8470 The Number Triangle

As the baby step for learning dynamic programming, the problem about number triangle is very symbolic. Here we will revisit this classic problem.

By starting at any bottom of the triangle below and moving to adjacent numbers on the row above, the maximum total from bottom to top is 23.

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

However, as a lonely outcome of the whole problem, the maximum total is not enough. Similarly, we are concerned with the trail. Consider the new triangle below and a corresponding letter triangle. The maximum total from bottom to top is 6 passing through two '2'(s).

```
   2
  1 1
 2 1 1
1 1 2 1

   a
  a b
 b a a
b a b a
```

A best trail can be written by connecting all corresponding letters in the trail consecutively as a string.

Then in the above triangle, the lexicographically smallest best trail starting from the first position of the last row is "`bbaa`".

The lexicographically smallest best trail starting from the second position of the last row is "`abaa`".

The lexicographically smallest best trail starting from the third position of the last row is "`baaa`".

No best trail starts from the fourth position of the last row.

Now, a program is required to determine all positions of the last row leading at least one best trail (a best trail starting from this position). Sort these possible positions by the lexicographical order of the lexicographically smallest best trails which they lead respectively.

Input

The first line of input contains an integer T ($1 \leq T \leq 36$), indicating that there are T test cases.

For each test case, the first line contains an integer N ($N \leq 1500$). The i-th line of the following N lines describes the i-th row of the triangle with n pairs of an integer and a lowercase letter. All integers in input are positive and smaller than 10.

Output

For each query, output a list of numbers in a line indicating the sorted list of possible positions. If two positions own the same lexicographically smallest best trail, output the one with a smaller index in priority order.

Sample Input

```
3
4
3 e
7 v 4 a
2 a 4 o 6 t
8 w 5 f 9 l 3 p
4
2 a
1 a 1 b
2 b 1 a 1 a
1 b 1 a 2 b 2 a
4
2 a
1 a 1 b
2 b 1 a 1 a
1 b 1 a 2 b 1 a
```

Sample Output

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