**Question 1**

1. What are the possible solutions for the problem?
2. Construct a tree step by step. Each tree node has two elements and two pointers: node number, attack power, first child and next sibling. Following the input, we construct tree from node 1. For input u, v, w, we search node u in the existing tree, and adding node v with attack power w to the node u. Then traverse the tree and calculate the minmax hp.
3. Using an array to store node. Each tree node has only one element and two pointers: attack power, first child and next sibling. At first, we construct all node into the array. We can visit each node by array index. Then following the input to construct tree. For input u, v, w, we find the node in the array and let u be v’s parent and assign w to v’s attack power. Then calculate hp grom root to each leaf and return the minmax hp.
4. How do you solve this problem?

**Tree Node**: 1 element and 2 pointers, attack power, first child and next sibling. The default number of attack power is zero, default pointer is *nullptr*.

**Initialize**: we first assign an array with n elements. Then we fill the array with n node.

**Construct tree**: for the input u, v, w, by accessing array[u-1] we can reach the node u, then we add v to the u. We first check whether node u’s first child is empty or not. If it is empty, we directly add v to u. If it is not empty, we move to u’s first child, and check whether its next sibling is empty or not until we find a empty next sibling and add v to it.

**Find minmax hp**: we have a global variable hp\_max to store the minimax hp. we can actually find that the MP before passing one edge is exactly the depth of that node. So, we just need to traverse the tree from the root to each leaf, which is node 1 with zero hp, and calculate each node’s hp and update the hp\_max. After we traverse the whole tree, we output the hp\_max.

1. Why is your solution better than others?

If we use the solution 1 as I mentioned above, it will lead to runtime error when constructing the tree, since there might exist two nodes that are not in the tree yet but with edges. Also, each time it inserts the node into the tree it needs to find the node in the tree from root. My solution assigns all nodes at first to the array and can be accessed by index, which is efficient. My solutions tree node has 1 element and 2 pointers, no need to store the node number since each node can be accessed by index.