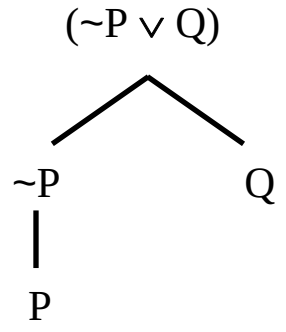
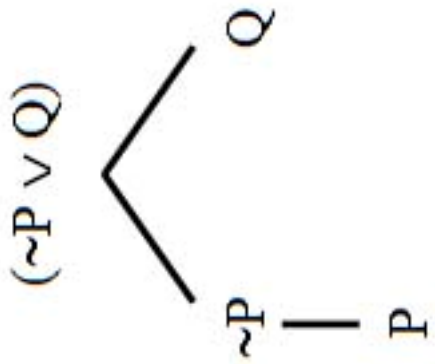
Typing Construction Trees in WebCT

Construction trees pose the same problem in WebCT as argument diagrams did: while they're easy enough to write out on paper, they're tough to type into a WebCT assignment. But the solution worked out for argument diagrams will apply as well to construction trees. Here is an example.

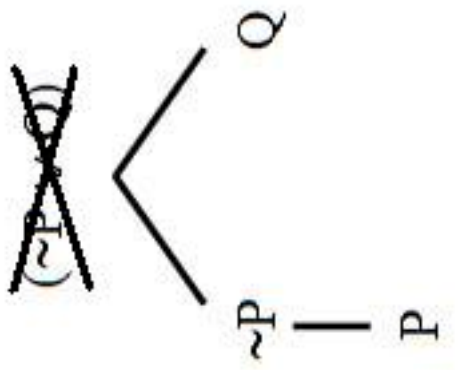


Having worked out this tree on paper, the **first step** in WebCT will be to turn the tree onto its side (rotating it 90 degree counterclockwise), so the completed sentence is on the **left**.



Second, with the tree on its side, we pick out each move where (reading from left to right) a sentence is broken down into its immediate part(s). We mark such a (branching or non-branching) step like so: "**= =>**".

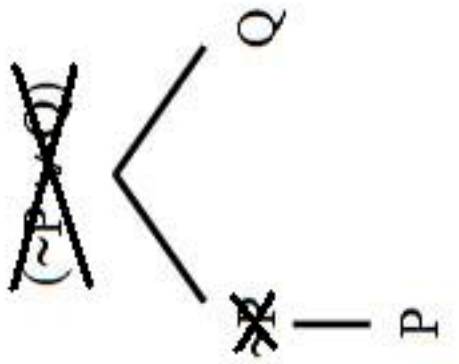
The whole sentence "(~P ∨ Q)" breaks down into its two immediate parts "~P" and "Q" – the two sentences forming the **scope** of the main connective, "∨".



In WebCT we type out this ‘disassembly’ into parts like so.

```
(~P \/ Q)  = => ~P, Q
```

Next, “~P” breaks down into its one scope sentence “P”.



We represent this breakdown step the same way.

```
(~P \/ Q)  = => ~P, Q

     ~P    = => P
```

With only atoms remaining, we have completed our ‘sideways’ construction tree’.