Problem A. DFS Order

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

Prof. Pang has a rooted tree which is rooted at 1 with 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, what is the minimum and the maximum position it can appear in the **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, what are the minimum value and the maximum value of j such that v appears in the j-th position.

Following is a pseudo-code for the depth-first search on a rooted tree. After its execution, `dfs_order` is the depth-first search order.

```
let dfs_order be an empty list

def dfs(vertex x):
    append x to the end of dfs_order.
    for (each son y of x): // sons can be iterated in arbitrary order.
        dfs(y)

dfs(root)
```

Input

The first line contains a single integer T $(1 \le T \le 10^6)$ denoting the number of test cases.

For each test case, the first line contains an integer n $(1 \le n \le 10^5)$. Each of the next $n-1$ lines contains two integers x and y, indicating node x is node y's parent $(1 \le x, y \le n)$. These edges form a tree rooted at 1.

It is guaranteed that the sum of n over all test cases is no more than 10^6.

Output

For each test case, print n lines. The i-th line contains two integers denoting the minimum and the maximum position node i can appear in the depth-first search order.

Example

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

Problem B. Beautiful String

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

Prof. Pang recently got a dictionary of the elvish language, including many strings representing their words. He thinks a partition of string s is beautiful if both of the following conditions are satisfied:

- $s = s_1 + s_2 + s_3 + s_4 + s_5 + s_6$, where $s_i (1 \leq i \leq 6)$ are nonempty substrings. $a + b$ means the concatenation of string a and b here.

- $s_1 = s_2 = s_5, s_3 = s_6$.

For example, you can partition the string "114514" into 6 parts : "114514" = "1" + "1" + "4" + "5" + "1" + "4". The first, second, fifth parts are the same, and the third and sixth parts are the same. Thus, the partition of $s =$ "114514" into $s_1 =$ "1", $s_2 =$ "1", $s_3 =$ "4", $s_4 =$ "5", $s_5 =$ "1", and $s_6 =$ "4" is beautiful.

Accordingly, the beauty of a string s is defined as the number of beautiful partitions of s.

Given a string t, please help Prof. Pang to figure out the sum of beauties of all substrings of t.

Input

The first line contains a single integer T ($1 \leq T \leq 50$) indicating the number of test cases.

For each test case, there is one single line containing the string t, consisting of digits from '0' to '9'.

It is guaranteed that the length of each t in each test case will not exceed 5000 and the total length will not exceed 30000.

Output

For each test case, output a single line containing an integer, indicating the sum of beauties of all substrings of t.

Example

standard input	standard output
2	1
114514	3
0000000	

Problem C. String-dle Count

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

While most people love to play Wordle these days, Prof. Pang has become addicted to String-dle.

String-dle is an interesting string-guessing game where the player tries to guess a string of k capital letters through several rounds. In each round, the player submits a string of length k as his guess, and the system grades the guess through the following pseudo-code:

```
def grading(answer, guess):
  let count be a hash map
  for i = 1 to k:
    if answer[i] not in count:
      count[answer[i]] = 1
    else:
      count[answer[i]] = count[answer[i]] + 1
  let grade be an array of length k
  for i = 1 to k:
    if answer[i] == guess[i]:
      grade[i] = 'O'
      count[guess[i]] = count[guess[i]] - 1
  for i = 1 to k:
    if answer[i] != guess[i]:
      if count[guess[i]] > 0:
        grade[i] = '-'
        count[guess[i]] = count[guess[i]] - 1
      else:
        grade[i] = 'x'
  return grade
```

The grade consisting of O (capital letter O), - (dash), and x (small letter x) is then returned to the player, and the player may base his next guess on the previous grades. The following is an example of one game Prof. Pang played:

```
G: CRANE
A: xx--x
G: UTTER
A: xxOxx
G: NASAL
A: OOxOO
G: NATAL
A: OOOOO
```

Strings after G are Prof. Pang's guesses and strings after A are the grades of the guesses.

Prof. Pang really enjoys the game. He believes that he has developed a perfect strategy for it. However, today he goes mad because he thinks the grading system is bugged! He wants to have someone write an analysis program to figure out the number of possible strings that can be the answer to the puzzle based on his list of guesses and grades.

Since the grading system might be bugged, it might not conform to the pseudo-code given above. So specifically, the task is to find how many strings that are consistent with the input. A string s is consistent with the input if for any guess g in the input and its corresponding grade d, grading(s, g)=d.

And of course, you will be doing the programming.

Input

The first line consists of two integers n and k ($1 \le n \le 10^4, 1 \le k \le 19$), which are the number of guesses and the length of the string, respectively.

The following lines consist of the guesses and the grades, one per line, correspondingly.

Output

An integer, denoting how many possible answers there are, modulo $10^9 + 7$.

Examples

standard input	standard output
2 5 CRANE xx--x NASAL 00x00	21
1 5 BBBAA xxxx-	0
2 5 ABCDE -xxxx ABCDE xxxxx	0
1 3 ABC ---	2
1 15 AAAAAAAAAAAAAAB -xxxxxxxxxxxxxx	918547951
1 15 AAAAAAAAAAAAAAA -xxxxxxxxxxxxxx	0
1 1 K x	25

Note

For the second example:

If the answer is ACDEF, the guess BBBAA will produce a grade of xxxx-.

Problem D. Two Walls

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	1024 megabytes

Prof. Pang has bought a humanoid cleaning robot to clean his yard. The robot is not very sophisticated. It can either move forward or change its direction at a time, all controlled by Prof. Pang's controller.

Prof. Pang's yard is a 2D plane. The robot needs to move from its current location A to the destination B to fulfill some "cleaning" needs of Prof. Pang. However, there are two straight walls CD and EF in Prof. Pang's yard. Since the robot is clumsy, it will fall over if it touches any of the walls (even at endpoints).

Now that Prof. Pang is lazy, he wants to minimize the number of times the robot changes its direction. Can you help him?

Input

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

For each test case, the first line contains two integers x_A, y_A, the coordinates of point A. The second line contains two integers x_B, y_B, the coordinates of point B. The third line contains four integers x_C, y_C, x_D, y_D, the coordinates of point C and D which are the endpoints of the first wall. The fourth line contains four integers x_E, y_E, x_F, y_F, the coordinates of point E and F which are the endpoints of the second wall.

It is guaranteed that neither the current location A nor the destination B of the robot are on any of the walls. A wall may degenerate to a point. It can be proved that the robot can always move from A to B without touching any of the walls. All values lie within $[-10^9, 10^9]$.

Output

For each test case, print one number d in one line, denoting the minimum number of turns.

Example

standard input	standard output
3	0
0 0	0
1 1	1
2 2 3 3	
4 4 5 5	
0 0	
1 1	
2 2 3 3	
2 2 3 3	
0 0	
10 10	
10 0 0 10	
1 1 2 2	

Note

The following are illustrations for the first sample and the third sample.
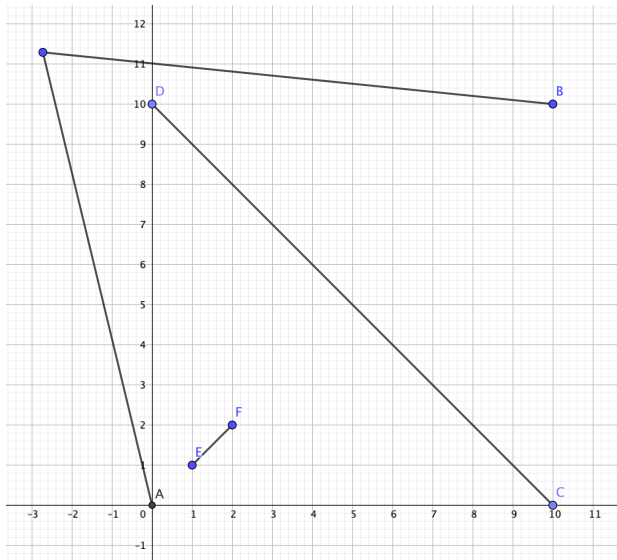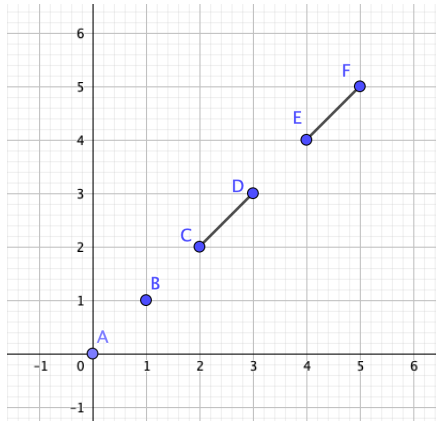
Problem E. Prof. Pang and Poker

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

Prof. Pang is playing a card game with his two friends Alice and Bob. All cards are drawn from a standard 52-card deck. A standard 52-card deck comprises 13 ranks in each of the four French suits: clubs (♣), diamonds (♢), hearts (♡) and spades (♠). Each suit includes an Ace (A), a King (K), a Queen (Q), and a Jack (J), each depicted alongside a symbol of its suit; and numerals or pip cards from the Deuce (Two) to the Ten, with each card depicting that many symbols (pips) of its suit. **No card can be drawn more than once.**

Example set of 52 playing cards; 13 of each suit: clubs, diamonds, hearts, and spades

Individual cards are ranked as follows (high-to-low): A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3, 2. **Suits do not affect ranks of the cards.** For example, Ace of diamonds and Ace of clubs have the same rank. No one of them is ranked strictly higher than the other.

Initially, Alice and Bob will hold one or more cards while Prof. Pang will hold exactly one card. **Each player can see cards held by himself/herself and cards held by other players.** They will play the game in the following multi-round rule:

- The initiative player chooses one card and put it out to start one round.

- The next player can pass or put out a new card, then the player after the next can also pass or put out a new card, and so on. The only constraint is that the rank of the newly put card should be strictly higher than all previous cards in this round.

- The round ends when two players choose to pass consecutively. The one who put out the last card becomes the initiative player in the next round.

- If someone put out all the cards in his/her hand, the game ends immediately.

In this game, Alice is the initiative player in the first round. Bob, Prof. Pang, and Alice are the next players of Alice, Bob, and Prof. Pang respectively. Prof. Pang will be happy if and only if he is the one that first put out all the cards. (Prof. Pang wants to be happy, of course.) Alice wants to drink milk tea so she decides to make Prof. Pang happy and then asks him to buy milk tea for her. However, Bob doesn't want it to happen, so he decides to avoid Prof. Pang from being happy. If they play the game optimally for themselves, will Prof. Pang be happy in the end?

Input

The first line contains a single integer T ($1 \le T \le 10^4$) denoting the number of test cases. For each test case:

The first line contains two integers n, m ($1 \leq n, m \leq 50$) denoting the number of cards in Alice's hand and Bob's hand initially.

The second line contains n strings a_i ($1 \leq i \leq n$) denoting the cards in Alice's hand.

The third line contains m strings b_i ($1 \leq i \leq m$) denoting the cards in Bod's hand.

The fourth line contains one string p denoting the card in Prof. Pang's hand.

For each card, the first character of its corresponding string denotes its rank. (Possible ranks are '2' ,'3','4','5','6','7','8','9','T','J','Q','K','A'. 'T' denotes 10.) The second character denotes its suit. 'C' denotes clubs. 'D' denotes diamonds. 'H' denotes hearts. 'S' denotes spades.

It is guaranteed that each card appears at most once in one test case.

Output

For each test case, print one line. Print "Pang" if Prof. Pang will be happy. Otherwise, print "Shou".

Example

standard input	standard output
2	Pang
2 2	Shou
2H 2D	
3H 3D	
4S	
2 2	
2H 2D	
3H 4D	
4S	

Note

- For the first case, Prof. Pang can always put out his only card "4S".

- For the second case, Bob can put out "4D" and become the initiative player in the second round regardless of the card Alice put out in the first round, then Bob put out "3H" and the game ends.

Problem F. Vacation

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	1024 megabytes

Prof. Pang has an annual leave of c days and he wants to go on vacation.

Now there are n days in a year. Prof. Pang can gain a_i happiness if he rests on the i-th day. The values of happiness, a_i, may be negative.

Prof. Pang wants you to do m operations:

- 1 x y, change the happiness of the x-th day to y.

- 2 l r, Prof. Pang wants to find a period of vacation in $[l, r]$. He wants to rest for several (possibly 0) days in a row and gain as much happiness as possible. However, he only has c days off, thus he can rest for no more than c consecutive days in $[l, r]$.

That means he wants to find
$$\max\left(\max_{\substack{l \le l' \le r' \le r \\ r'-l'+1 \le c}} \left(\sum_{i=l'}^{r'} a_i\right), 0\right).$$

Input

The first line contains three integers $n, m, c (1 \le n \le 2 \times 10^5, 1 \le m \le 5 \times 10^5, 1 \le c \le n)$ indicating the number of days in a year, the number of operations, and Prof. Pang's annual leave days.

The next line contains n integers $a_1, a_2, \ldots, a_n (-10^9 \le a_i \le 10^9)$ indicating the values of happiness of every day.

The next m lines are the m operations in the format described above.

It is guaranteed that $1 \le x \le n, -10^9 \le y \le 10^9, 1 \le l \le r \le n$.

Output

For each operation of the second type, print the answer.

Example

standard input	standard output
5 6 3	8
0 -5 -3 8 -3	10
2 3 5	0
1 2 5	5
2 1 5	
1 4 -3	
2 3 5	
2 1 5	

Problem G. Check Pattern is Bad

Input file:	standard input
Output file:	standard output
Time limit:	10 seconds
Memory limit:	1024 megabytes

Prof. Pang is given an $n \times m$ board. Some cells are colored black, some cells are colored white, and others are uncolored.

Prof. Pang doesn't like **check patterns**, so he wants to color all uncolored cells such that there is no check pattern on the board.

4 cells forming a 2×2 square are said to have the check pattern if they are colored in one of the following ways: $\begin{smallmatrix} \text{BW} \\ \text{WB} \end{smallmatrix}$ or $\begin{smallmatrix} \text{WB} \\ \text{BW} \end{smallmatrix}$. Here 'W' ("wakuda" in Chewa language) means the cell is colored black and 'B' ("biancu" in Corsican language) means the cell is colored white.

Input

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

The first line of each test case contains two integers n and m ($1 \le n, m \le 100$) denoting the dimensions of the board.

Each of the next n lines contains m characters. The j-th character of the i-th line represents the status of the cell on the i-th row and j-th column of the board. The character is 'W' if the cell is colored black, 'B' if the cell is colored white, and '?' if the cell is uncolored.

It is guaranteed that the sum of nm over all test cases is no more than 10^6.

Output

For each test case, output a line containing "NO" if you cannot color all the uncolored cells such that there is no check pattern on the board.

Otherwise, output a line containing "YES". In the next n lines, output the colored board in the same format as the input. The output board should satisfy the following conditions.

- It does not have any check pattern.

- It consists of only 'B' and 'W'.

- If a cell is already colored in the input, its color cannot be changed in the output.

If there are multiple solutions, output any of them.

Example

standard input	standard output
3	YES
2 2	BW
??	WW
??	NO
3 3	YES
BW?	BWB
W?B	WWW
?BW	BWB
3 3	
BW?	
W?W	
?W?	

Problem H. Check Pattern is Good

Input file:	standard input
Output file:	standard output
Time limit:	10 seconds
Memory limit:	1024 megabytes

Prof. Shou is given an $n \times m$ board. Some cells are colored black, some cells are colored white, and others are uncolored.

Prof. Shou likes **check patterns**, so he wants to color all uncolored cells and maximizes the number of check patterns on the board.

4 cells forming a 2×2 square are said to have the check pattern if they are colored in one of the following ways: $\begin{smallmatrix} \texttt{BW} \\ \texttt{WB} \end{smallmatrix}$ or $\begin{smallmatrix} \texttt{WB} \\ \texttt{BW} \end{smallmatrix}$. Here 'W' ("wakuda" in Chewa language) means the cell is colored black and 'B' ("biancu" in Corsican language) means the cell is colored white.

Input

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

The first line of each test case contains two integers n and m ($1 \le n, m \le 100$) denoting the dimensions of the board.

Each of the next n lines contains m characters. The j-th character of the i-th line represents the status of the cell on the i-th row and j-th column of the board. The character is 'W' if the cell is colored black, 'B' if the cell is colored white, and '?' if the cell is uncolored.

It is guaranteed that the sum of nm over all test cases is no more than 10^6.

Output

For each test case, output a line containing the maximum number of check patterns on the board.

In the next n lines, output the colored board in the same format as the input. The output board should satisfy the following conditions.

- It consists of only 'B' and 'W'.

- If a cell is already colored in the input, its color cannot be changed in the output.

- The number of check patterns equals the answer you print.

If there are multiple solutions, output any of them.

Example

standard input	standard output
3	1
2 2	WB
??	BW
??	1
3 3	BWB
BW?	WWB
W?B	BBW
?BW	4
3 3	BWB
BW?	WBW
W?W	BWB
?W?	

Problem I. Future Coder

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Prof. Pang builds his famous coding team recently. To pursue a gold medal in ICPC, hundreds of pupils join his team. Unfortunately, one of Prof. Pang's students believes that for any integers a and b, $a \times b \geq a + b$. To disprove this proposition, Prof. Pang writes n numbers $a_1, a_2, \ldots, a_n$ on a paper and wants you to count how many pairs of numbers (a_i, a_j) $(1 \leq i < j \leq n)$ satisfies $a_i \times a_j < a_i + a_j$.

Input

The first line contains a single integer T $(1 \leq T \leq 10^6)$ denoting the number of test cases.

For each test case, the first line contains a single integer n $(1 \leq n \leq 10^6)$. The second line contains n integers $a_1, a_2, \ldots, a_n$ $(-10^9 \leq a_i \leq 10^9)$.

It is guaranteed that the sum of n over all test cases will not exceed 10^6.

Output

For each test case, print one line containing the answer.

Example

standard input	standard output
2	19
8	0
3 -1 4 1 -5 9 2 -6	
1	
0	

Problem J. Elden Ring

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	1024 megabytes

Prof. Pang is getting addicted to the game called Elden Ring, in which the world is a connected graph including n vertices indexed from 1 to n and m undirected edges. Players start at vertex 1 and travel across the world to slay the god on vertex n.

However, it's not that easy. For any vertex i except vertex 1, there is exactly one boss whose level is l_i, and the player starts the game with level l_1. For each day, the player can travel to any vertex i from vertex 1 and challenge the boss there. If the current level of the player is greater than the boss, the boss will be eliminated from the world (inactivated) and the level of the player will be increased by A. Notice that traveling through a vertex that has an active boss is forbidden. (In other words, Prof. Pang can travel from vertex 1 to vertex i if there is a path in the graph from vertex 1 to vertex i such that each vertex on this path, except for vertex i, has no active boss.) Meanwhile, at the beginning of each day, all the remaining bosses in the world will also be promoted by B levels.

To finish a playthrough of the game, you need to slay the boss on vertex n (Elden Beast). Given the information of the world, Prof. Pang is wondering how many days he needs at least to do so.

The Player can only challenge one boss each day.

Input

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

For each test case, the first line includes four integers n, m, A, B ($2 \le n \le 2 \times 10^5, 1 \le m, A, B \le 2 \times 10^5$). In next m lines, each line contains two integers a_i, b_i ($1 \le a_i, b_i \le n$), denoting the endpoints of the i-th undirected edge. The last line contains n integers l_i ($1 \le l_i \le 2 \times 10^5$), representing the initial levels of the player and bosses mentioned above.

It is guaranteed that the sum of n over all test cases will not exceed 10^6 and the sum of m over all test cases will not exceed 10^6.

Output

For each test case, output a single line containing an integer, indicating the minimum number of days Prof. Pang needs to finish the game. If it is impossible to do so, please output -1.

Example

standard input	standard output
2	2
5 4 5 8	4
1 2	
1 3	
1 4	
4 5	
15 1 1 1 1	
5 4 10 5	
1 2	
1 3	
1 4	
4 5	
10 4 4 4 19	

Problem K. Vision Test

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Prof. Pang has an extraordinary vision. He can see the pixels on a 4K monitor. To test Prof. Pang's vision, Prof. Shou will show Prof. Pang several pixels and let Prof. Pang guess a straight line that contains these pixels. Given k pixels with coordinates (i, y_i) $(0 \le i < k)$, Prof. Pang must find nonnegative integers a, b and c (which represent the line $y = \frac{ax+b}{c}$) such that $y_i = \lfloor \frac{ai+b}{c} \rfloor$ for all $0 \le i < k$.

Prof. Shou will ask Prof. Pang multiple questions. They are given as follows: Prof. Shou has a fixed array $x_1, \ldots, x_n$. For each question, Prof. Shou chooses a range in the array, $x_l, \ldots, x_r$. Then he defines $y_i = x_{l+i}$ for $0 \le i \le r - l$ and asks Prof. Pang to answer the question for the $r - l + 1$ pixels $(0, y_0), \ldots, (r - l, y_{r-l})$.

Please help Prof. Pang answer all the questions. For each question, output the answer with the **minimum** (c, a, b) **in lexical order**.

It is guaranteed that the answer exists when Prof. Pang chooses the whole array $x_1, x_2, \ldots, x_n$. So the answer always exists when Prof. Pang chooses an interval of this array.

Input

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

For each test case, the first line contains an integer n $(1 \le n \le 10^5)$. The second line contains n numbers $x_1, \ldots, x_n$ $(0 \le x_i \le 10^9)$.

The next line contains an integer q $(1 \le q \le 10^5)$ denoting the number of questions.

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

It is guaranteed that the sum of n over all test cases will not exceed 10^5 and that the sum of q over all test cases will not exceed 10^5.

Output

In the order of input, output one line with three integers a, b, c denoting the answer for each question.

Example

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

Problem L. Fenwick Tree

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

Prof. Pang is giving a lecture on the Fenwick tree (also called binary indexed tree).

In a Fenwick tree, we have an array $c[1 \ldots n]$ of length n which is initially all-zero ($c[i] = 0$ for any $1 \le i \le n$). Each time, Prof. Pang can call the following procedure for some position pos ($1 \le pos \le n$) and value val:

```
def update(pos, val):
    while (pos <= n):
        c[pos] += val
        pos += pos & (-pos)
```

Note that `pos & (-pos)` equals to the maximum power of 2 that divides `pos` for any positive integer `pos`.

In the procedure, val can be **any real** number. After calling it some (zero or more) times, Prof. Pang forgets the exact values in the array c. He only remembers whether $c[i]$ is zero or not for each i from 1 to n. Prof. Pang wants to know what is the minimum possible number of times he called the procedure assuming his memory is accurate.

Input

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

For each test case, the first line contains an integer n ($1 \le n \le 10^5$). The next line contains a string of length n. The i-th character of the string is 1 if $c[i]$ is nonzero and 0 otherwise.

It is guaranteed that the sum of n over all test cases is no more than 10^6.

Output

For each test case, output the minimum possible number of times Prof. Pang called the procedure. It can be proven that the answer always exists.

Example

standard input	standard output
3	3
5	0
10110	3
5	
00000	
5	
11111	

Note

For the first example, Prof. Pang can call `update(1,1)`, `update(2,-1)`, `update(3,1)` in order.

For the third example, Prof. Pang can call `update(1,1)`, `update(3,1)`, `update(5,1)` in order.

Problem M. Prof. Pang and Ants

Input file:	standard input
Output file:	standard output
Time limit:	8 seconds
Memory limit:	1024 megabytes

Near Prof. Pang's big house, there is an ant group which has m ants and lives underground in a cave with n holes. They get out of the holes for food. So where is the food? It should be in Prof. Pang's big refrigerator. The ants want to steal the food from the big refrigerator.

Specifically, for an ant, it needs 1 second to leave from the cave through any hole and 1 second to enter the cave through any hole. Different holes have different locations. The distance between the i-th hole and the refrigerator is a_i. Thus, after leaving the cave through the i-th hole, an ant needs a_i seconds to go to the refrigerator. After stealing some food from the refrigerator, an ant needs a_i seconds to go back to the i-th hole. Stealing food costs no time for the ants since they are skillful enough.

Each ant will leave the cave to steal something from the refrigerator and return to (enter) the cave after stealing. Each ant must leave the cave exactly once and then return to the cave. One ant can arbitrarily choose one hole to leave and also arbitrarily choose one hole to enter, where the two holes need not be the same. For any hole, during any second, there can be at most one ant leaving or entering the cave through it. There cannot be one ant leaving the cave and another ant entering the cave using the same hole during the same second. Because of this capacity constraint, some ants may need to wait before they leave the hole and/or before they return to the hole after stealing.

So you, Prof. Pang's good friend, need to calculate the minimum time cost for ants to achieve the goal and play a trick on Prof. Pang so that the ants can steal the food without being caught by Prof. Pang. The time cost is defined as the total length of time during which at least one ant is not in the cave. When an ant is entering or leaving the cave through some hole, it is not in the cave.

Input

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

For each test case, the first line contains two integers n, m ($1 \le n \le 10^5, 1 \le m \le 10^{14}$) denoting the number of the holes and the number of the ants respectively. The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) denoting the time needed to get to the refrigerator from the i-th hole.

It is guaranteed that the sum of n over all test cases will not exceed 5×10^5.

Output

For each test case, print one line containing one integer denoting the minimum time cost in seconds.

Example

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

Note

In the third test case, it takes the ant 2 seconds to leave and enter the cave through the first hole. And it takes the ant 2 seconds to move to the refrigerator and back to the hole.