A. Favorite Sequence

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a favorite sequence $a[1 \dots n]$ consisting of n integers. He wrote it out on the whiteboard as follows:

- he wrote the number a_1 to the left side (at the beginning of the whiteboard);
- he wrote the number a_2 to the right side (at the end of the whiteboard);
- then as far to the left as possible (but to the right from a_1), he wrote the number a_3;
- then as far to the right as possible (but to the left from a_2), he wrote the number a_4;
- Polycarp continued to act as well, until he wrote out the entire sequence on the whiteboard.

The beginning of the result looks like this (of course, if $n \geq 4$).

For example, if $n = 7$ and $a = [3, 1, 4, 1, 5, 9, 2]$, then Polycarp will write a sequence on the whiteboard $[3, 4, 5, 2, 9, 1, 1]$.

You saw the sequence written on the whiteboard and now you want to restore Polycarp's favorite sequence.

Input

The first line contains a single positive integer t $(1 \leq t \leq 300)$ — the number of test cases in the test. Then t test cases follow.

The first line of each test case contains an integer n $(1 \leq n \leq 300)$ — the length of the sequence written on the whiteboard.

The next line contains n integers $b_1, b_2, \dots, b_n$ $(1 \leq b_i \leq 10^9)$ — the sequence written on the whiteboard.

Output

Output t answers to the test cases. Each answer — is a sequence a that Polycarp wrote out on the whiteboard.

Example

input

```
6
7
3 4 5 2 9 1 1
4
9 2 7 1
11
8 4 3 1 2 7 8 7 9 4 2
1
42
2
11 7
8
1 1 1 1 1 1 1 1
```

output

```
3 1 4 1 5 9 2
9 1 2 7
8 2 4 4 3 9 1 7 2 8 7
42
11 7
1 1 1 1 1 1 1 1
```

Note

In the first test case, the sequence a matches the sequence from the statement. The whiteboard states after each step look like this:

$$[3] \Rightarrow [3, 1] \Rightarrow [3, 4, 1] \Rightarrow [3, 4, 1, 1] \Rightarrow [3, 4, 5, 1, 1] \Rightarrow [3, 4, 5, 9, 1, 1] \Rightarrow [3, 4, 5, 2, 9, 1, 1].$$

B. Last Year's Substring

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a string $s[1 \dots n]$ of length n consisting of decimal digits. Polycarp performs the following operation with the string s **no more than once** (i.e. he can perform operation 0 or 1 time):

- Polycarp selects two numbers i and j ($1 \le i \le j \le n$) and removes characters from the s string at the positions $i, i+1, i+2, \dots, j$ (i.e. removes substring $s[i \dots j]$). More formally, Polycarp turns the string s into the string $s_1 s_2 \dots s_{i-1} s_{j+1} s_{j+2} \dots s_n$.

For example, the string $s =$"20192020" Polycarp can turn into strings:

- "2020" (in this case $(i,j) = (3,6)$ or $(i,j) = (1,4)$);
- "2019220" (in this case $(i,j) = (6,6)$);
- "020" (in this case $(i,j) = (1,5)$);
- other operations are also possible, only a few of them are listed above.

Polycarp likes the string "2020" very much, so he is wondering if it is possible to turn the string s into a string "2020" in no more than one operation? Note that you can perform zero operations.

Input

The first line contains a positive integer t ($1 \le t \le 1000$) — number of test cases in the test. Then t test cases follow.

The first line of each test case contains an integer n ($4 \le n \le 200$) — length of the string s. The next line contains a string s of length n consisting of decimal digits. It is allowed that the string s starts with digit `0`.

Output

For each test case, output on a separate line:

- "YES" if Polycarp can turn the string s into a string "2020" in no more than one operation (i.e. he can perform 0 or 1 operation);
- "NO" otherwise.

You may print every letter of "YES" and "NO" in any case you want (so, for example, the strings yEs, yes, Yes and YES will all be recognized as positive answer).

Example

input

```
6
8
20192020
8
22019020
4
2020
5
20002
6
729040
6
200200
```

output

```
YES
YES
YES
NO
NO
NO
```

Note

In the first test case, Polycarp could choose $i = 3$ and $j = 6$.

In the second test case, Polycarp could choose $i = 2$ and $j = 5$.

In the third test case, Polycarp did not perform any operations with the string.

C. Unique Number

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a positive number x. Find the smallest positive integer number that has the sum of digits equal to x and all digits are **distinct** (unique).

Input

The first line contains a single positive integer t ($1 \le t \le 50$) — the number of test cases in the test. Then t test cases follow.

Each test case consists of a single integer number x ($1 \le x \le 50$).

Output

Output t answers to the test cases:

- if a positive integer number with the sum of digits equal to x and all digits are different exists, print the smallest such number;
- otherwise print -1.

Example

input

```
4
1
5
15
50
```

output

```
1
5
69
-1
```

D. Add to Neighbour and Remove

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp was given an array of $a[1 \dots n]$ of n integers. He can perform the following operation with the array a no more than n times:

- Polycarp selects the index i and adds the value a_i to **one of his choice** of its neighbors. More formally, Polycarp adds the value of a_i to a_{i-1} or to a_{i+1} (if such a neighbor does not exist, then it is impossible to add to it).
- After adding it, Polycarp removes the i-th element from the a array. During this step the length of a is decreased by 1.

The two items above together denote one single operation.

For example, if Polycarp has an array $a = [3, 1, 6, 6, 2]$, then it can perform the following sequence of operations with it:

- Polycarp selects $i = 2$ and adds the value a_i to $(i-1)$-th element: $a = [4, 6, 6, 2]$.
- Polycarp selects $i = 1$ and adds the value a_i to $(i+1)$-th element: $a = [10, 6, 2]$.
- Polycarp selects $i = 3$ and adds the value a_i to $(i-1)$-th element: $a = [10, 8]$.
- Polycarp selects $i = 2$ and adds the value a_i to $(i-1)$-th element: $a = [18]$.

Note that Polycarp could stop performing operations at any time.

Polycarp wondered how many minimum operations he would need to perform to make all the elements of a equal (i.e., he wants all a_i are equal to each other).

Input

The first line contains a single integer t ($1 \le t \le 3000$) — the number of test cases in the test. Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 3000$) — the length of the array. The next line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$) — array a.

It is guaranteed that the sum of n over all test cases does not exceed 3000.

Output

For each test case, output a single number — the minimum number of operations that Polycarp needs to perform so that all elements of the a array are the same (equal).

Example

input

```
4
5
3 1 6 6 2
4
1 2 2 1
3
2 2 2
4
6 3 2 1
```

output

```
4
2
0
2
```

Note

In the first test case of the example, the answer can be constructed like this (just one way among many other ways):

$[3, 1, 6, 6, 2] \xrightarrow{i=4,\ add\ to\ left} [3, 1, 12, 2] \xrightarrow{i=2,\ add\ to\ right} [3, 13, 2] \xrightarrow{i=1,\ add\ to\ right} [16, 2] \xrightarrow{i=2,\ add\ to\ left} [18]$. All elements of the array $[18]$ are the same.

In the second test case of the example, the answer can be constructed like this (just one way among other ways):

$[1, 2, 2, 1] \xrightarrow{i=1,\ add\ to\ right} [3, 2, 1] \xrightarrow{i=3,\ add\ to\ left} [3, 3]$. All elements of the array $[3, 3]$ are the same.

In the third test case of the example, Polycarp doesn't need to perform any operations since $[2, 2, 2]$ contains equal (same) elements only.

In the fourth test case of the example, the answer can be constructed like this (just one way among other ways):

$[6, 3, 2, 1] \xrightarrow{i=3,\ add\ to\ right} [6, 3, 3] \xrightarrow{i=3,\ add\ to\ left} [6, 6]$. All elements of the array $[6, 6]$ are the same.

E1. Close Tuples (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the easy version of this problem. The only difference between easy and hard versions is the constraints on k and m (in this version $k = 2$ and $m = 3$). Also, in this version of the problem, you DON'T NEED to output the answer by modulo.

You are given a sequence a of length n consisting of integers from 1 to n. **The sequence may contain duplicates (i.e. some elements can be equal)**.

Find the number of tuples of $m = 3$ elements such that the maximum number in the tuple differs from the minimum by no more than $k = 2$. Formally, you need to find the number of triples of indices $i < j < z$ such that

$$\max(a_i, a_j, a_z) - \min(a_i, a_j, a_z) \le 2.$$

For example, if $n = 4$ and $a = [1, 2, 4, 3]$, then there are two such triples ($i = 1, j = 2, z = 4$ and $i = 2, j = 3, z = 4$). If $n = 4$ and $a = [1, 1, 1, 1]$, then all four possible triples are suitable.

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases. Then t test cases follow.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the sequence a.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the sequence a.

It is guaranteed that the sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

Output t answers to the given test cases. Each answer is the required number of triples of elements, such that the maximum value in the triple differs from the minimum by no more than 2. Note that in difference to the hard version of the problem, you **don't need** to output the answer by modulo. You must output the exact value of the answer.

Example

input

```
4
4
1 2 4 3
4
1 1 1 1
1
1
10
5 6 1 3 2 9 8 1 2 4
```

output

```
2
4
0
15
```

E2. Close Tuples (hard version)

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of this problem. The only difference between the easy and hard versions is the constraints on

k **and** m**. In this version of the problem, you need to output the answer by modulo** $10^9 + 7$**.**

You are given a sequence a of length n consisting of integers from 1 to n. **The sequence may contain duplicates (i.e. some elements can be equal)**.

Find the number of tuples of m elements such that the maximum number in the tuple differs from the minimum by no more than k. Formally, you need to find the number of tuples of m indices $i_1 < i_2 < \ldots < i_m$, such that

$$\max(a_{i_1}, a_{i_2}, \ldots, a_{i_m}) - \min(a_{i_1}, a_{i_2}, \ldots, a_{i_m}) \le k.$$

For example, if $n = 4$, $m = 3$, $k = 2$, $a = [1, 2, 4, 3]$, then there are two such triples ($i = 1, j = 2, z = 4$ and $i = 2, j = 3, z = 4$). If $n = 4$, $m = 2$, $k = 1$, $a = [1, 1, 1, 1]$, then all six possible pairs are suitable.

As the result can be very large, you should print the value modulo $10^9 + 7$ **(the remainder when divided by** $10^9 + 7$**).**

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — the number of test cases. Then t test cases follow.

The first line of each test case contains three integers n, m, k ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 100$, $1 \le k \le n$) — the length of the sequence a, number of elements in the tuples and the maximum difference of elements in the tuple.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the sequence a.

It is guaranteed that the sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

Output t answers to the given test cases. Each answer is the required number of tuples of m elements modulo $10^9 + 7$, such that the maximum value in the tuple differs from the minimum by no more than k.

Example

input

```
4
4 3 2
1 2 4 3
4 2 1
1 1 1 1
1 1 1
1
10 4 3
5 6 1 3 2 9 8 1 2 4
```

output

```
2
6
1
20
```

F. The Treasure of The Segments

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp found n segments on the street. A segment with the index i is described by two integers l_i and r_i — coordinates of the beginning and end of the segment, respectively. Polycarp realized that he didn't need all the segments, so he wanted to delete some of them.

Polycarp believes that a set of k segments is good if there is a segment $[l_i, r_i]$ ($1 \le i \le k$) from the set, such that it intersects every segment from the set (the intersection must be a **point or segment**). For example, a set of 3 segments $[[1, 4], [2, 3], [3, 6]]$ is good, since the segment $[2, 3]$ intersects each segment from the set. Set of 4 segments $[[1, 2], [2, 3], [3, 5], [4, 5]]$ is not good.

Polycarp wonders, what is the minimum number of segments he has to delete so that the remaining segments form a good set?

Input

The first line contains a single integer t ($1 \le t \le 2 \cdot 10^5$) — number of test cases. Then t test cases follow.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of segments. This is followed by n lines describing the segments.

Each segment is described by two integers l and r ($1 \le l \le r \le 10^9$) — coordinates of the beginning and end of the segment, respectively.

It is guaranteed that the sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of segments that need to be deleted in order for the set of remaining segments to become good.

Example

input

```
4
3
1 4
2 3
3 6
4
1 2
2 3
3 5
4 5
5
1 2
3 8
4 5
6 7
9 10
5
1 5
2 4
3 5
3 8
4 8
```

output

```
0
1
2
0
```