IOI Training Camp 2010 – Final 3, 27 June, 2010

Problem 3 Tree Width[1]

A very useful measure of the complexity of a graph is its *treewidth*. Given an (undirected) graph $G = (V, E)$, *a tree decomposition* of G is an (undirected) tree $T = (S, F)$ along with a labelling f that labels each tree node $s \in S$ with a nonempty subset of V satisfying the following two properties:

1. If (u, v) is an edge in G, then there is a vertex $s \in S$ such that $f(s)$ contains both u and v.

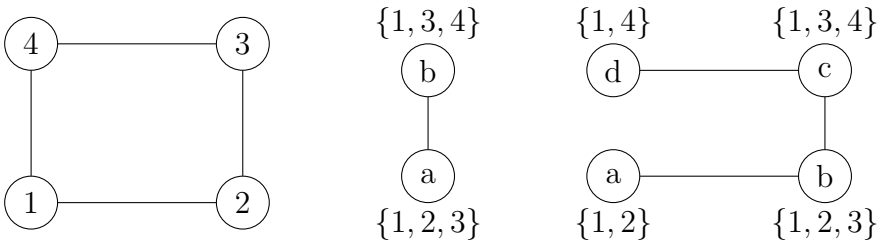
2. If s and t are vertices in T such that $u \in f(s)$ and $u \in f(t)$, then $u \in f(p)$ for every node p along the unique path connecting s and t in T.

The *width* of such a tree decomposition is the size of the largest set labelling the vertices of T. That is, given $T = (S, F)$ and f, the tree width is $\max_{s \in S} |f(s)|$.

The trivial tree decompostion of any graph $G = (V, E)$ is a tree with a single node s such that $f(s) = V$. This has width $|V|$.

If $G = (V, E)$ is already a tree, we can obtain a decomposition of width 2 as follows. Take $T = (V, E)$. Root the tree at r and set $f(r) = \{r\}$. For each non-root node u, set $f(u) = \{u, v\}$, where v is the parent of u—that is, the neighbour of u on the path from u to r.

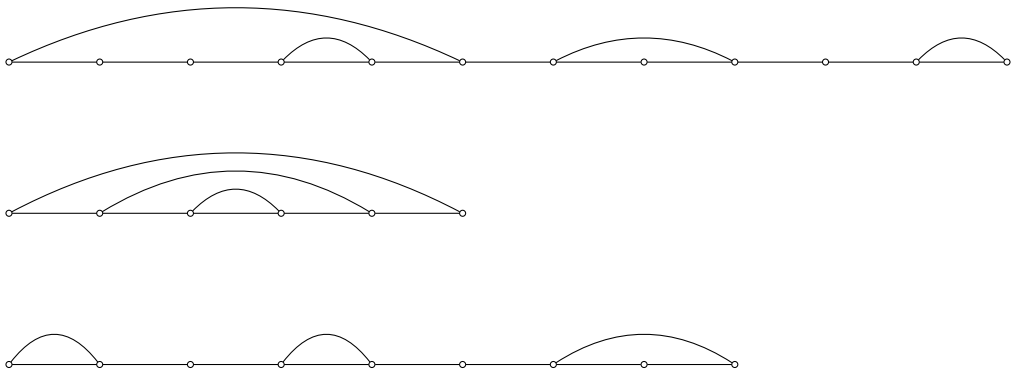
Here are two tree decompostions of the cycle on 4 vertices. The width of both of these decompositions is 3.



[1]Problem formulated by K. Narayan Kumar.

In order to simplify your life, since we are rather kind, we restrict ourselves to what we call *parenthesis graphs*. These are graphs described using well-bracketed words consisting of the letters {(,),.}. Here are some examples of such expressions: `(..())(.).()`, `((()))`, `().().(.)`.

Each such expression represents a graph. Suppose the expression has N letters. Create N vertices $1, 2, \ldots, N$ and add an edge from i to $i+1$ for each $1 \leq i < N$. The only other edges are those relating a pair of matched parentheses. If the letter at i is a `'('` and the matching `')'` appears at position j then there is an edge from vertex i to vertex j. Here are the graphs corresponding to the three expressions given above:



Your task is to write down a tree decomposition with as small a width as possible for a parenthesis graph presented to you as an expression. The number of vertices in your tree decomposition must be less than or equal to $2N$.

This is an output only task. You will be supplied with 20 inputs, each containing one such expression. You must submit 20 outputs corresponding to these inputs.

Input format

The first line of input contains a single integer N giving the length of the expression. This is followed by a line with N letters each of which is either `'('` or `'.'` or `')'`. You may assume that this expression is well-bracketed.

Output format

The first line should contain an integer M, $1 \leq M \leq 2N$, the number of vertices in the tree in your tree decomposition. This should be followed by $M-1$ lines each containing two integers in the range $1, 2, \ldots M$ describing the edges of the tree. This should be followed by M lines describing the labels of the vertices $1 \ldots M$ in that order. The first number on each such line should be the number of vertices k in the label and this should be followed by k integers in the range $1 \ldots N$ describing the label.

Test Data

You may assume that $2 \leq N \leq 1000$.

Evaluation

For each input, if the proposed tree decomposition is not a valid one no marks will be awarded. Otherwise, the score awarded will be determined by comparing the width of your tree decomposition to those in other correct contestant submissions as well as our model solution. All test cases have equal weightage.

Sample input

```
().().(.)
```

Sample output

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

Time and memory limits

This is an output only task, so there are no time or memory limits.

Submitting your solutions

The input files are called `width0.in`,...,`width19.in`. These files will be in a folder named `width` in your home directory. Your output files should be named `width0.out`, ... `width19.out` and should also be in the folder `width` in your home directory.

A program `width-submit-check.sh` is provided with the input files. Running this as `./width-submit-check.sh` checks that the output files are in the correct folder and reports some basic statistics about this. Note that this does *not* check the correctness of your output.

You may move the inputs and the program `width-submit-check.sh` elsewhere if you wish, but your outputs *must* be in the folder `~/width` for them to be evaluated. Any output file whose size is greater than $8,500,000$ bytes will not be evaluated. The program `width-submit-check.sh` reports the sizes of your output files.