

Problemsetting (problemsetting)

This problem is loosely based on real events.

The organising committee of an unspecified programming contest (UPC) is planning to hold many rounds this year, each round consisting of n problems of varying difficulties.


As such, there have been very many problem proposals. Each problem proposal has been assigned a difficulty rating between 1 and n .

The difficulty rating for a specific problem can either be:

- An integer i ($1 \leq i \leq n$) — meaning that this problem is appropriate only as the i -th problem in an UPC round; or
- Equal to $i.5$, for some integer i ($1 \leq i < n$) — meaning that this problem is appropriate as either the i -th or the $i + 1$ -th problem in an UPC round.

Given n (the number of problems in an UPC round), and the number of problem proposals for each difficulty rating, print the maximum number of rounds that can be held using these problems.

Note that each problem can only be used in at most one round.

 Among the attachments of this task you may find a template file `problemsetting.*` with a sample incomplete implementation.

Input

Each test contains multiple test cases. The first line of input contains a single integer t — the number of test cases.

The first line of each test case contains a single integer n — the number of problems in an UPC round.

The second line of each test case contains n integers a_1, a_2, \dots, a_n , where a_i denotes the number of problem proposals with a difficulty rating of i .

The third line of each test case contains $n - 1$ integers b_1, b_2, \dots, b_{n-1} , where b_i denotes the number of problem proposals with a difficulty rating of $i.5$.

Output

For each test case, print a single integer, the maximum number of rounds that can be held using the available problems.

Constraints

- $1 \leq t \leq 10\,000$.
- $2 \leq n \leq 200\,000$.
- $0 \leq a_i \leq 1\,000\,000\,000$ for each $i = 1 \dots n$.
- $0 \leq b_i \leq 1\,000\,000\,000$ for each $i = 1 \dots n - 1$.
- It is guaranteed that the sum of n across all test cases does not exceed 200 000.

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- Subtask 1 (0 points)

Examples.
- Subtask 2 (5 points)

$b_i = 0$ for each $i = 1 \dots n - 1$.
- Subtask 3 (10 points)

$a_i \leq 1$ for each $i = 1 \dots n$; $b_i \leq 1$ for each $i = 1 \dots n - 1$.
- Subtask 4 (30 points)

$a_i \leq 100$ for each $i = 1 \dots n$; $b_i \leq 100$ for each $i = 1 \dots n - 1$.
- Subtask 5 (20 points)

$a_i = 0$ for each $i = 1 \dots n$.
- Subtask 6 (35 points)

No additional limitations.

Examples

input	output
5	1
6	0
0 1 1 0 1 1	2
1 0 1 0 1	3
5	4
0 1 1 1 0	
1 0 1 0	
2	
1 1	
2	
6	
0 0 0 0 0 0	
5 2 5 100 9	
7	
1 3 0 4 1 0 2	
4 3 1 3 1 5	

Explanation

In the first test case, there are 7 problem proposals, which have difficulty ratings of 1.5, 2, 3, 3.5, 5, 5.5 and 6, respectively.

We can create a single round using these problems:

- The problem with a difficulty rating of 1.5 will be used as the first problem in the round.
- The problem with a difficulty rating of 2 will be used as the second problem in the round.
- The problem with a difficulty rating of 3 will be used as the third problem in the round.
- The problem with a difficulty rating of 3.5 will be used as the fourth problem in the round.

- The problem with a difficulty rating of 5.5 will be used as the fifth problem in the round.
- The problem with a difficulty rating of 6 will be used as the sixth problem in the round.

In the second test case, we can show that it is impossible to create a complete round from the given proposals.

In the third test case, it is possible to create the following rounds from the given proposals:

- $[1, 1.5]$
- $[1.5, 2]$

In the fourth test case, it is possible to create the following rounds from the given proposals:

- $[1.5, 1.5, 3.5, 3.5, 5.5, 5.5]$
- $[1.5, 1.5, 3.5, 3.5, 5.5, 5.5]$
- $[1.5, 2.5, 2.5, 3.5, 5.5, 5.5]$