Problem A. Tower

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Prof. Pang built n block towers with different heights. The i-th tower has height a_i.

Prof. Shou doesn't like these towers because of their arbitrary heights. He decides to **first remove exactly m of them**, and then perform some (or none) of the following operations:

- Choose a tower and increase its height a_i by 1.
- Choose a tower and decrease its height a_i by 1.
- Choose a tower and divide its height a_i by 2. If the new height is not an integer, it is rounded down.

Prof. Shou can never choose a removed tower. If after an operation, the height of a tower will become 0, that operation is not allowed. Under these constraints, Prof. Shou can perform an arbitrary number of operations in arbitrary order.

Prof. Shou would like all the towers that are not removed to have the same heights. Please calculate the minimum number of operations to achieve this.

Input

The first line contains one integer T ($1 \le T \le 10$), the number of test cases.

For each test case, the first line contains two integers n, m ($1 \le n \le 500, 0 \le m < n$), the number of towers, and the number of towers Prof. Shou should delete before performing the operations.

The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$), the initial heights of the towers.

Output

For each test case, output the minimum number of operations in one line.

Example

standard input	standard output
3 2 0 2 6 5 0 1 2 3 4 5 5 3 1 2 3 4 5	2 4 1

Problem B. Torch

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

Prof. Pang and Prof. Shou go to explore a cave together. Prof. Pang walks ahead of Prof. Shou.

Each of them has a torch for illumination. The torches need fuel to burn. Prof. Pang's torch can burn for a_1 seconds once it has been refilled, and it takes b_1 seconds to refuel the torch after it burns out. Prof. Shou's torch can burn for a_2 seconds once it has been refilled, and it takes b_2 seconds to refuel the torch after it burns out. The person who is refueling the torch cannot walk simultaneously. For safety reasons, they cannot refuel the torch until the fuel runs out.

Because Prof. Pang is too fat and the cave is too narrow, Prof. Shou cannot surpass Prof. Pang during the exploration, which means that Prof. Shou is at least 1 unit behind Prof. Pang.

Each of them can walk forward a distance of 1 unit per second when his torch is burning. Every second, Prof. Pang moves first, then Prof. Shou does. In order to get to their destination earlier, they will move as long as they can walk forward.

Now Prof. Shou has n questions, and for the i-th question, he wants to know that at time q_i, how many units of the distance he has moved forward from the starting point? Prof. Shou starts 1 unit behind Prof. Pang. The initial time is 0. Both Prof. Pang and Prof. Shou refueled the torch before the initial time.

Input

The first line contains one integer T ($1 \le T \le 10^5$), the number of test cases.

For each test case, the first line contains 5 integers a_1, b_1, a_2, b_2, n ($1 \le a_1, b_1, a_2, b_2, n \le 10^6$) denoting the time Prof. Pang's torch can burn, the time for Prof. Pang to refuel his torch, the time Prof. Shou's torch can burn, the time for Prof. Shou to refuel his torch, and the number of Prof. Shou's queries. Each of the next n lines describes a query. Query i is denoted by one integer q_i ($1 \le q_i \le 10^{16}$).

It is guaranteed that over all test cases, each of the following numbers is no more than 10^6:

- the sum of a_1,
- the sum of a_2,
- the sum of b_1,
- the sum of b_2,
- and the sum of n.

Output

For each query, print one line containing the answer – the number of units that Prof. Shou has walked forward from the starting point.

Example

standard input	standard output
3 2 3 2 4 2 7 8 1 1 1 1 2 3 4 9 7 10 3 5 5 10 20 30 50	3 4 2 2 5 9 13 18 28

Problem C. DFS Order 2

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

Prof. Pang has a rooted tree that is rooted at vertex 1 and has n nodes. These n nodes are numbered from 1 to n.

Now he wants to start the depth-first search at the root. He wonders for each node v, how many ways it can appear in the j-th position of **depth-first search order**. The depth-first search order is the order of nodes visited during the depth-first search. A node appears in the j-th ($1 \le j \le n$) position in this order means it is visited after $j-1$ other nodes. Because sons of a node can be iterated in arbitrary order, multiple possible depth-first orders exist.

Prof. Pang wants to know for each node v, how many different **depth-first search order**s such that v appears in the j-th position. For each v, j ($1 \le v, j \le n$), compute the answer. Because the answer can be very large, output it modulo 998244353.

Following is a pseudo-code for the depth-first search on a rooted tree. After calling MAIN(), `dfs_order` is the depth-first search order.

Algorithm 1 An implementation of depth-first search

```
1:  procedure DFS(vertex x)
2:      Append x to the end of dfs_order
3:      for each son y of x do                ▷ Sons can be iterated in arbitrary order.
4:          DFS(y)
5:      end for
6:  end procedure
7:  procedure MAIN()
8:      Let dfs_order be a global variable
9:      dfs_order ← empty list
10:     DFS(1)
11: end procedure
```

Input

The first line contains one integer n ($1 \le n \le 500$), the number of vertices in the tree.

Each of the next $n-1$ lines describes an edge of the tree. Edge i is denoted by two integers u_i and v_i, the labels of vertices it connects ($1 \le u_i, v_i \le n, u_i \ne v_i$).

It is guaranteed that the given edges form a tree.

Output

For each vertex v from 1 to n, output one line containing n integers modulo 998244353. The j-th integer in the v-th line should be the number of different depth-first search orders such that v appears in the j-th position.

Example

standard input	standard output
5 1 2 1 3 3 4 3 5	4 0 0 0 0 0 2 0 0 2 0 2 2 0 0 0 0 1 2 1 0 0 1 2 1

Problem D. Frozen Scoreboard

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

There was an ICPC contest two thousand years ago in the Qin dynasty. There were m problems and n teams in the contest. We only know how many problems each team solved and how much total time they used from the historical records. These are called the **final result**s of the teams. We don't know which problems they solved or their submission times.

Recently, we seem to had a discovery. We found the **frozen scoreboard** of the teams. From the frozen scoreboard of a team, we know their submissions during the whole contest, but we don't know the verdicts of the submissions in the last hour. And some people found that for some teams, their frozen scoreboards may contradict their final results in the historical records.

Given the final results and the frozen scoreboards of the teams, please construct a **final scoreboard** for each team that is consistent with both its final result and its frozen scoreboard.

From the submissions during the contest, we can calculate the final scoreboard and the final result as follows:

For a fixed team i, its **final scoreboard** is an array of m elements where the j-th element shows some information about team i's submissions on problem j.

- If team i didn't submit to problem j, the cell should be a single character "." (without quotes).
- If team i submitted x times to problem j and none of the submissions was accepted, the cell should contain $-\ x$.
- Otherwise, consider all submissions from team i to problem j. Each submission has a submission time. Suppose the earliest accepted submission is the x-th one. Then the cell should contain $+\ x/y$ where y ($0 \le y \le 299$) is the submission time of the x-th submission. y is an integer representing the submission time in minutes.

Note that in the final scoreboard, we don't care about submissions after the first accepted one. It is possible that two or more submissions happened in the same minute.

The **final result** of a team is computed from its **final scoreboard**. For each team, we can calculate the number of problems it solved. This number is equal to the number of "+" in the team's final scoreboard.

We can also calculate its total time. If team i solved problem j in the y-th minute after $x-1$ unaccepted submissions (in other words, the j-th cell in the i-th row of the final scoreboard is $+\ x/y$), problem j contributes $20(x-1)+y$ time to team i. If team i didn't solve problem j, problem j contributes 0 time to team i, no matter team i submitted to problem j or not. The total time of team i is the sum of contributions of each problem.

The rules for the **frozen scoreboard** will be introduced in the input section. We will distinguish submissions in the final hour and other submissions. A submission was in the final hour if its submission time is between 240 and 299.

Input

The first line contains two integers n, m ($1 \le n \le 1000, 1 \le m \le 13$), the number of teams in the contest, and the number of problems in the contest.

Then there are n blocks describing the **final result** and the **frozen scoreboard** of each team.

The i-th block represents team i. In the i-th block, the first line contains two integers a_i, b_i ($0 \le a_i \le m, 0 \le b_i \le 10^5$), the number of problems team i solved **during the whole contest** and

the total time of team i for solving the a_i problems. These two numbers represent the final result of the contest. The next m lines describe the status of team i in the frozen scoreboard. For each $1 \le j \le m$,

- If the j-th line is $+\ x/y$ ($1 \le x \le 100, 0 \le y \le 239$), team i solved problem j at time y and the accepted solution is their x-th submission on problem j.
- If the j-th line is $?\ x\ y$ ($1 \le x \le y \le 100$), team i didn't solve the problem j in the first four hours. Team i submitted problem j for y times in which x submissions are in the last hour. Note that submissions made in the last hour after the accepted one will count in the **frozen scoreboard**, but not in the **final scoreboard**.
- If the j-th line is $-\ x$, team i didn't solve the problem j in the first four hours. Team i submitted problem j for x ($1 \le x \le 100$) times before the last hour and did not submit problem j in the last hour.
- If the j-th line is a single character "." (without quotes), team i didn't submit problem j at all.

Output

For each team i, if its final result contradicts its frozen scoreboard, output `No` in one line. Otherwise, output `Yes` in the first line and then output m lines. The j-th line should contain

- $+\ x/y$ ($1 \le x \le 100, 0 \le y \le 299$), if the x-th submission from team i to problem j is accepted and is in the y-th minute of the contest. All submissions from team i to team j before the x-th one was not accepted. Please don't output extra spaces before and after slash "/".
- $-\ x$ ($1 \le x \le 100$), if team i submitted to problem j for x times and none of the submissions was accepted.
- . if team i didn't submit to problem j at all.

Please note that in the input and the output, there is always a space following each ?, +, and −.

Examples

standard input	standard output
1 13 7 951 + 1/6 ? 3 4 + 4/183 - 2 + 3/217 . . . + 2/29 + 1/91 . + 1/22 .	Yes + 1/6 + 2/263 + 4/183 - 2 + 3/217 . . . + 2/29 + 1/91 . + 1/22 .
6 2 1 100 . ? 3 4 2 100 + 1/1 + 1/2 0 0 - 5 - 6 2 480 ? 100 100 ? 100 100 2 480 ? 99 100 ? 100 100 1 2000 ? 100 100 ? 100 100	No No Yes - 5 - 6 Yes + 1/240 + 1/240 No Yes + 87/280 - 100

Note

Here is an example of the frozen scoreboard in the first sample.

Problem E. Identical Parity

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Let the value of a sequence be the sum of all numbers in it.

Determine whether there exists a permutation of length n such that the values of all subsegments of length k of the permutation share the same parity. The values share the same parity means that they are all odd numbers or they are all even numbers.

A subsegment of a permutation is a contiguous subsequence of that permutation. A permutation of length n is a sequence in which each integer from 1 to n appears exactly once.

Input

The first line contains one integer T ($1 \le T \le 10^5$), the number of test cases.

For each test case, the only line contains two integers n, k ($1 \le k \le n \le 10^9$).

Output

For each test case, output "`Yes`" (without quotes) if there exists a valid permutation, or "`No`" (without quotes) otherwise.

You can output "`Yes`" and "`No`" in any case (for example, strings "`YES`", "`yEs`" and "`yes`" will be recognized as positive responses).

Example

standard input	standard output
3 3 1 4 2 5 3	No Yes Yes

Note

In the first test case, it can be shown that there does not exist any valid permutation.

In the second test case, $[1, 2, 3, 4]$ is one of the valid permutations. Its subsegments of length 2 are $[1, 2], [2, 3], [3, 4]$. Their values are $3, 5, 7$, respectively. They share the same parity.

In the third test case, $[1, 2, 3, 5, 4]$ is one of the valid permutations. Its subsegments of length 3 are $[1, 2, 3], [2, 3, 5], [3, 5, 4]$. Their values are $6, 10, 12$, respectively. They share the same parity.

Problem F. Grid Points

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are given a simple polygon in the first quadrant of the Cartesian plane. This means that the coordinate (x, y) of any point in the polygon satisfies $x > 0$ and $y > 0$.

Consider all grid points in the polygon. Order them in increasing order of **slopes**. Output the k-th grid point in this order.

A grid point is a point (x, y) such that x and y are integers. A point on the boundary of a polygon is considered to be in the polygon. The slope of point (x, y) is y/x. If two points have the same slope, they are ordered lexicographically first by x, then by y. In other words, a point (x_1, y_1) is lexicographically less than (x_2, y_2) if $(x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 < y_2)$.

Input

The first line contains one integer T ($1 \le T \le 500$), the number of test cases.

For each test case, the first line contains two integers n, k ($n \ge 3, 1 \le k$). Each of the next n lines describes a vertex of the polygon. Each vertex is denoted by two integers x, y ($1 \le x, y \le 10^9$), representing its coordinate (x, y). The vertices are given in counterclockwise order.

It is guaranteed that the polygon is simple, i.e. the vertices are distinct, two edges may overlap only when they are consecutive on the boundary, and the overlap contains exactly 1 point. It is guaranteed that k is no more than the number of grid points in the polygon.

It is guaranteed that the sum of n over all test cases is no more than 2000.

Output

For each test case, output two integers x, y in one line representing the coordinate (x, y) of the k-th grid point.

Example

standard input	standard output
4 3 3 1 1 3 1 3 3 4 500000000000000000 1 1 1000000000 1 1000000000 1000000000 1 1000000000 9 22 9 6 6 7 9 7 10 10 6 9 3 9 1 6 1 5 7 3 5 22447972861454999 270353376 593874603 230208698 598303091 237630296 255016434 782669452 568066304 654623868 958264153	3 2 500000000 500000000 7 8 730715389 644702744

Problem G. Quick Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

When Prof. Pang was young, he wrote the following code for quick sort. Please calculate how many swaps are performed when calling QUICKSORT$(A, 1, n)$. A is a given permutation with length n.

Algorithm 2 An implementation of quick sort

```
1: procedure QUICKSORT(A, lo, hi)
2:     if lo ≥ 0 and hi ≥ 0 and lo < hi then
3:         p ← PARTITION(A, lo, hi)
4:         QUICKSORT(A, lo, p)
5:         QUICKSORT(A, p + 1, hi)
6:     end if
7: end procedure
8: procedure PARTITION(A, lo, hi)
9:     pivot ← A[floor((hi + lo)/2)]
10:    i ← lo − 1
11:    j ← hi + 1
12:    while True do
13:        repeat
14:            i ← i + 1
15:        until A[i] ≥ pivot
16:        repeat
17:            j ← j − 1
18:        until A[j] ≤ pivot
19:        if i ≥ j then
20:            return j
21:        end if
22:        Swap A[i] with A[j]
23:    end while
24: end procedure
```

Input

The first line contains one integer T ($1 \le T \le 10^5$), the number of test cases.

For each test case, the first line contains one positive integer n ($1 \le n \le 5 \times 10^5$). The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$) denoting the permutation A. It is guaranteed that $a_1, \ldots, a_n$ form a permutation, i.e. $a_i \ne a_j$ for $i \ne j$.

It is guaranteed that the sum of n over all test cases is no more than 5×10^5.

Output

For each test case, output one line containing the number of swaps performed when calling `quicksort`$(A, 1, n)$.

Example

standard input	standard output
3 3 3 2 1 5 2 4 5 3 1 10 7 2 4 6 1 9 10 8 5 3	1 4 7

Problem H. Set of Intervals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Prof. Pang has a multi-set of intervals $S = \{[l_i, r_i]\}(l_i < r_i)$.

Prof. Pang will perform the following operation for $|S| - 1$ times:

- Select two intervals $[a, b]$ and $[c, d]$ from S, and then choose two integers x, y satisfying $x \in [a, b], y \in [c, d], x < y$. After that, delete $[a, b]$ and $[c, d]$ from S, and add $[x, y]$ to S.

It's easy to find that S contains exactly one interval after the operations, and Prof. Pang will get the interval as a gift.

Now Prof. Pang wants you to calculate how many different intervals he can get.

Input

The first line contains one integer T ($1 \le T \le 10^4$), the number of test cases.

For each test case, the first line contains one integer n ($1 \le n \le 10^5$) — the size of S. Each of the following n lines contains two integers l_i and r_i ($1 \le l_i < r_i \le 10^9$), describing the i-th interval in S.

It is guaranteed that the sum of n over all test cases is no more than 10^5.

Output

For each test case, output one line containing the answer to Prof. Pang's question.

Example

standard input	standard output
4 1 1 1000000000 2 1 1000000000 1 1000000000 4 1 2 3 4 5 6 7 8 4 1 3 2 4 5 8 6 7	1 499999999500000000 26 28

Problem I. Shortest Path

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

You are given an undirected weighted graph G with vertices $1, 2, \ldots, n$. Please output the sum of the answers to the following x questions:

- The i-th question ($1 \le i \le x$): What is the minimum length of path that starts at vertex 1, ends at vertex n, and contains exactly i edges?

For each question, if such a path does not exist, the answer is considered to be 0. A path may use one edge multiple times. Output the answer modulo 998244353.

Input

The first line contains one integer T ($1 \le T \le 2000$), the number of test cases.

For each test case, the first line contains three integers n, m, x ($1 \le n \le 2000, 0 \le m \le 5000, 1 \le x \le 10^9$). Each of the next m lines describes an edge of the graph. Edge i is denoted by three integers a_i, b_i, w_i ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^9$), the labels of vertices it connects and its weight. Note that self-loops and parallel edges may exist.

It is guaranteed that the sum of n over all test cases is no more than 2000 and the sum of m over all test cases is no more than 5000.

Output

For each test case, output one integer modulo 998244353 denoting the answer.

Example

standard input	standard output
4 3 2 10 1 2 5 2 3 4 3 0 1000000000 3 3 100 1 2 3 1 3 4 2 3 5 4 6 1000000000 1 2 244 1 2 325 1 4 927 3 3 248 2 4 834 3 4 285	125 0 15300 840659991

Problem J. Skills

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Prof. Pang has 3 different skills to practice, including soda drinking, fox hunting, and stock investing. We call them Skill 1, Skill 2, and Skill 3. In each of the following n days, Prof. Pang can choose one of the three skills to practice. In the i-th day ($1 \le i \le n$), if Prof. Pang chooses Skill j ($1 \le j \le 3$) to practice, his level of Skill j will increase by $a_{i,j}$. Initially, Prof. Pang's levels of all skills are 0.

Prof. Pang forgets skills if he does not practice. At the end of each day, if he has not practiced Skill j for k days, his level of Skill j will decrease by k. For example, if he practices Skill 1 on day 1 and Skill 2 on day 2, at the end of day 2, he has not practiced Skill 1 for 1 day and has not practiced Skill 3 for 2 days. Then his levels of Skill 1 and Skill 3 will decrease by 1 and 2, respectively. His level of Skill 2 does not decrease at the end of day 2 because he practices Skill 2 on that day. In this example, we also know that his levels of Skill 2 and Skill 3 both decrease by 1 at the end of day 1.

Prof. Pang's level of any skill will not decrease below 0. For example, if his level of some skill is 3 and at the end of some day, this level is decreased by 4, it will become 0 instead of -1.

Prof. Pang values all skills equally. Thus, he wants to maximize the sum of his three skill levels after the end of day n.

Given $a_{i,j}$ ($1 \le i \le n, 1 \le j \le 3$), find the maximum sum.

Input

The first line contains a single integer T ($1 \le T \le 1000$) denoting the number of test cases.

For each test case, the first line contains an integer n ($1 \le n \le 1000$). The $(i+1)$-th line contains three integers $a_{i,1}, a_{i,2}, a_{i,3}$ ($0 \le a_{i,j} \le 10000$ for any $1 \le i \le n, 1 \le j \le 3$).

It is guaranteed that the sum of n over all test cases is no more than 1000.

Output

For each test case, output the maximum possible sum of skill levels in one line.

Example

standard input	standard output
2 3 1 1 10 1 10 1 10 1 1 5 1 2 3 6 5 4 7 8 9 12 11 10 13 14 15	26 41

Problem K. Stack Sort

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

You are given a permutation with n numbers, $a_1, a_2, \ldots, a_n (1 \le a_i \le n, a_i \ne a_j$ when $i \ne j)$.

You want to sort these numbers using m stacks. Specifically, you should complete the following task:

Initially, all stacks are empty. You need to push each number a_i to the top of one of the m stacks one by one, in the order of $a_1, a_2, \ldots, a_n$. **After pushing all numbers in the stacks**, you pop all the elements from the stacks in a clever order so that the first number you pop is 1, the second number you pop is 2, and so on. **If you pop an element from a stack S, you cannot pop any element from the other stacks until S becomes empty.**

What is the minimum possible m to complete the task?

Input

The first line contains one integer T ($1 \le T \le 10^5$), the number of test cases.

For each test case, the first line contains one positive integer n ($1 \le n \le 5 \times 10^5$). The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$) denoting the permutation. It is guaranteed that $a_1, \ldots, a_n$ form a permutation, i.e. $a_i \ne a_j$ for $i \ne j$.

It is guaranteed that the sum of n over all test cases is no more than 5×10^5.

Output

For each test case, output the minimum possible m in one line.

Example

standard input	standard output
3 3 1 2 3 3 3 2 1 5 1 4 2 5 3	3 1 4

Problem L. Tree Distance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

You are given an unrooted weighted tree T with vertices $1, 2, \ldots, n$. Please answer some queries.

We define $\texttt{dist}(i, j)$ as the distance between vertex i and vertex j in T.

For each query, you are given two integers l, r. Please answer the value of

$$\min_{l \le i < j \le r} (\texttt{dist}(i, j)).$$

Input

The first line contains one integer n ($1 \le n \le 2 \times 10^5$), the number of vertices in the tree.

Each of the next $n - 1$ lines describes an edge of the tree. Edge i is denoted by three integers a_i, b_i, w_i ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^9$), the labels of vertices it connects and its weight.

Then one line contains one integer q ($1 \le q \le 10^6$), the number of queries.

Each of the following q lines contains two integers l, r ($1 \le l \le r \le n$) describing a query.

It is guaranteed that the given edges form a tree.

Output

For each query, output the answer in one line. If there is no i, j such that $1 \le i < j \le r$, the answer is -1.

Example

standard input	standard output
5 1 2 5 1 3 3 1 4 4 3 5 2 5 1 1 1 4 2 4 3 4 2 5	-1 3 7 7 2

Problem M. Best Carry Player

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Prof. Pang is given n numbers $a_1, \ldots, a_n$. It is easy to add the numbers up using a computer. But Prof. Pang treasures his computer so much and wants to reduce its workload. He decides to simulate the following program by hand.

Algorithm 3 Sum of elements

```
1: s ← 0
2: for i from 1 to n do
3:     s ← s + a[i]
4: end for
```

Unlike a computer, the time needed for Prof. Pang to simulate the program is proportional to the total number of **carries**[1] when calculating $s+a[i]$ for each i from 1 to n. Prof. Pang adds numbers **by column addition in base-ten**, just like what we normally do in primary school. For example, there are two carries in the following addition.

carry	*1*		*1*	
		6	7	6
+		5	1	8
	1	1	9	4

Please permute the array $a_1, \ldots, a_n$ so that the total number of carries when Prof. Pang simulates the program is as small as possible. (By "permute an array", we mean that you can change the order of the elements arbitrarily.)

Input

The first line contains one integer T ($1 \le T \le 10^5$), the number of test cases.

For each test case, the first line contains one positive integer n ($1 \le n \le 10^5$). The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$) denoting the numbers Prof. Pang is given.

It is guaranteed that the sum of n over all test cases is no more than 10^5.

Output

For each test case, output one line containing the minimum amount of carries.

Example

standard input	standard output
2 3 9 99 999 1 12345	5 0

[1] which means "进位" in Chinese