the array separated by space.

Output:
The output will be the sorted merged array.

User Task:
The task is to complete the function **mergeKArrays**() which takes two arguments, and returns pointer to the modified array.

Constraints:
1 <= T <= 50

use of heap min

$1 <= N <= 10^3$
$1 <= K <= 10$

Example:
Input:
1
3
1 2 3 4 5 6 7 8 9

Output:
1 2 3 4 5 6 7 8 9

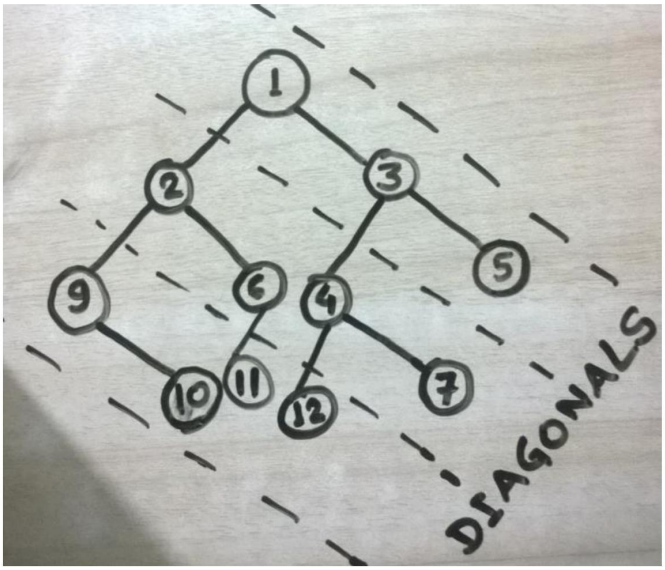
Explanation:
Testcase 1:
Above test case has 3 sorted arrays of size 3, 3, 3
arr[][] = [[1, 2, 3],

[4, 5, 6],

[7, 8, 9]]
The merged list will be [1, 2, 3, 4, 5, 6, 7, 8, 9].

Diagonal Sum In Binary Tree

Consider lines of slope -1 passing between nodes (dotted lines in below diagram). The diagonal sum in a binary tree is the sum of all node's data lying between these lines. Given a Binary Tree, print all diagonal sums.



Note: The **Input/Ouput** format and **Example** given are used for system's internal purpose, and should be used by a user for **Expected Output** only. As it is a function problem, hence a user should not read any input from stdin/console. The task is to complete the function specified, and not to write the full code.

Input:
The first line consists of T test cases. The first line of every test case consists of N, denoting the number of edges in the tree. The second and third line of every test case consists of N, nodes of the binary tree.

Output:
Print space separated integers which are the diagonal sums for every diagonal present in the tree with slope -1.

Constraints:
1<=T<=100
1<=N<=100

Example:
Input:
2
3
4 1 L 4 3 R 3 3 L
5
10 8 L 10 2 R 8 3 L 8 5 R 2 2 L

Output:
7 4
12 15 3

Josephus problem

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.
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.

Input Format:
The first line of input contains an integer **T** denoting the number of test cases . Then T test cases follow. Each test case contains 2 integers **n and k** .

Output Format:
For each test case, in a new line, output will be the safe position which satisfies the above condition.

Your Task:
This is a function problem. You are required to complete the **function josephus** that takes **two parameters n and k** and **returns** an integer denoting **safe position** .

Constraints:
1 <= T <= 100
1 <= k, n <= 20

Example:
Input:
2
3 2
5 3
Output
3
4

Explanation:
Testcase 1: There are 3 persons so skipping 1 person i.e 1st person 2nd person will be killed. Thus the safe position is 3.

Number of paths

The problem is to count all the possible paths from top left to bottom right of a **MxN** matrix with the constraints that from each cell you can either move to **right** or **down**.

Input:
The first line of input contains an integer **T,** denoting the number of test cases. The first line of each test case is **M** and **N**, M is number of rows and N is number of columns.

Output:
For each test case, print the number of paths.

Constraints:
1 ≤ T ≤ 30
1 ≤ M,N ≤ 10

Example:
Input
2
3 3
2 8

Output
6
8

Explanation:
Testcase 1: Let the given input 3*3 matrix is filled as such:
A B C
D E F
G H I
The possible paths which exists to reach 'I' from 'A' following above conditions are as follows:
ABCFI, ABEHI, ADGHI, ADEFI, ADEHI, ABEFI.

Sort a stack

else temp = top
combine (V, S) push(temp)

Given a stack, the task is to sort it such that the top of the stack has the greatest element.

Input:
The first line of input will contains an integer T denoting the no of test cases . Then T test cases follow. Each test case contains an integer N denoting the size of the stack. Then in the next line are N space separated values which are pushed to the the stack.

Output:
For each test case output will be the popped elements from the sorted stack.

Constraints:
1<=T<=100
1<=N<=100

Example(To be used only for expected output):
Input:
2
3
3 2 1
5
11 2 32 3 41

Output:
3 2 1
41 32 11 3 2

Explanation:
For first test case stack will be
1
2
3
After sorting
3
2
1

When elements popped : 3 2 1

Solve ()
temp = top()
Solve (S)
combine (temp, A)

Note: The **Input/Ouput** format and **Example** given are used for system's internal purpose, and should be used by a user for **Expected Output** only. As it is a function problem, hence a user should not read any input from stdin/console. The task is to complete the function specified, and not to write the full code.

(14)

N meetings in one room

(1) Sort a/c to end time

There is **one** meeting room in a firm. There are **N** meetings in the form of (S[i], F[i]) where S[i] is start time of meeting i and F[i] is finish time of meeting i.

What is the *maximum* number of meetings that can be accommodated in the meeting room?

Input:
The first line of input consists number of the test cases. The description of T test cases is as follows:
The first line consists of the size of the array, second line has the array containing the starting time of all the meetings each separated by a space, i.e., S [i]. And the third line has the array containing the finishing time of all the meetings each separated by a space, i.e., F [i].

Output:
In each separate line print the order in which the meetings take place separated by a space.

Constraints:
1 ≤ T ≤ 70
1 ≤ N ≤ 100
1 ≤ S[i], F[i] ≤ 100000

Example:
Input:
2
6
1 3 0 5 8 5
2 4 6 7 9 9
8
75250 50074 43659 8931 11273 27545 50879 77924
112960 114515 81825 93424 54316 35533 73383 160252

Output:
1 2 4 5
6 7 1

Explanation:
Testcase 1: Four meetings can held with given start and end timings.

Max length chain

You are given N pairs of numbers. In every pair, the first number is always smaller than the second number. A pair (c, d) can follow another pair (a, b) if b < c. Chain of pairs can be formed in this fashion. Your task is to complete the function **maxChainLen** which returns an integer denoting the longest chain which can be formed from a given set of pairs.

Input:
The first line of input contains an integer T denoting the no of test cases then T test cases follow .Then T test cases follow . The first line of input contains an integer N denoting the no of pairs . In the next line are 2*N space separated values denoting N pairs.

Output:

For each test case output will be the length of the longest chain formed.

Constraints:
1<=T<=100
1<=N<=100

Example(To be used only for expected output):
Input
2
5
5 24 39 60 15 28 27 40 50 90
2
5 10 1 11

Output
3
1

Explanation
(i) the given pairs are {{5, 24}, {39, 60}, {15, 28}, {27, 40}, {50, 90} },the longest chain that can be formed is of length 3, and the chain is {{5, 24}, {27, 40}, {50, 90}}
(ii) The max length chain possible is only of length one.

Minimum Platforms

Given arrival and departure times of all trains that reach a railway station. Your task is to find the minimum number of platforms required for the railway station so that no train waits.

Note: Consider that all the trains arrive on the same day and leave on the same day. Also, arrival and departure times will not be same for a train, but we can have arrival time of one train equal to departure of the other.

In such cases, **we need different platforms,** i.e at any given instance of time, **same platform can not be used for both departure of a train and arrival of another.**

Input:
The first line of input contains T, the number of test cases. For each test case, first line will contain an integer N, the number of trains. Next two lines will consist of **N** space separated time intervals denoting arrival and departure times respectively.
Note: Time intervals are in the 24-hour format(hhmm), of the for **HHMM ,** where the first two charcters represent hour (between 00 to 23) and last two characters represent minutes (between 00 to 59).

Output:
For each test case, print the minimum number of platforms required for the trains to arrive and depart safely.

Constraints:
1 <= T <= 100

1 <= N <= 1000
1 <= A[i] < D[i] <= 2359

Example:
Input:
2
6
0900 0940 0950 1100 1500 1800
0910 1200 1120 1130 1900 2000
3
0900 1100 1235
1000 1200 1240

Output:
3
1

Explanation:
Testcase 1: Minimum 3 platforms are required to safely arrive and depart all trains.

17

Minimum Spanning Tree

Given a weighted, undirected and connected graph. The task is to find the sum of weights of the edges of the Minimum Spanning Tree.

Input:
The first line of input contains an integer **T** denoting the number of testcases. Then T test cases follow. The first line of each testcase contains two integers N (starting from 1), E denoting the number of nodes and number of edges. Then in the next line are 3*E space separated values a b w where a, b denotes an **edge** from **a** to **b** and **w** is the weight of the edge.

Output:
For each test case in a new line print the sum of weights of the edges of the Minimum Spanning Tree formed of the graph.

User task:
Since this is a functional problem you don't have to worry about input, you just have to complete the function **spanningTree()** which takes a graph **g** as its argument and returns an integer denoting the sum of weights of the edges of the Minimum Spanning Tree.

Constraints:
1 <= T <= 10
1 <= N<= 100
N-1 <= E<= 1000
1 <= w <= 1000

Example:
Input:
2
3 3
1 2 5 2 3 3 1 3 1

return sum;

2 1
1 2 5

Output:
4
5

Example:
Testcase 1: Sum of weights of edges in the minimum spanning tree is 4.

Count subsequences of type a^i b^j c^k

Given a string s, the task is to count number of subsequences of the form $a^ib^jc^k$, where i >= 1, j >=1 and k >= 1.

Note: Two subsequences are considered different if the set of array indexes picked for the 2 subsequences are different.

Examples:

```
Input  : abbc

Output : 3

Subsequences are abc, abc and abbc


Input  : abcabc

Output : 7

Subsequences are abc, abc, abbc, aabc

abcc, abc and abc
```

Input:
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains a string s.

Output:
For each test case in a new line print the required output.

Constraints:
1<=T<=100
1<=Length of string <=100

Example:
Input:

① if s[i] == 'a'
a = 1 + 2*a

② if s[i] == 'b'
b = a + 2*b

2
abbc
abcabc
Output:
3
7

Count of strings that can be formed using a, b and c under given constraints

Given a length n, count the number of strings of length n that can be made using ‘a’, ‘b’ and ‘c’ with at-most one ‘b’ and two ‘c’s allowed.

Input:

The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. The first line of each test case contains an integer N denoting the length of the string.

Output:

Output the count of the strings that can be formed under the given constraint.

Constraints:

1<= T <=100

1<= N <=1000

Example:

Input:

2

1

3

Output:

3

19

Solution: https://www.geeksforgeeks.org/count-strings-can-formed-using-b-c-given-constraints/

Unique BST's

Given an integer **N**, how many **structurally unique binary search trees** are there that store values 1...N?

Input:
First line of input contains **T** denoting the number of testcases. T testcases follow. Each testcase contains a single line of input containing N.

Ouput:
For each testcase, in a new line, print the answer.

Constraints:
1<=T<=15
1<=N<=11

Example:
Input:
2
2
3
Output:
2
5

Explanation:

Testcase1:
for N = 2, there are 2 unique BSTs

```
  1          2
   \        /
    2      1
```

Testcase2:
for N = 3, there are 5 possible BSTs

```
  1        3     3      2     1
   \      /     /      / \     \
    3    2     1      1   3     2
   /    /       \                \
  2    1         2                3
```

Solution Guide:
https://www.geeksforgeeks.org/total-number-of-possible-binary-search-trees-with-n-keys/

Count ways to N'th Stair(Order does not matter)

There are **N** stairs, and a person standing at the bottom wants to reach the top. The person can climb either **1 stair or 2 stairs at a time**. Count the number of ways, the person can reach the top (**order does not matter**).
Note: Order does not matter means for n=4 {1 2 1},{2 1 1},{1 1 2} are considered same.

Input:
The first line contains an integer 'T' denoting the total number of test cases. In each test cases, an integer **N** will be given.

Output:
For each testcase, in a new line, print number of possible ways to reach **N**th stair.

Constraints:
1 <= T <= 1000
1 <= N <= 10^6

Example:
Input:
2
4
5
Output:
3
3

Explanation:
Testcase 1: There are 3 ways to reach 4th stair.

Solution Guide: https://www.geeksforgeeks.org/count-ways-reach-nth-stair/

22

Distinct occurrences

Given two strings S and T, find count of distinct occurrences of T in S as a sub-sequence. Your task is to complete the function **subsequenceCount** which takes two strings as argument S and T and returns the count of the sub-sequences.

Input:
The first line of input contains an integer t denoting the no of test cases . Then t test cases follow. The first line of each test case contains 2 strings S and T.

Output:
For each test case in a new line print the count of distinct occurrences of T in S as a sub-sequence.

Constraints:
1<=t<=100
1<=length of (S,T)<=100

Example(To be used only for expected output):
Input
2
banana ban
geeksforgeeks ge

Output
3
6

Explanation:
(i) For first test case S = banana, T = ban there are 3 sub-sequences which are [ban], [ba n], [b

an].
(ii) For second test case S = geeksforgeeks, T=ge there are 6 sub-sequences which are [ge], [ge], [g e], [g e] [g e] and [g e].

Solution Guide: https://www.geeksforgeeks.org/count-distinct-occurrences-as-a-subsequence/

Reach a given score

Consider a game where a player can score **3** or **5** or **10** points in a move. Given a total score **n**, find number of distinct combinations to reach the given score.

Input:
The first line of input contains an integer **T** denoting the number of test cases. T testcases follow.The first line of each test case is **n**.

Output:
For each testcase, in a new line, print **number of ways/combinations** to reach the given score.

Constraints:
1 ≤ **T** ≤ 100
1 ≤ **n** ≤ 1000

Example:
Input:
3
8
20
13

Output:
1
4
2

Explanation
For 1st example when n = 8
{ 3, 5 } and {5, 3} are the two possible permutations but these represent the same cobmination. Hence output is 1.

Solution Guide: https://www.geeksforgeeks.org/count-number-ways-reach-given-score-game/

Count number of hops

A frog jumps either **1**, **2** or **3** steps to go to top. In how many ways can it reach the top.

Input:
The first line of input contains an integer T denoting the number of test cases. T testcases follow. Each testcase contains one line of input **N** denoting the total number of steps.

Output:
For each testcase, in a new line, print the number of ways to reach the top.

Constraints:
1 ≤ T ≤ 100
1 ≤ N ≤ 50

Example:
Input:
2
1
5
Output:
1
13

Solution Guide: https://www.geeksforgeeks.org/count-number-of-ways-to-cover-a-distance/

Maximum Profit

In stock market , a person buys a stock and sells it on some future date. Given the stock prices of **N** days in an form of an array **A[ ]** and a positive integer **K**, find out the maximum profit a person can make in atmost **K** transactions. A transaction is equivalent to (buying + selling) of a stock and new transaction can start only when the previous transaction has been completed.

Input
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow.
The first line of each test case contains a positve integer K, denoting the number of transactions.
The second line of each test case contains a positve integer N, denoting the length of the array A[ ].
The third line of each test case contains a N space separated positive integers, denoting the prices of each day in the array A[ ].

Output
Print out the maximum profit earned by the person.
No profit will be equivalent to **0.**

Constraints
1 <= T <= 100
0 < K <= 10
2 <= N <=30
0 <= A[ ] <= 1000

Examples

Input
3
2
6
10 22 5 75 65 80
3

4
20 580 420 900
1
5
100 90 80 50 25

Output
87
1040
0

Explanation:
Output 1: Trader earns 87 as sum of 12 and 75 i.e. Buy at price 10, sell at 22, buy at 5 and sell at 80
Output 2: Trader earns 1040 as sum of 560 and 480 i.e. Buy at price 20, sell at 580, buy at 420 and sell at 900
Output 3: Trader cannot make any profit as selling price is decreasing day by day.Hence, it is not possible to earn anything.

Solution Guide:
https://www.geeksforgeeks.org/maximum-profit-by-buying-and-selling-a-share-at-most-k-times/