Codeforces Round 961 (Div. 2)

A. Diagonals

1 second, 256 megabytes

Vitaly503 is given a checkered board with a side of n and k chips. He realized that all these k chips need to be placed on the cells of the board (no more than one chip can be placed on a single cell).

Let's denote the cell in the i-th row and j-th column as (i, j). A diagonal is the set of cells for which the value $i + j$ is the same. For example, cells $(3, 1)$, $(2, 2)$, and $(1, 3)$ lie on the same diagonal, but $(1, 2)$ and $(2, 3)$ do not. A diagonal is called occupied if it contains at least one chip.

Determine what is the minimum possible number of occupied diagonals among all placements of k chips.

Input

Each test consists of several sets of input data. The first line contains a single integer t $(1 \le t \le 500)$ — the number of sets of input data. Then follow the descriptions of the sets of input data.

The only line of each set of input data contains two integers n, k ($1 \le n \le 100, 0 \le k \le n^2$) — the side of the checkered board and the number of available chips, respectively.

Output

For each set of input data, output a single integer — the minimum number of occupied diagonals with at least one chip that he can get after placing all k chips.

input

```
7
1 0
2 2
2 3
2 4
10 50
100 239
3 9
```

output

```
0
1
2
3
6
3
5
```

In the first test case, there are no chips, so 0 diagonals will be occupied. In the second test case, both chips can be placed on diagonal $(2, 1), (1, 2)$, so the answer is 1. In the third test case, 3 chips can't be placed on one diagonal, but placing them on $(1, 2), (2, 1), (1, 1)$ makes 2 diagonals occupied. In the 7th test case, chips will occupy all 5 diagonals in any valid placing.

B1. Bouquet (Easy Version)

1.5 seconds, 512 megabytes

This is the easy version of the problem. The only difference is that in this version, the flowers are specified by enumeration.

A girl is preparing for her birthday and wants to buy the most beautiful bouquet. There are a total of n flowers in the store, each of which is characterized by the number of petals, and a flower with k petals costs k coins. The girl has decided that the difference in the number of petals between any two flowers she will use in her bouquet should not exceed one. At the same time, the girl wants to assemble a bouquet with the maximum possible number of petals. Unfortunately, she only has m coins, and she cannot spend more. What is the maximum total number of petals she can assemble in the bouquet?

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 10\,000)$ — the number of test cases. This is followed by descriptions of the test cases.

The first line of each test case contains two integers n, m ($1 \le n \le 2 \cdot 10^5, 1 \le m \le 10^{18}$) — the number of flowers in the store and the number of coins the girl possesses, respectively. The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$, where a_i is the number of petals of the i-th flower in the store.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the maximum possible number of petals in the bouquet that the girl can assemble while meeting all the conditions listed above.

input

```
5
5 10
1 1 2 2 3
8 20
4 2 7 5 6 1 1 1
8 100000
239 30 610 122 24 40 8 2
11 13
2 4 11 1 1 2 3 5 4 3 2
8 1033
206 206 206 207 207 207 207 1000
```

output

```
7
13
610
13
1033
```

In the first test case, you can assemble a bouquet with $(1, 1, 2, 2), (2, 2, 3), (1, 1), (2, 2)$. The maximum over all valid bouquets not greater than 10 is 7 for $(2, 2, 3)$. In the third test case, you can assemble a bouquet with only one flower of any type, so the answer is 610. In the fourth test case, you can assemble a bouquet with $(4, 4, 5)$, which gives you 13 petals, and it is the maximum amount of petals that the girl can buy.

B2. Bouquet (Hard Version)

1.5 seconds, 256 megabytes

This is the hard version of the problem. The only difference is that in this version, instead of listing the number of petals for each flower, the number of petals and the quantity of flowers in the store is set for all types of flowers.

A girl is preparing for her birthday and wants to buy the most beautiful bouquet. There are a total of n different types of flowers in the store, each of which is characterized by the number of petals and the quantity of this type of flower. A flower with k petals costs k coins. The girl has decided that the difference in the number of petals between any two flowers she will use to decorate her cake should not exceed one. At the same time, the girl wants to assemble a bouquet with the maximum possible number of petals. Unfortunately, she only has m coins, and she cannot spend more. What is the maximum total number of petals she can assemble in the bouquet?

Input

Each test consists of several test cases. The first line contains a single integer t $(1 \le t \le 10\,000)$ — the number of test cases. This is followed by descriptions of the test cases.

The first line of each test case contains two integers n, m ($1 \le n \le 2 \cdot 10^5, 1 \le m \le 10^{18}$) — the number of types of flowers in the store and the number of coins the girl possesses, respectively. The second line of each test case contains n **different** integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$, where a_i is the number of petals of the i-th flower type in the store (for different indexes $i \ne j$, it must be $a_i \ne a_j$). The third line of each test case contains n integers $c_1, c_2, \ldots, c_n$ $(1 \le c_i \le 10^9)$, where c_i is the quantity of the i-th flower type in the store.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print one integer — the maximum possible number of petals in a bouquet that a girl can collect, observing all the conditions listed above.

input

```
7
3 10
1 2 3
2 2 1
3 1033
206 207 1000
3 4 1
6 20
4 2 7 5 6 1
1 2 1 3 1 7
8 100000
239 30 610 122 24 40 8 2
12 13123 112 1456 124 100 123 10982
6 13
2 4 11 1 3 5
2 2 1 2 2 1
8 10330
206 210 200 201 198 199 222 1000
9 10 11 12 13 14 15 16
2 10000000000
11 12
87312315 753297050
```

output

```
7
1033
19
99990
13
10000
9999999999
```

In the first test case, some valid bouquets are $(1, 1, 2, 2), (2, 2, 3), (1, 1), (2, 2)$. The maximum over all valid bouquets not greater than 10 is 7 for $(2, 2, 3)$. In the second test case, you can assemble a valid bouquet with $(206, 206, 207, 207, 207)$ with a sum of 1033, which is the maximum number of petals the girl can buy. In the third test case, you can assemble a valid bouquet with $(5, 5, 5, 4)$ with a sum of 19. It can be seen that no valid bouquet can have 20 petals.

C. Squaring

2 seconds, 256 megabytes

`ikrpprpp` found an array a consisting of integers. He likes justice, so he wants to make a fair — that is, make it non-decreasing. To do that, he can perform an *act of justice* on an index $1 \le i \le n$ of the array, which will replace a_i with a_i^2 (the element at position i with its square). For example, if $a = [2, 4, 3, 3, 5, 3]$ and `ikrpprpp` chooses to perform an act of justice on $i = 4$, a becomes $[2, 4, 3, 9, 5, 3]$.

What is the minimum number of acts of justice needed to make the array non-decreasing?

Input

First line contains an integer t $(1 \le t \le 1000)$ — the number of test cases. It is followed by the description of test cases.

For each test case, the first line contains an integer n — size of the array a. The second line contains n $(1 \le n \le 2 \cdot 10^5)$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$).

Output

For each testcase, print an integer — minimum number of acts of justice required to make the array a non-decreasing. If it is impossible to do that, print -1.

input

```
7
3
1 2 3
2
3 2
3
3 1 5
4
1 1 2 3
3
4 3 2
9
16 2 4 2 256 2 4 2 8
11
10010 10009 10008 10007 10006 10005 10004 10003 10002 10001 10000
```

output

```
0
1
-1
0
3
15
55
```

In the first test case, there's no need to perform acts of justice. The array is fair on its own!

In the third test case, it can be proven that the array cannot become non-decreasing.

In the fifth test case, ikrpprppp can perform an act of justice on index 3, then an act of justice on index 2, and finally yet another act of justice on index 3. After that, a will become $[4, 9, 16]$.

D. Cases

2 seconds, 256 megabytes

You're a linguist studying a mysterious ancient language. You know that

1. Its words consist only of the first c letters of the Latin alphabet.
2. Each word has a *case* which can be unambiguously determined by its last letter (different letters correspond to different cases). For example, words "ABACABA" and "ABA" (if they exist) have the same case in this language because they both have the same ending 'A', whereas "ALICE" and "BOB" have different cases. If the language does not have a case corresponding to some letter, it means that the word cannot end with this letter.
3. The length of each word is k or less.

You have a single text written in this language. Unfortunately, as the language is really ancient, spaces between words are missing and all letters are uppercase. You wonder what is the minimum number of cases the language can have. Can you find this out?

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. It is followed by descriptions of the test cases.

The first line of each test case contains three integers n, c, k ($1 \le k \le n \le 2^{18}$, $1 \le c \le 18$) — the length of the text, the number of letters in the language, and the maximum length of the word.

The second line contains a string of n characters — the text itself. Each character is one of the first c uppercase letters of the Latin alphabet.

It is guaranteed that the sum of n over all test cases does not exceed 2^{18} and the sum of 2^c over all test cases does not exceed 2^{18}.

Output

For each test case, output a single line consisting of a single integer — the minimum number of cases in the language.

input

```
7
5 5 1
ABCDE
3 1 2
AAA
3 2 2
AAB
10 2 2
ABABABABAB
4 4 4
DCBA
1 17 1
Q
9 3 2
ABCABCABC
```

output

```
5
1
2
1
1
1
2
```

In the first test case, there must be five cases in the language (for each of the letters 'A', 'B', 'C', 'D', and 'E' there must be a case that has a corresponding ending).

In the third test case, one case with ending 'B' is sufficient.

E1. Let Me Teach You a Lesson (Easy Version)

2 seconds, 256 megabytes

This is the easy version of a problem. The only difference between an easy and a hard version is the constraints on t and n. You can make hacks only if both versions of the problem are solved.

Arthur is giving a lesson to his famous $2n$ knights. Like any other students, they're sitting at the desks in pairs, but out of habit in a circle. The knight $2i-1$ is sitting at the desk with the knight $2i$.

Each knight has *intelligence*, which can be measured by an integer. Let's denote the intelligence of the i-th knight as a_i. Arthur wants the maximal difference in total intelligence over all pairs of desks to be as small as possible. More formally, he wants to minimize $\max\limits_{1 \le i \le n}(a_{2i-1}+a_{2i}) - \min\limits_{1 \le i \le n}(a_{2i-1}+a_{2i})$.

However, the Code of Chivalry only allows swapping the opposite knights in the circle, i.e., Arthur can simultaneously perform $a_i := a_{i+n}$, $a_{i+n} := a_i$ for any $1 \le i \le n$. Arthur can make any number of such swaps. What is the best result he can achieve?

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. It is followed by descriptions of the test cases.

The first line of each test case contains a single integer n ($1 \le n \le 2000$) — the number of desks.

The second line consists of $2n$ integers $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le 10^9$) — the intelligence values of the knights.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output a single line containing one integer — the minimal difference Arthur can achieve.

input

```
5
2
6 6 4 4
1
10 17
3
1 10 1 10 1 10
3
3 3 4 5 5 4
5
1 2 3 4 5 6 7 8 9 10
```

output

```
0
0
0
2
4
```

In the first test case, Arthur can swap the second and the fourth knights. Then the total intelligence at both desks will be 10.

In the third test case, Arthur can make 0 operations, which will result in the total intelligence of 11 at each of the desks.

In the fourth test case, Arthur can swap knights with indices 2 and 5 and achieve the difference of 2. It can be proven that he cannot improve his result any further.

E2. Let Me Teach You a Lesson (Hard Version)

2 seconds, 256 megabytes

This is the hard version of a problem. The only difference between an easy and a hard version is the constraints on t and n. You can make hacks only if both versions of the problem are solved.

Arthur is giving a lesson to his famous $2n$ knights. Like any other students, they're sitting at the desks in pairs, but out of habit in a circle. The knight $2i-1$ is sitting at the desk with the knight $2i$.

Each knight has *intelligence*, which can be measured by an integer. Let's denote the intelligence of the i-th knight as a_i. Arthur wants the maximal difference in total intelligence over all pairs of desks to be as small as possible. More formally, he wants to minimize $\max\limits_{1 \le i \le n} (a_{2i-1} + a_{2i}) - \min\limits_{1 \le i \le n} (a_{2i-1} + a_{2i})$.

However, the Code of Chivalry only allows swapping the opposite knights in the circle, i.e., Arthur can simultaneously perform $a_i := a_{i+n}$, $a_{i+n} := a_i$ for any $1 \le i \le n$. Arthur can make any number of such swaps. What is the best result he can achieve?

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases. It is followed by descriptions of the test cases.

The first line of each test case contains a single integer n ($1 \le n \le 100\,000$) — the number of desks.

The second line consists of $2n$ integers $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le 10^9$) — the intelligence values of the knights.

It is guaranteed that the sum of n over all test cases does not exceed $100\,000$.

Output

For each test case, output a single line containing one integer — the minimal difference Arthur can achieve.

input

```
5
2
6 6 4 4
1
10 17
3
1 10 1 10 1 10
3
3 3 4 5 5 4
5
1 2 3 4 5 6 7 8 9 10
```

output

```
0
0
0
2
4
```

In the first test case, Arthur can swap the second and the fourth knights. Then the total intelligence at both desks will be 10.

In the third test case, Arthur can make 0 operations, which will result in the total intelligence of 11 at each of the desks.

In the fourth test case, Arthur can swap knights with indices 2 and 5 and achieve the difference of 2. It can be proven that he cannot improve his result any further.