Problem A: Corn Maze

Filip just returned from a corn maze and now he is trying to reconstruct the map. There were nine stations in the corn maze and at each station one could pick up an $n \times n$ bitmap of a part of the maze. The complete map is a $(3n+2) \times (3n+2)$ bitmap which consists of all nine pieces and an additional outer rim of width 1 pixel. The outer rim has all pixels occupied, except the pixel at position $(1, 2)$ (entrance) and the pixel at position $(3n+2, 3n+1)$ (finish). Filip had all nine $n \times n$ pieces arranged inside the outer rim but unfortunately they became unglued. Filip is trying to figure out how many possible mazes can he create by rearranging the nine pieces inside the outer rim. He does not want to rotate the pieces. The pieces have to completely tile the space inside the outer rim and, of course, one should be able to pass from the entrance to the finish.

Input specification:

The first line contains the number of instances of the problem. Each instance is described on several lines. The first line contains n. Then 9 bitmaps follow, each bitmap is described on n lines, each line containing n zeros and ones separated by white space (zeros represent empty spaces, ones represent occupied spaces). You may assume that n is at most 200.

Output specification:

The output contains one line for each instance. The line contains the number of different mazes Filip can create in that instance.

Sample input:

```
1
1
0
0
0
0
1
0
0
0
0
```

Sample output:

```
7
```

Problem B: Planters

Julia has a lot of house plants in planters of different sizes. The plants have grown quite a bit and each needs to be replanted to a larger planter. So Julia sets to work. She found some extra planters, bought some more, and now she is ready to start replanting. Once she moves a plant to a larger planter, the original planter can be used for another plant. But wait! Is it indeed possible to replant the plants so that each gets a larger planter?

Input specification:

The first line contains the number of test cases. Each test case is described on several lines. The first line contains n and m, the number of plants and the number of additional, initially empty, planters. The next line contains n numbers indicating the sizes of the planters with plants, there is one plant in each planter. The next line contains m numbers indicating the sizes of additional empty planters. You may assume that n and m are at most 1,000,000 and that the sizes are positive integers not larger than 10,000,000.

Output specification:

The output contains one line for each test case. The line contains a single string "YES" or "NO", indicating whether it is possible to replant the plants so that each plant gets a larger planter.

Sample input:

```
2
4 2
1 4 3 2
5 1
3 1
2 2 2
3
```

Sample output:

```
YES
NO
```

Problem C: Cards

Žofka and her dad invented a new card game. First, take a deck with an even number of cards where each card has an integer number written on it. Then, as part of the game set up, split the cards randomly into pairs and lay each pair face up on the table. The players now play individually and in parallel: Each player chooses exactly one number from each pair of cards and writes the number on paper (without moving the cards). After choosing from all pairs, the player sums his or her chosen numbers. The player whose sum is the closest to the median of all possible values that can be obtained this way wins. Can you help them find a winning sum?

Recall that the median is the value that appears in the middle position if we sort all of the values. If the number of values is even, then we average the values in the two middle positions.

Input specification:

The first line contains the number of test cases. Each test case is described on several lines. The first line contains n, the number of pairs of cards. The next n lines describe the pairs: The i-th line contains two integers, the numbers on the cards in the i-th pair. You may assume that n is at most 30 and each of the numbers is between 0 and 10,000.

Output specification:

The output contains one line for each test case. The line contains a single number: a winning sum for the test case. Out of all possible sums obtained by choosing exactly one number from each pair and adding them up, output the sum that is the closest to the median of these sums. If there are several possible answers, output the smallest one.

Sample input:

```
2
3
0 1
2 1
4 6
2
1 1
2 3
```

Sample output:

```
7
3
```

Explanation:

For the first sample input, we can get the following sums: $0 + 2 + 4 = 6$, $0 + 2 + 6 = 8$, $0 + 1 + 4 = 5$, $0 + 1 + 6 = 7$, $1 + 2 + 4 = 7$, $1 + 2 + 6 = 9$, $1 + 1 + 4 = 6$, $1 + 1 + 6 = 8$. If we sorted the sums, we would get $5, 6, 6, 7, 7, 8, 8, 9$, and the median is 7. Therefore, 7 is the closest sum to the median.

For the second sample input, we can get the following sums: $1 + 2 = 3$, $1 + 3 = 4$, $1 + 2 = 3$, $1 + 3 = 4$. In sorted order they are $3, 3, 4, 4$ and the median is 3.5. Sums 3 and 4 are the closest to the median, so we output the smaller: 3.

Problem D: Magnetic Tiles

Bob has a collection of square magnetic tiles. On each square tile there are 4 magnets, one on each side. There are two ways each magnet can be oriented—South (S) or North (N). There are only 6 types of tiles (we can rotate each tile so that if we list the orientations of the magnets on the edges we get one of the following six lists: SSSS, SSSN, SSNN, SNSN, SNNN, and NNNN). To build a good cube one takes six tiles and arranges them in a cube in such a way that every pair of neighboring magnets has opposite orientation (there are 12 pairs of magnets in each cube). Bob has one more strange requirement: each good cube can have at most 2 different types of tiles. Bob wants to know what is the largest number of good cubes he can build from his collection of magnetic tiles.

Input specification:

The first line contains the number of instances of the problem. Each instance is described on one line containing the number of tiles of each type (in the following order: SSSS, SSSN, SSNN, SNSN, SNNN, and NNNN). You may assume that each number is between 0 and 1,000,000,000.

Output specification:

The output contains one line for each instance. The line contains the largest number of good cubes that can be built from the tiles in that instance.

Sample input:

```
3
0 0 30 30 0 0
10 0 0 0 0 10
2 5 0 0 5 2
```

Sample output:

```
10
0
2
```

Problem E: Paper cut

Anna has a lot of left-over pieces of paper from an earlier project. Each of the pieces forms a polygon that is placed on a coordinate system by aligning one of the sides with the x-axis and, as we traverse the polygon, each vertex has x coordinate at least as large as the previous vertex. For her next project, Anna wants to cut a rectangle with the largest possible area out of each piece. Because of the special pattern on the paper, the rectangle has to have one of its sides aligned with the x-axis. How big can the rectangle be?

Input specification:

The first line contains the number of test cases. Each test case is described on several lines. The first line contains n, the number of vertices of the polygon that represents a piece of paper. The next n lines describe the coordinates of the vertices: the i-th line contains x_i and y_i, the coordinates of the i-th vertex when traversing the polygon in clockwise order. The first and the last vertex are on the x-axis, that is, $y_1 = y_n = 0$. Additionally, $x_1 \leq x_2 \leq \cdots \leq x_n$ and $y_i > 0$ for each $i \in \{2, \ldots, n-1\}$. All of the coordinates are integers with absolute value not larger than 10,000,000 and n is at least 3 and at most 1,000,000. All vertices are distinct and no three consecutive vertices are collinear.

Output specification:

The output contains one line per test case. The line contains a single number, the area of the largest rectangle with one side on the x-axis that fits in the polygon. Output the area with precision of two decimal places, rounded down (that is, for example, 3.1415 will get rounded down to 3.14, and 2.7182 will get rounded down to 2.71).

Sample input:

```
4
3
0 0
4 8
8 0
11
-4 0
-1 4
-1 8
0 5
2 9
4 4
6 5
6 8
8 8
8 2
```

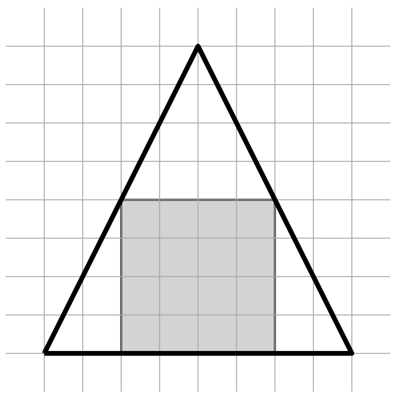
```
12 0
4
0 0
1 4
2 1
3 0
4
0 0
0 10000000
10000000 10000000
10000000 0
```

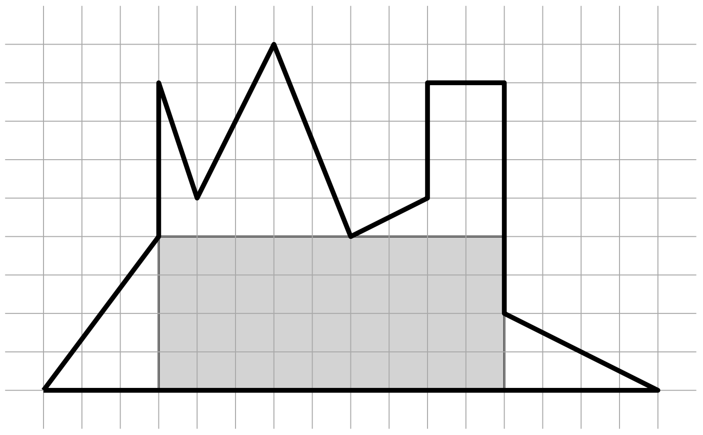
Sample output:

```
16.00
36.00
2.33
100000000000000.00
```

Explanation:

The first two sample inputs and the corresponding rectangles are shown below:





Problem F: Flea Circus

In the famous flying flea circus there are n fleas and n boxes. The boxes are numbered $1, \ldots, n$ and the fleas are also numbered $1, \ldots, n$. Initially there is exactly one flea on each box. We are given n numbers $a_1, \ldots, a_n$ where a_i is the number of the flea on box i. Each flea wants to be on the box whose number is the same as the number of the flea. Each second the tamer cracks the whip, some of the fleas pair up and then each pair switches their position (after the swap there is again exactly one flea on each box). How quickly can the fleas achieve happiness (that is, what is the smallest number of seconds needed for each flea to reach the box whose number agrees with the number of the flea)?

Input specification:

The first line contains the number of instances of the problem. Each instance is described on two lines. The first line contains n (the number of the fleas and also the number of the boxes). The next line contains numbers $a_1, \ldots, a_n$. You may assume that $n \leq 1{,}000$.

Output specification:

The output contains one line for each instance. The line contains the smallest number of steps needed for each flea to end up on the box whose number agrees with the number of the flea.

Sample input:

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

Sample output:

```
0
1
2
```

Explanation:

For sample input 1 the fleas are already happy. For sample input 2 pairs 1-2 and 3-4 switch position (at the first whip crack) and all fleas are happy. For sample input 3 first pair 2-3 switches position (at the first whip crack) and then pair 1-3 switches position (at the second whip crack) making all fleas happy.

Problem G: An Odd Median Obsession

Medard has an odd obsession with medians. Recall that the median of a sequence of numbers $a_1, \ldots, a_{2k+1}$ is b_{k+1} where $b_1 \leq b_2 \leq \cdots \leq b_{2k+1}$ is the sorting of $a_1, \ldots, a_{2k+1}$ (Medard doesn't like the definition of the median for sequences of even length so we will spare you the details). He has a weighted directed graph containing two vertices s, t, and he wants to find the odd s-t-path with the smallest possible median weight (we call a path odd if it has an odd number of edges).

The vertices of the graph are $\{1, \ldots, n\}$. There are m edges $(a_1, b_1), \ldots, (a_m, b_m)$ where $1 \leq a_i < b_i \leq n$ for all $i \in \{1, \ldots, m\}$ (note that each edge goes from a vertex with a smaller number to a vertex with a larger number). The weight of the i-th edge is w_i. The starting vertex is $s = 1$ and the ending vertex is $t = n$. An s-t-path is a sequence of vertices $v_1, \ldots, v_k$ where $v_1 = s$, $v_k = t$ and (v_i, v_{i+1}) is an edge for $i \in \{1, \ldots, k-1\}$. We call the path odd if k is even (sounds odd, I know). The median weight of the path is the median of the sequence $w((v_1, v_2)), w((v_2, v_3)), \ldots, w((v_{k-1}, v_k))$.

Input specification:

The first line contains the number of instances of the problem. Each instance is described on several lines. The first line contains n and m (the number of vertices and the number of edges). Then m lines follow, the i-th line contains the three integers a_i, b_i, w_i. You may assume that $n \leq$ 10,000, $m \leq$ 200,000 and the weights are nonnegative integers not larger than 1,000,000,000. The graph is simple (that is, there is at most one edge between each pair of vertices).

Output specification:

The output contains one line for each instance. The line contains the smallest possible median weight of any odd s-t-path in the instance. If there is no odd s-t-path in the instance then the line should contain "NO".

Sample input:

```
2
3 2
1 2 1
2 3 1
4 5
1 2 1
2 3 2
3 4 3
1 3 1
1 4 5
```

Sample output:

```
NO
2
```

Explanation:

For sample input 1, there is no odd s-t-path (the only path is $1, 2, 3$, which is even). For sample input 2, there are 2 odd s-t paths: $1, 2, 3, 4$ (median weight 2) and $1, 4$ (median weight 5).

Problem H: Hideout

In a galaxy far, far away, the rebels just arrived at planet XY and noticed several imperial lookout posts. The rebels need to build a base that will serve as their hideout and they have enough Force to deactivate some of the posts to make the base undetectable. Each active post can detect the rebel base, unless there is another post in the way. Where should the rebels build the base so that they need to deactivate the smallest number of posts?

The rebels can also build their base at the location of one of the imperial posts. But they need to deactivate (and destroy) it.

Input specification:

The first line contains the number of test cases. Each test case is described on several lines. The first line contains n, the number of lookout posts. The next n lines give the coordinates of the lookout posts: The i-th line contains the coordinates x_i and y_i of the i-th lookout post. (Planet XY is an infinite plane.) You may assume that no two lookout posts share the same coordinates. You may also assume that n is at most 50 and the coordinates are integers with absolute value at most 100.

Output specification:

The output contains one line for each test case. The line contains a single number, the minimum number of posts that need to be deactivated.

Sample input:

```
3
9
0 3
-5 0
-1 4
-1 -4
1 2
1 -2
5 0
3 -4
2 1
10
-5 0
-3 0
-1 0
0 0
0 -2
0 2
0 3
```

```
0 5
2 0
4 0
3
0 0
2 0
3 4
```

Sample output:

```
4
5
2
```

Explanation:

The inputs are shown below, with the location of the base. For the first input (on the left), the base is at coordinates $(7, -4)$ where only the four highlighted posts need to be deactivated and the remaining posts cannot detect the base since there is another post in the way. For the second input (on the right), the base is at coordinates $(0, 0)$, where a post has to be deactivated. Additional four posts also need to be deactivated, bringing the total to five. For the third input, the base can be, for example, at coordinates $(-3.5, 0)$ and two posts need to be deactivated. **Note that the coordinates of the base do not have to be integer and do not have to have absolute value at most** 100**.**