A. Make it Increasing

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

You are given an array a consisting of n positive integers, and an array b, with length n. Initially $b_i = 0$ for each $1 \leq i \leq n$.

In one move you can choose an integer i ($1 \leq i \leq n$), and add a_i to b_i or subtract a_i from b_i. What is the minimum number of moves needed to make b increasing (that is, every element is strictly greater than every element before it)?

Input

The first line contains a single integer n ($2 \leq n \leq 5000$).

The second line contains n integers, a_1, a_2, ..., a_n ($1 \leq a_i \leq 10^9$) — the elements of the array a.

Output

Print a single integer, the minimum number of moves to make b increasing.

Examples

input

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

output

```
4
```

input

```
7
1 2 1 2 1 2 1
```

output

```
10
```

input

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

output

```
16
```

Note

Example 1: you can subtract a_1 from b_1, and add a_3, a_4, and a_5 to b_3, b_4, and b_5 respectively. The final array will be [-1, 0, 3, 4, 5] after 4 moves.

Example 2: you can reach [-3, -2, -1, 0, 1, 2, 3] in 10 moves.

B. Optimal Partition

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

You are given an array a consisting of n integers. You should divide a into continuous non-empty subarrays (there are 2^{n-1} ways to do that).

Let $s = a_l + a_{l+1} + \ldots + a_r$. The value of a subarray $a_l, a_{l+1}, \ldots, a_r$ is:

- $(r - l + 1)$ if $s > 0$,
- 0 if $s = 0$,
- $-(r - l + 1)$ if $s < 0$.

What is the maximum sum of values you can get with a partition?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 5 \cdot 10^5$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 5 \cdot 10^5$).

The second line of each test case contains n integers a_1, a_2, ..., a_n ($-10^9 \le a_i \le 10^9$).

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

Output

For each test case print a single integer — the maximum sum of values you can get with an optimal parition.

Example

input

```
5
3
1 2 -3
4
0 -2 3 -4
5
-1 -2 3 -1 -1
6
-1 2 -3 4 -5 6
7
1 -1 -1 1 -1 -1 1
```

output

```
1
2
1
6
-1
```

Note

Test case 1: one optimal partition is $[1, 2]$, $[-3]$. $1+2>0$ so the value of $[1, 2]$ is 2. $-3<0$, so the value of $[-3]$ is -1. $2+(-1)=1$.

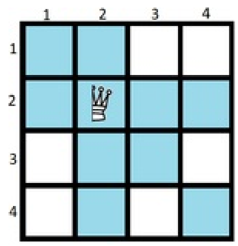
Test case 2: the optimal partition is $[0, -2, 3]$, $[-4]$, and the sum of values is $3+(-1)=2$.

C. Half Queen Cover

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a board with n rows and n columns, numbered from 1 to n. The intersection of the a-th row and b-th column is denoted by (a, b).

A half-queen attacks cells in the same row, same column, and on one diagonal. More formally, a half-queen on (a, b) attacks the cell (c, d) if $a=c$ or $b=d$ or $a-b=c-d$.



The blue cells are under attack.

What is the minimum number of half-queens that can be placed on that board so as to ensure that each square is attacked by at least one half-queen?
Construct an optimal solution.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the size of the board.

Output

In the first line print a single integer k — the minimum number of half-queens.

In each of the next k lines print two integers a_i, b_i ($1 \le a_i, b_i \le n$) — the position of the i-th half-queen.

If there are multiple solutions, print any.

Examples

input

```
1
```

output

```
1
1 1
```

input

```
2
```

output

```
1
1 1
```

input

```
3
```

output

```
2
1 1
1 2
```

Note

Example 1: one half-queen is enough. Note: a half-queen on $(1, 1)$ attacks $(1, 1)$.

Example 2: one half-queen is enough too. $(1, 2)$ or $(2, 1)$ would be wrong solutions, because a half-queen on $(1, 2)$ does not attack the cell $(2, 1)$ and vice versa. $(2, 2)$ is also a valid solution.

Example 3: it is impossible to cover the board with one half queen. There are multiple solutions for 2 half-queens; you can print any of them.

D. Edge Elimination

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

You are given a tree (connected, undirected, acyclic graph) with n vertices. Two edges are adjacent if they share exactly one endpoint. In one move you can remove an arbitrary edge, if that edge is adjacent to an even number of remaining edges.

Remove all of the edges, or determine that it is impossible. If there are multiple solutions, print any.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of vertices in the tree.

Then $n - 1$ lines follow. The i-th of them contains two integers u_i, v_i ($1 \le u_i, v_i \le n$) the endpoints of the i-th edge. It is guaranteed that the given graph is a tree.

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

Output

For each test case print "NO" if it is impossible to remove all the edges.

Otherwise print "YES", and in the next $n - 1$ lines print a possible order of the removed edges. For each edge, print its endpoints in any order.

Example

input

```
5
2
1 2
3
1 2
2 3
4
1 2
2 3
3 4
5
1 2
2 3
3 4
3 5
7
1 2
1 3
2 4
2 5
```

```
3 6
3 7
```

output

```
YES
2 1
NO
YES
2 3
3 4
2 1
YES
3 5
2 3
2 1
4 3
NO
```

Note

Test case 1: it is possible to remove the edge, because it is not adjacent to any other edge.

Test case 2: both edges are adjacent to exactly one edge, so it is impossible to remove any of them. So the answer is "NO".

Test case 3: the edge $2-3$ is adjacent to two other edges. So it is possible to remove it. After that removal it is possible to remove the remaining edges too.

E. Centroid Probabilities

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

Consider every tree (connected undirected acyclic graph) with n vertices (n **is odd**, vertices numbered from 1 to n), and for each $2 \le i \le n$ the i-th vertex is adjacent to exactly one vertex with a smaller index.

For each i ($1 \le i \le n$) calculate the number of trees for which the i-th vertex will be the centroid. The answer can be huge, output it modulo $998\,244\,353$.

A vertex is called a centroid if its removal splits the tree into subtrees with at most $(n-1)/2$ vertices each.

Input

The first line contains an odd integer n ($3 \le n < 2 \cdot 10^5$, n is odd) — the number of the vertices in the tree.

Output

Print n integers in a single line, the i-th integer is the answer for the i-th vertex (modulo $998\,244\,353$).

Examples

input

```
3
```

output

```
1 1 0
```

input

```
5
```

output

```
10 10 4 0 0
```

input

```
7
```

output

```
276 276 132 36 0 0 0
```

Note

Example 1: there are two possible trees: with edges $(1-2)$, and $(1-3)$ — here the centroid is 1; and with edges $(1-2)$, and $(2-3)$ — here the centroid is 2. So the answer is $1, 1, 0$.

Example 2: there are 24 possible trees, for example with edges $(1-2)$, $(2-3)$, $(3-4)$, and $(4-5)$. Here the centroid is 3.

F. Yin Yang

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

output: standard output

You are given a rectangular grid with n rows and m columns. n and m are divisible by 4. Some of the cells are already colored black or white. It is guaranteed that no two colored cells share a corner or an edge.

Color the remaining cells in a way that both the black and the white cells becomes orthogonally connected or determine that it is impossible.

Consider a graph, where the black cells are the nodes. Two nodes are adjacent if the corresponding cells share an edge. If the described graph is connected, the black cells are orthogonally connected. Same for white cells.

Input

The input consists of multiple test cases. The first line of the input contains a single integer t ($1 \le t \le 4000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n, m ($8 \le n, m \le 500$, n and m are divisible by 4) — the number of rows and columns.

Each of the next n lines contains m characters. Each character is either 'B', 'W' or '.', representing black, white or empty cell respectively. Two colored (black or white) cell does not share a corner or an edge.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $250\,000$.

Output

For each testcase print "NO" if there is no solution, otherwise print "YES" and a grid with the same format. If there are multiple solutions, you can print any.

Example

input

```
4
8 8
.W.W....
.....B.W
.W.W....
.....W.W
B.B.....
....B.B.
B.W.....
....B.B.
8 8
B.W..B.W
........
W.B..W.B
........
........
B.W..B.W
........
W.B..W.B
8 12
W.B.........
....B...B.W.
B.B.........
....B...B.B.
.B..........
........B...
.W..B.B...W.
............
16 16
.W............W.
...W..W..W.W....
.B...........B.W
....W....W......
W......B....W.W.
..W.......B.....
....W...W....B.W
.W....W....W....
...B...........W
W.....W...W..B..
..W.W...W......B
............W...
.W.B...B.B....B.
.....W.....W....
..W......W...W..
W...W..W...W...W
```

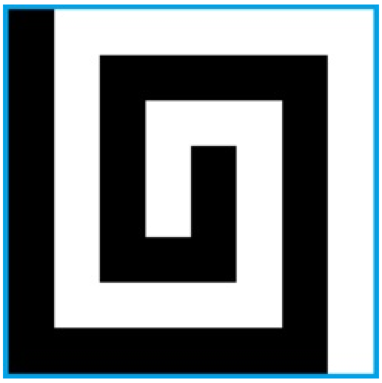
output

```
YES
BWWWWWWW
BWBBBBBW
BWBWWWBW
BWBWBWBW
BWBWBWBW
BWBBBWBW
BWWWWWBW
BBBBBBBW
NO
YES
WWBBBBBBBBBB
```

```
BWWWBBBBBBWB
BBBWBBBWWWWB
BBBWBBBWBBBB
BBBWBBBWBBBB
BBBWWWWWBBBB
BWWWBBBWWWWB
BBBBBBBBBBBB
YES
WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWW
WBBBBBBBBBBBBBWW
WBBBWBWWWWBWWBWW
WBBWWBBBWWBWWBWW
WBWWWBWWWWBWWBWW
WBBWWBBBWWBWWBWW
WWBWWWWWWWWWWWWW
WWBBBBBBBBBBBBWW
WBBBWWWBWWWBWBWW
WWWBWBBBWBBBWBBB
WWWBWBWWWWWBWWBW
WWWBWBBBWBBBWWBW
WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWW
```

Note

Solution for test case 1:



Test case 2: one can see that the black and the white part can't be connected in the same time. So the answer is "NO".