Problem E: Approximate Sorting

Source: `sorting.{c,cpp,java}`

In many programming languages, library functions are provided for sorting elements in an array. In order for these sorting functions to work for different types of elements, the user supplies a comparison function **`less(x,y)`** which returns true if and only if **`x`** comes before **`y`** in the sorted order. Of course, the comparison function has to 'make sense'. For example, for any two different elements **`x`** and **`y`**, exactly one of **`less(x,y)`** and **`less(y,x)`** should be true. For the purpose of this problem, an array is sorted when there are no inversions with respect to the comparison function. An inversion with respect to **`less(x,y)`** in an array **`A`** of size **`n`** (indexed from 0) is a pair of integers **`0 ≤ i < j < n`** such that **`less(A[j], A[i]) = true`** (note that this may not be equivalent to **`less(A[i], A[j]) = false`**).

Unfortunately, some programmers are not very careful in defining the comparison function. In such cases, there may be no way to sort the elements in an array to satisfy the comparison function. The best we can do is to produce a permutation minimizing the number of inversions with respect to the given comparison function.

Input

For each case, an integer **`n`**, $1 \le n \le 18$, indicating the size of the array is given on one line. The elements in the array are labelled **`0`**, **`1`**, **`2`**, ..., **`n-1`**. The next **`n`** lines each contains a binary string of length **`n`**, with the **`j`**th character of the **`i`**th line indicating the result of the comparison function **`less(i, j)`** (0 means false and 1 means true). The end of input is indicated by a case in which **`n = 0`**. The last case should not be processed.

Output

For each case, output the permutation that has the smallest number of inversions with respect to the given comparison function. This is followed by a line containing a single integer indicating the number of inversions in the permutation. If there are multiple permutations with the same number of inversions, output the one that is lexicographically smallest.

Sample Input

```
4
0111
0000
0100
0110
3
011
011
011
6
101010
011010
110110
000000
111010
001010
0
```

Sample Output

```
0 3 2 1
0
0 1 2
1
0 1 5 2 3 4
5
```