

Prime Path

Given an $N \times N$ grid G and each cell of the grid contains an integer.

In one step you can travel from any cell $G_{i_1 j_1}$ to $G_{i_2 j_2}$ if $abs(j_1 - j_2) + abs(i_1 - i_2) \leq p$ at a cost of $floor(\sqrt{G_{i_1 j_1}})$.

Here p is the number of unique prime factors of the integer at $G_{i_1 j_1}$.

Find the minimum cost to move from the top-left corner cell to the bottom-right corner cell, no matter how many steps you take.

Notes

- $abs(x)$ means taking the absolute value of a number x or $|x|$.
- $floor(x)$ represents the largest integer y such that $y \leq x$.
- \sqrt{x} represents the square root of x .
- Use 1 based indexing

Function description

Complete the *solve* function. This function takes the following 2 parameters and returns the minimum cost to reach the bottom-right corner of the grid:

- N : Represents the size of the grid
- G : Represents a double dimension array of size $N \times N$ denoting the grid

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains T , which represents the number of test cases.
- For each test case:
 - The first line contains an integer N representing the size of the grid.
 - The next N lines contain N -separated integers denoting the grid.

Output format

For each test case, print the minimum cost required to reach the bottom-right corner of the grid in a new line.

Output format

For each test case, print the minimum cost required to reach the bottom-right corner of the grid in a new line.

Constraints


$$1 \leq T \leq 5$$

$$1 \leq N \leq 10^3$$

$$2 \leq G_{ij} \leq 10^6$$

$$1 \leq i \leq N$$

$$1 \leq j \leq N$$

Sample input 



Sample output 



```
2
3
13 12 15
5 6 9
2 4 8
3
2 3 5
30 4 6
5 2 8
```

```
7
5
```

Explanation

The first line contains the number of test cases, $T = 2$.

Explanation

The first line contains the number of test cases, $T = 2$.

Test case 1

Given

- $N = 3$
- Grid G is as follows:

13	12	15
5	6	9
2	4	8

Approach

You can follow these steps:

1. You start at $(1, 1)$ From $(1, 1)$ we can move to $(2, 1)$ since $(2-1)+(1-1) \leq 1$ (the number of unique prime factors of 13 is 1). The cost of this step is 3.
2. From $(2, 1)$ you can move to $(2, 2)$ since $(2-2)+(2-1) \leq 1$ (the number of unique prime factors of 5 is 1). The cost of this step is 2.
3. From $(2, 2)$ we can move to $(3, 3)$ since $(3-2)+(3-2) \leq 2$ (the number of unique prime factors of 6 is 2). The cost of this step is 2.

Thus to move from $(1, 1)$ to $(3, 3)$ you require 3 steps and the minimum required cost is 7.

Sample input 1 ↗



```
2
3
18 3 4
6 5 9
13 20 14
3
18 10 20
14 2 5
14 19 10
```

Sample output 1



```
9
7
```

Sample input 2 ↗



```
2
3
14 20 4
19 12 5
20 15 12
3
3 6 3
6 18 18
4 15 9
```

Sample output 2



```
6
6
```

Note:

Your code must be able to print the sample output from the provided sample input. However, your code is run against multiple hidden test cases. Therefore, your code must pass these hidden test cases to solve the problem statement

Limits

Time Limit: 5.0 sec(s) for each input file

Memory Limit: 256 MB

Source Limit: 1024 KB

Scoring

Score is assigned if any testcase passes

Allowed Languages

Bash, C, C++14, C++17, Clojure, C#, D, Erlang, F#, Go, Groovy, Haskell, Java 8, Java 14, JavaScript(Node.js), Julia, Kotlin, Lisp (SBCL), Lua, Objective-C, OCaml, Octave, Pascal, Perl, PHP, Python, Python 3, Python 3.8, R(RScript), Racket, Ruby, Rust, Scala, Swift, TypeScript, Visual Basic

N Toys

Given N toys in a shop, each with a price represented by an array A . You have a certain amount of money, C , that you can use to buy toys. There are also K toys that are broken and you don't want to buy them.

For each Q query, find the maximum number of toys you can buy with the amount you have while avoiding broken toys.

Notes

- Use 1-based indexing.
- Query definition:
 - The first element represents an integer C , where $C = \text{Query}[i][0]$.
 - The second element represents an integer K , where $K = \text{Query}[i][1]$.
 - The next K integers represent the indices of broken toys which are $\text{Query}[i][j]$ for all $j > 1$.
- Treat each query independently.

Function description

Complete the *solve* function. This function takes the following 4 parameters and returns an array of an answer to the query:

- N : Represents the size of array A
- A : Represents the array of costs of each toy

- N : Represents the size of array A
- A : Represents the array of costs of each toy
- Q : Represents the number of queries
- $Query$: Represents the query array

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains T , which represents the number of test cases.
- For each test case:
 - The first line contains a single integer N representing the size of array A .
 - The second line contains N space-separated integers representing the cost of toys.
 - The third line contains an Integer Q representing the number of queries.
 - The next Q lines contain the query array.

Output format

For each test case, print Q space-separated integers representing the maximum number of toys you can purchase in a new line.

Median Path

Given a tree of n nodes and $n - 1$ edges. Each node has a value, which is stored in an array C . C_i ($1 \leq i \leq n$) denotes the value of node i .

Find the sum of the medians of all simple paths of odd length starting from node 1.

Notes

- Use *1-based* indexing
- The median is the number in the middle of a list of numbers when the numbers are arranged from smallest to biggest (or biggest to smallest).
- In graph theory, a simple path is a path in a graph that does not go through the same vertex twice.
- A tree is a graph where any two nodes are connected by exactly one path. This means that there is no way to get from one node to another by going through the same node twice. Trees are also acyclic, which means that there are no loops. This means that you can't start at any node and end up back at the same node by following the edges

Function description

Complete the *solve* function. This function takes the following 3 parameters and returns an Integer:

- *n*: Represents the number of nodes in a tree
- *C*: Represents an array denoting the node values
- *edges*: Represents a 2D array of edges

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains *T*, which represents the number of test cases.
- For each test case:
 - The first line contains an Integer *n* denoting the number of nodes.
 - The second line contains *n* space-separated integers representing the node values.
 - Next, *n - 1* line contains two space-separated integers representing an edge between the nodes.

Output format


For each test case, print the answer in a new line representing the sum of medians of every *simple path* of odd length starting from node 1.

Constraints

$$1 \leq T \leq 10$$

$$1 \leq n \leq 10^5$$

$$1 \leq C_i \leq 10^9$$

Sample input 



Sample output

```
1
5
7 6 9 10 1
1 2
2 3
1 4
2 5
6
```

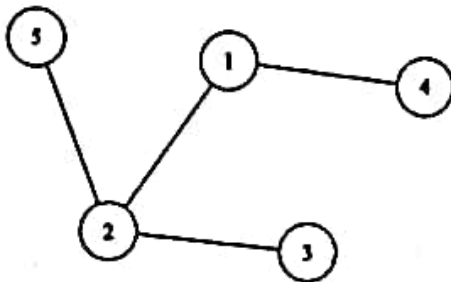
```
20
7
```

Explanation

The first test case

Given

- $n = 5$
- $C = [7, 6, 9, 10, 1]$
- Tree:



Approach

All paths of odd length and their median are:

- C_1 value in path from 1 to 1 is ($\text{node}_1 = 7$) and the median is 7.
- C_1 values in path from 1 to 3 are ($\text{node}_1 = 7, \text{node}_2 = 6, \text{node}_3 = 9$) and their median is 7.
- C_1 values in path from 1 to 5 are ($\text{node}_1 = 7, \text{node}_2 = 6, \text{node}_3 = 1$) and their median is 6.

Hence, the final answer will be $7 + 7 + 6 = 20$.

Hybrid Maximum

A hybrid sequence is a sequence that can be divided into two disjoint subsequences. Every subsequence is an array.

Given two arrays, A and B, of sizes N and M, respectively. You are also given a hybrid sequence S.

The expression $(\sum_{i=1}^{N+M} \max\{S_1, \dots, S_i\} - \sum_{i=1}^{N+M} \min\{S_1, \dots, S_i\})$ calculates the difference between the sum of the maximum values and the sum of the minimum values in the hybrid sequence S.

Find the maximum value of this expression for all possible hybrid sequences.

Notes

- Use *1-based* indexing.
- A subsequence is a sequence obtained by deleting zero or more numbers (not necessarily consecutive) from the original sequence without changing the order of the remaining elements.

Function description

Complete the function *hybridMaximum*. This function takes the following 4 parameters and returns the maximum value of the expression that can be obtained:

- *N*: Represents the number of elements in *A*
- *A*: Represents the elements in *A*
- *M*: Represents the number of elements in *B*
- *B*: Represents the elements in *B*

Input format for custom testing

Note: Use this input format if you are testing against custom input or writing code in a language where we don't provide boilerplate code.

- The first line contains *T*, which represents the number of test cases..
- For each test case:
 - The first line contains an integer *N* denoting the number of elements in *A*.
 - The second line contains *N* space-separated integers denoting the elements in *A*.
 - The third line contains an integer *M* denoting the number of elements in *B*.

- The third line contains an integer M denoting the number of elements in B .
- The fourth line contains M space-separated integers denoting the elements in B .

Output format

For each test case, print the maximum value of the expression that can be obtained, in a new line.

Constraints

$$1 \leq T \leq 1000$$

$$1 \leq N, M \leq 1000$$

$$1 \leq A_i \leq 10^6 \quad \forall i \in [1, N]$$

$$1 \leq B_i \leq 10^6 \quad \forall i \in [1, M]$$

$$\text{Sum of } N \times M \text{ over all test cases} \leq 10^7$$

Sample input ↗

```
1
3
1 2 3
2
4 1
```

Sample output

```
12
```