Official Problem Set

DO NOT OPEN UNTIL CONTEST BEGINS

2017 ACM ICPC ASIA GWALIOR REGIONAL CONTEST

NOTE: The order of problems here can be different than the one displayed on the contest page.

Problem code: **TRICHEF**
Problem name: **Chef and Triangles**

Chef has come to a 2 dimensional garden in which there are **N** points. Each point has coordinates **(x, y)**, where **x** can either be **1** or **2** or **3**. Chef will now choose every triplet of these **N** points and make a triangle from it. You need to tell the sum of areas of all the triangles the Chef makes.

Note that some of the triplets might not form proper triangles, and would end up as a line or a point (ie. degenerate), but that is fine because their area will be zero.

Input

- The first line contains a single integer **T**, the number of test cases. The description of each testcase follows.
- The first line of each test case contains an integer **N** denoting the number of points on the plane.
- The next **N** lines contain 2 space separated integers **x** and **y** denoting the coordinates of the points.

Output

For each test case, output a single line containing the answer. Your answer will be considered correct if the absolute error is less than or equal to 10^{-2}.

Constraints

- $1 \leq \mathbf{T} \leq 20$
- $1 \leq \mathbf{N} \leq 2000$
- $1 \leq \mathbf{x} \leq 3$
- $1 \leq \mathbf{y} \leq 10^6$
- All (**x**, **y**) pairs are distinct

Example

Input:

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

Output:

```
1.0
2.0
```

Explanation:

Test Case 1: There is only one triangle which has non-zero area, and it's area is 1, hence the output.

Test Case 2: Let the points be A(1,1), B(2,2), C(2,1), D(3,3). There are 3 non degenerate triangles possible.

- area ABC = 0.5
- area BCD = 0.5
- area ACD = 1

Total area = 2

Problem code: **QTRAGAIN**
Problem name: **Queries on tree again**

You have a rooted tree with **N** nodes which are numbered from 1 to **N**. It is rooted at node 1. Each node has a value associated with it, which is 0 initially. The distance between two nodes is the number of edges in the unique path between them. You need to support the following two operations:

- Update operation at node **x** with value **y**:
  - Add 2^{y} to node **x**.
  - Consider every node in the subtree of **x**, and suppose d is distance from **x** to the node. If **y** - d ≥ 0, add $2^{(y - d)}$ to the value associated with that node.
  - Consider all the ancestors of **x**, and suppose d is distance from **x** to the ancestor. If **y** - d ≥ 0, add $2^{(y - d)}$ to the value associated with that ancestor.
- Query operation at node **x**:
  - Output the current value associated with node **x**, modulo $10^9 + 7$.

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 line of each test case contains two space-separated integers **N, Q** denoting the number of nodes, and the number of operations respectively.
- Each of the next **N** - 1 lines contains two space-separated integers: **u**, **v**, denoting that there is an edge between nodes **u** and **v** in the tree.
- Each of the next **Q** lines contain description of an operation, which will be of one of these formats:
  - Update operation: You will be given three space-separated integers: 1, **x**, **y**.
  - Query operation: You will be given two space-separated integers: 2, **x**.

Output

For each Query operation, output a single integer in a new line corresponding to the answer of the query.

Constraints

- $1 \leq \mathbf{T} \leq 10^6$
- $1 \leq \mathbf{N} \leq 10^5$
- $1 \leq \mathbf{Q} \leq 1.5 * 10^6$
- $1 \leq \mathbf{u}, \mathbf{v}, \mathbf{x} \leq \mathbf{N}$
- $1 \leq \mathbf{y} \leq 60$
- Sum of **N** over all test cases in a single file $\leq 10^6$
- Sum of **Q** over all test cases in a single file $\leq 1.5 * 10^6$

Example

Input

```
1
4 8
1 2
2 3
2 4
1 1 2
1 2 3
1 3 1
1 4 2
2 1
2 2
2 3
2 4
```

Output

```
9
13
7
9
```

Explanation

Let the values associated with the nodes be v_1, v_2, v_3, and v_4.

- Initially, $v_1 = 0$, $v_2 = 0$, $v_3 = 0$, and $v_4 = 0$
- After first update operation, $v_1 = 4$, $v_2 = 2$, $v_3 = 1$, and $v_4 = 1$
- After second update operation, $v_1 = 4 + 4$, $v_2 = 2 + 8$, $v_3 = 1 + 4$, and $v_4 = 1 + 4$
- After third update operation, $v_1 = 4 + 4$, $v_2 = 2 + 8 + 1$, $v_3 = 1 + 4 + 2$, and $v_4 = 1 + 4$
- After fourth update operation, $v_1 = 4 + 4 + 1$, $v_2 = 2 + 8 + 1 + 2$, $v_3 = 1 + 4 + 2$, and $v_4 = 1 + 4 + 4$

So after this, we have $v_1 = 9$, $v_2 = 13$, $v_3 = 7$, and $v_4 = 9$, and this is outputted in the following query operations.

Problem code: **OPTCODE**
Problem name: **Optimize The Slow Code**

Chef hates unoptimized codes and people who write such codes. One fine day he decided to look through the kitchen's codebase and found a function whose pseudo-code is given here:

```
input: integer N, list X[1, 2, ..., N], list Y[1, 2, ..., N]

output: integer res

function:

set res = 0;
for i := 1 to N do
   for j := 1 to N do
      for k := 1 to N do
         if (X[i] = X[j]) OR (X[j] = X[k]) OR (X[k] = X[i])
            continue
         else
            set res = max(res, Y[i] + Y[j] + Y[k])
return res
```

Luckily enough this code gets triggered only if the Head Chef makes a submission. But still there is a possibility that this can crash the judge. So help Chef by writing a new function which does the same thing but is faster.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The first line of each test case contains an integer **N** denoting the number of elements in the two lists.
- The i-th of the next **N** lines contains a pair of space-separated integers denoting the values of **X[i]** and **Y[i]** respectively.

Output

For each test case, output an integer corresponding to the return value of the function.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{N} \leq 10^5$
- $1 \leq \mathbf{X[i]}, \mathbf{Y[i]} \leq 10^8$

Example

Input

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

Output

```
0
11
```

Explanation

Testcase 2: The maximum is attained when i = 1, j = 2 and k = 5. This leads to res being 3 + 4 + 4 = 11. This value is attained in other iterations as well, but it never exceeds this, and hence this is the answer.

Problem code: **COUPSYS**
Problem name: **Coupon System**

Chef is organizing an exam for certifying algorithmic skills. The exam will consist of three levels: 1, 2 and 3, and will be organized at various cities numbered from 1 to 100.

You have received **n** coupons which give discounts for the exam. Each coupon can be applied for a particular city, for a given level and will provide a certain discount.

You want to attempt all the three levels of the exam, and you are willing to go to different cities to avail the maximum discount. So, for each level, find out the maximum discount that you can avail, and which city you'd need to go to to do so. If there are multiple such cities, choose the city that has the least number. That is, if the discounts are same, city i is preferable over city j, if i < j.

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 line of each test case contains a single integer **n** denoting the number of coupons.
- Each of the next **n** lines contains three space-separated integers city **c**, level **l** and discount **x** applicable in the i-th coupon.

Output

For each test case, output three lines. The first line should contain two space-separated integers: maximum discount for level 1, and the city where you can avail this discount. In case of ties, find the smallest numbered such city. Similarly, second and third line should contain similar info for second and third levels.

Constraints

- $1 \leq \mathbf{T} \leq 100$
- $3 \leq \mathbf{n} \leq 100$
- $1 \leq \mathbf{c} \leq 100$
- $1 \leq \mathbf{l} \leq 3$
- $1 \leq \mathbf{x} \leq 100$
- It is guaranteed that for each level, there will at least one coupon.

Example

Input

```
1
7
2 1 10
1 1 20
3 2 100
2 3 50
2 2 100
5 3 75
100 3 75
```

Output

```
20 1
100 2
75 5
```

Explanation

For Level 1, there are two discount coupons, which give a discount of 10 and 20 respectively. We choose the maximum, and that is for city 1. Hence the first line of output is "20 1".

For Level 2, there are two discount coupons, both of which give a discount of 100. We choose the one which is for the city with the smaller number, and that is for city 2. Hence the second line of output is "100 2".

For Level 3, there are three discount coupons, which give discounts of 50, 75 and 75. The maximum discount is 75, and so that leaves us with two coupons to decide from. We choose the one which is for the city with the smaller number, and that is for city 5. Hence the third line of output is "75 5".

Problem code: **KALADIN**
Problem name: **Proprietary Probabilistic Problem**

You are given a bag containing **N** balls, numbered from 1 to **N**, each having a given non-zero integral volume.

The likelihood of a ball being taken out of the bag is directly proportional to its volume. More specifically, the probability of a ball being taken out of the bag is equal to it's volume divided by the sum of volumes of all the balls in the bag at that point.

You are bored and decide to play a game. It has **N** Rounds. In the i-th Round, you want to find the ball numbered i. To do so, you take out balls one by one, **without replacement**, till you take out the ball numbered i. Then the Round ends, and you place all the balls back into the bag and proceed with the next Round.

You need to find, for each Round, what is the expected number of balls you would need to remove.

Please note that during a single Round, you do not replace any balls you remove. However, every Round starts from scratch with all the balls put in.

Input

- The first line of each input contains a single integer **N**, the number of balls in the bag.
- The second line contains **N** integers, the i^{th} integer representing the volume of the i^{th} ball.

Output

Print **N** space separated values on a single line where the i^{th} value is the expected number of balls you remove in Round i. Your answer will be considered correct if the absolute error or relative error is less than 10^{-6} for each ball.

Constraints

- $1 \leq \mathbf{N} \leq 10^3$
- $1 \leq$ ball volumes $\leq 10^9$

Example

Input:
```
2
3 5
```

Output:
```
1.625000 1.375000
```

Explanation

Finding expected number of balls removed in Round 1: The probability of extracting the first ball in the first try is 3/8. Therefore, we will either extract 1 ball with probability 3/8 or two balls with probability 5/8. Expected number of balls extracted therefore is 3/8 + 2 * (5/8) = 1.625.

Similarly for the second ball, 5/8 + 2 * (3/8) = 1.375

Problem code: **LIGHTARE**
Problem name: **Find Lighted Area**

You are given a square with side length 2***L**. The coordinates of its vertices are at (-**L**, -**L**), (**L**, -**L**), (**L**, **L**) and (-**L**, **L**). You are also given **N** disks (each disk is opaque, and light rays cannot pass through them) each of radius **R**. The centers of each of the disks are at a fixed distance **D** from origin (0, 0). This distance **D** is strictly greater than **R**. It is guaranteed that all the disks lie entirely inside the square.

Suppose, there is a light source at the origin. Find the area of the region of the square that will be lit by this light source. A point inside the square will be lit if the line segment between the origin and the point don't intersect or touch any of the discs.

All distances mentioned are Euclidean distances.

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains four integers: **N**, **R, D, L**.

Each of the next **N** lines contain two space separated real numbers denoting the x and y coordinates of the center of the i-th disk. Each real number may contain up to 16 digits after the decimal point. These coordinates will satisfy all the conditions mentioned in the statement, i.e. they will be at distance exactly **D** from the origin.

Output

For each test case, output a real number representing the illuminated area. Your answer will be considered correct if the absolute or relative error of the answer doesn't exceed 10^{-6}.

Constraints

- $1 \leq \mathbf{T} \leq 10^5$
- $1 \leq \mathbf{N} \leq 10^5$
- $2 \leq \mathbf{D} \leq 10^5$
- $1 \leq \mathbf{R} < \mathbf{D}$
- $R + D < \mathbf{L} \leq 3 * 10^5$
- $-10^6 \leq \mathbf{x}, \mathbf{y} \leq 10^6$
- Sum of **N** over all the test cases in a single file would be at most $4 * 10^5$
- For any two centers of disks, it is guaranteed that the distance between them is $\geq 10^{-6}$.

Example

Input

```
4
1  1 2 4
2  0
2  1 2 4
2 0
-2 0
3 1 2 4
2 0
-2 0
0 2
4 1 2 4
2 0
-2 0
0 2
0 -2
```

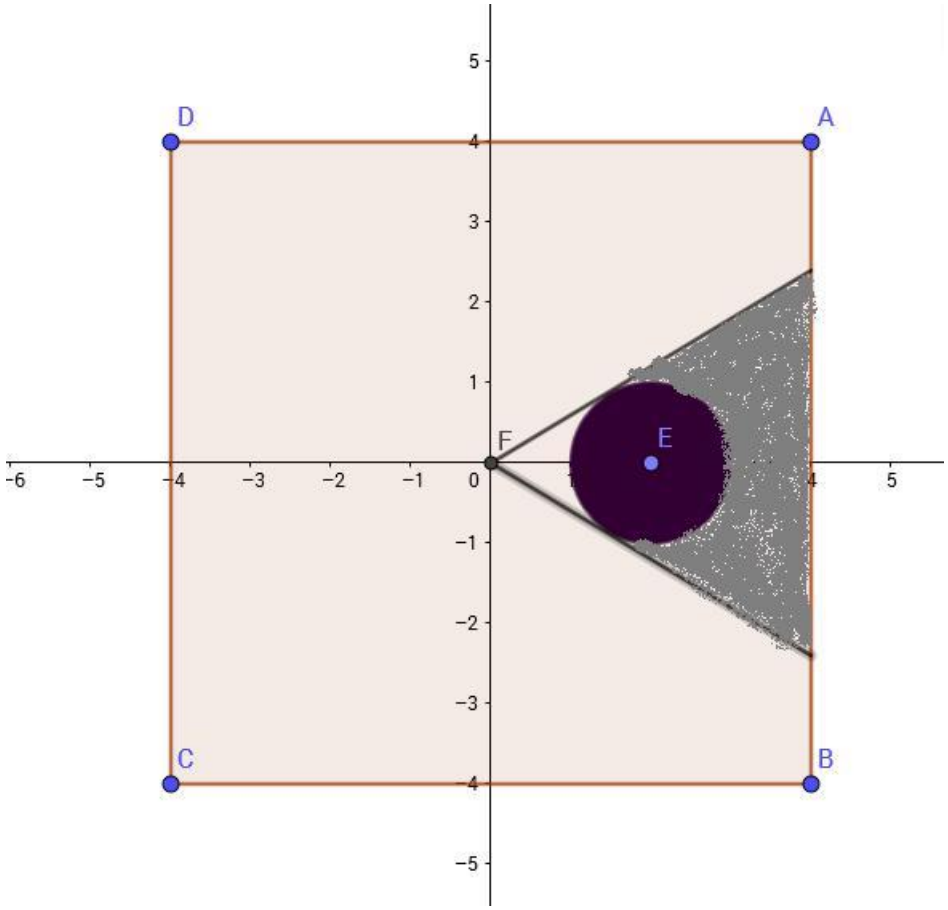
Output

```
55.4472489493382673
46.8944978986765346
38.3417468480148019
29.7889957973530693
```

Explanation

More examples are given in the webpage, but not printed here. Please check them as well.

The image below shows the scenario for testcase 1:



ABCD represents the square, which has coordinates according to **L** = 4. The only circle is at coordinates (2, 0). Notice that the distance of the center from the origin is 2, which is **L**. Its radius is **R** = 1. The light source is at (0, 0).

The grayed area along with the circle represents the area which is not lit by the light source. The area of the whole square with this area subtracted is the answer.

Problem code: **TALESQUA**
Problem name: **A Tale of Three Squares**

There are three squares, each with side length **a** placed on the x-axis. The coordinates of centers of these squares are (**x_1**, **a**/2), (**x_2**, **a**/2) and (**x_3**, **a**/2) respectively. All of them are placed with one of their sides resting on the x-axis.

You are allowed to move the centers of each of these squares along the x-axis (either to the left or to the right) by a distance of at most **K**. Find the maximum possible area of intersections of all these three squares that you can achieve. That is, the maximum area of the region which is part of all the three squares in the final configuration.

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 line of each test case contains two space-separated integers **a, K** denoting side length of the squares, and the maximum distance that you can move the center of any square.
- The second line contains three space separated integers **x_1**, **x_2**, **x_3**

Output

For each test case, output a real number corresponding to the maximum area of the intersection of the three squares that you can obtain. Your answer will be considered correct if it has an absolute error of less than or equal to 10^{-2}.

Constraints

- $1 \leq \mathbf{T} \leq 10^5$
- $1 \leq \mathbf{a} \leq 10^5$
- $0 \leq \mathbf{K} \leq 10^6$
- $-10^6 \leq \mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3} \leq 10^6$

Example

Input

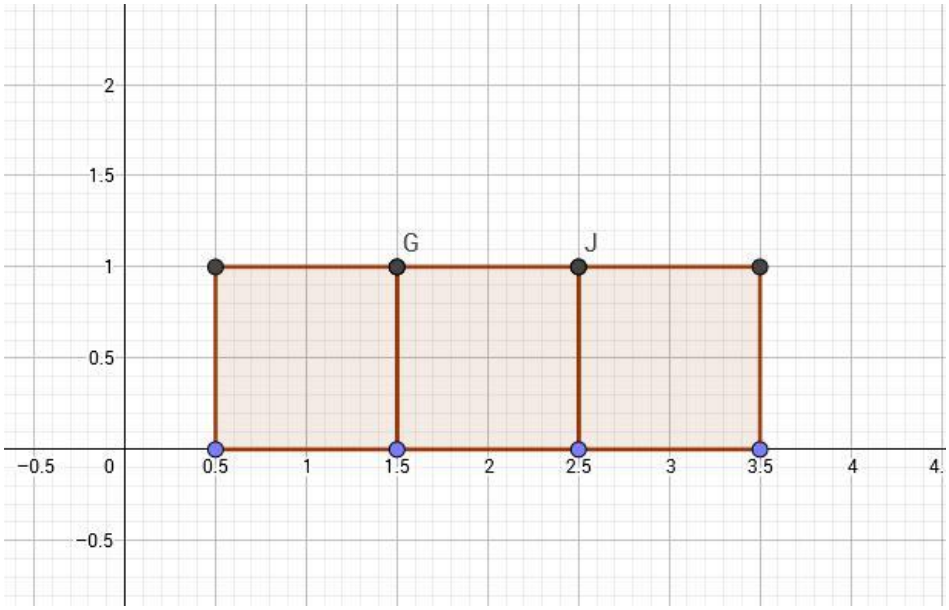
```
3
1 0
1 2 3
1 1
1 2 3
1 1
1 4 6
```

Output

```
0.000000
1.0000
0.0
```

Explanation

Testcase 1: The figure below shows the three squares:



Since **K** = 0, they cannot be moved, and since there is no region which belongs to all three squares, the answer is 0.

Testcase 2: The starting configuration is the same as above, but now each of the squares can move 1 unit. So we can move the first square 1 unit to the right and the third square one unit to the left, and have all the three squares at x-coordinate = 2. Thus the entire square is part of all three squares, and the answer is 1.

Problem code: **TALETRI**
Problem name: **A Tale of Two Right Angled Triangles**

You are given two right angled triangles. You are given their side lengths. The first triangle is ABC, where side AB is the hypotenuse. Length of AB is **c**, that of AC is **b**, and that of BC is **a**. Similarly, the second triangle is DEF, where DE is the hypotenuse. Length of DE is **f**, that of DF is **e**, and that of EF is **d**.

First we define *minimum area enclosing axis-parallel rectangle* of some figures. It will be a rectangle which is axis parallel (i.e. each of its sides should be parallel to x or y axis) that encloses all the figures and has the minimum possible area. By enclosing, we mean that all the figures should be inside the rectangle.

You have to find a set of coordinates to assign to the points A, B, C, D, E, F such that the *minimum area enclosing axis-parallel rectangle* of the two triangles has sides **L** and **R**, i.e. one side is of length **L**, and the other is of length **R**. Also, each of the sides AC, BC, DF, EF should be parallel to either x or y axis. If it is not possible to assign the coordinates, output -1.

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 only line of each test case contains eight space separated integers denoting the lengths **a, b, c, d, e, f, L, R**.

Output

For each test case, either output -1 if it is impossible to assign the coordinates. Otherwise, output six lines each containing two space separated integers corresponding to the coordinates of points A, B, C, D, E, F in that order. You can output any possible coordinates that have their absolute values less than or equal to 10^9.

Constraints

- $1 \leq \mathbf{T} \leq 10^5$
- $1 \leq \mathbf{a, b, c, d, e, f, L, R} \leq 10^8$
- $\mathbf{a^2 + b^2 = c^2}$
- $\mathbf{d^2 + e^2 = f^2}$

Example

Input

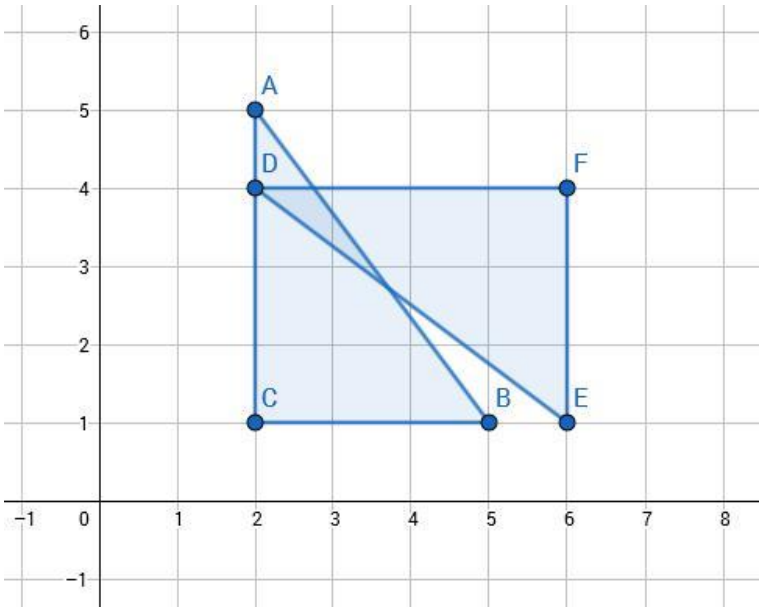
```
2
3 4 5 3 4 5 4 4
3 4 5 3 4 5 2 4
```

Output

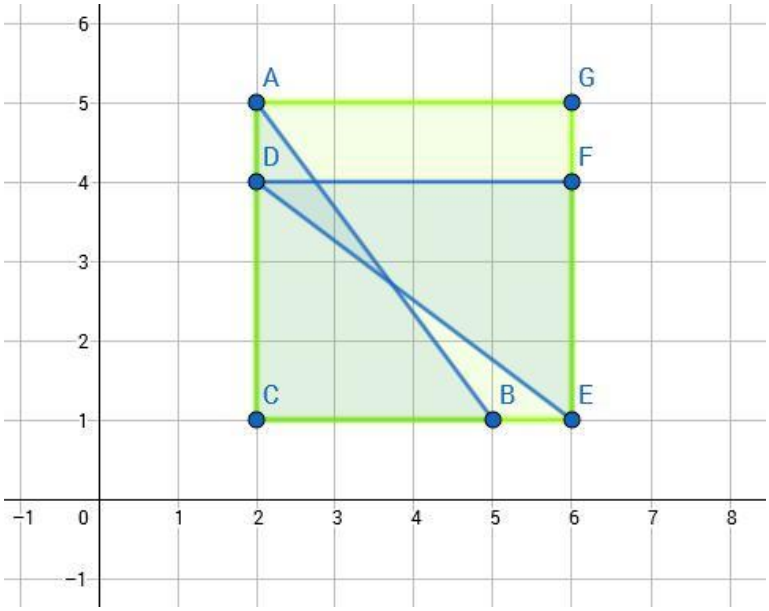
```
2 5
5 1
2 1
2 4
6 1
6 4
-1
```

Explanation

Testcase 1: The output corresponds to the image shown below:



Notice that all the 6 side lengths are maintained, and the sides AC, BC, DF, EF are all axis-parallel. The *minimum area enclosing axis-parallel rectangle* of these two triangles is shown in the next image:



Notice that this is a rectangle with side lengths 4 and 4, just as we wanted. Hence this is a valid output.

Problem code: **ZUBWALLT**
Problem name: **Organize The Wallet**

Ranju has many currency notes in his wallet. There are notes of denominations **10, 20, 50, 100, 200, 500**, and **2000** rupees in his wallet right now, but the notes are all messed up. He wants to organize his wallet so that all the notes of each denomination are together. **The notes don't necessarily need to be in sorted order.**

In each step, Ranju can take a note and insert this note at any place among the remaining notes. Please determine the minimum number of steps he requires to achieve his goal.

Input

The first line of the input is the number of test cases, **T**. Description of each test case is given below.

The first line of each testcase contains a single integer, **N**, the number of notes in the wallet.

The next line contains **N** integers: $\mathbf{A_1, A_2, A_3, \ldots, A_N}$ which represent the notes in the wallet at the initial state.

Output

For each test case, print the minimum number of steps required to organize the wallet, in a new line.

Constraints

- $\mathbf{1 \leq T \leq 5}$
- $\mathbf{1 \leq N \leq 10^5}$
- Each of $\mathbf{A_i}$ will be from the set **{10, 20, 50, 100, 200, 500, 2000}**

Example

Input:
```
2
6
500 500 100 200 200 100
9
100 200 200 500 100 100 500 100 100
```

Output:
```
1
2
```

Explanation

Testcase 1: We can move the last 100 rupee note and insert it to the right of 1st 100 rupee note. The result would be 500 500 100 100 200 200.

Testcase 2: One possible way would be the following:

- In the first step, we can move the first note, which has denomination 100 next to 7th note, which has denomination 500. The result will be : 200 200 500 100 100 500 100 100 100.
- In the second step, we can move the 6th note, which has denomination 500 to the right of the 3rd note, which has denomination 500. The result will be 200 200 500 500 100 100 100 100 100