Там, в краю далёком…

Решение.

Буду в это раз краток. Задача решается с помощью двух проходов поиском в глубину. Сложность решения – O(N), где N – количество вершин.

Зафиксируем произвольно некоторую вершину дерева, назначим её корнем и обозначим root. Дерево станет корневым, и дуги получат ориентацию. Будем, как обычно, говорить, что направление движения от корня к листьям – это направление вниз.

Первым поиском в глубину вычисляем для каждой вершины v длину самого длинного пути, начинающегося в этой вершине и идущего вниз, а также длину самого длинного пути из этой вершины и идущего вниз, но не имеющего с первым путём общих вершин (кроме v, разумеется). Назовем эти величины max1(v) и max2(v), соответственно.

Из корня все пути ведут вниз, поэтому для корня самая удалённая вершина находится на расстоянии max1(root).

Для всех остальных вершин путь до самой удалённой вершины может начинаться либо с дуги, ведущей вниз, либо с дуги, ведущей вверх, в предка этой вершины. Обозначим саму вершину v, её предка – u, а длину дуги между ними – d. Длину самого длинного пути, идущего из v вниз, мы знаем, она равна max1(v).

Осталось найти max3(v) - длину самого длинного пути, начинающегося дугой v-u. Если самый длинный путь из u вниз не проходит через вершину v, то max3(v) = d + max1(u), в противном случае max3(v) = d + max2(u). Отличить эти два случая несложно: можно просто вместе с max1(u) хранить и вторую вершину пути, реализующего эту длину; можно и не хранить эту вторую вершину, ведь условие «самый длинный путь из вершины u вниз проходит через вершину v» равносильно соотношению max1(u)=d+max1(v).

Теперь, когда мы вычислили max3(v), прежде чем обрабатывать потомков вершины v, изменим max1(v) и max2(v):

если max3(v) > max2(v), то заменим max2(v) на max3(v). Здесь потребуются некоторые пояснения. Если max3(v)<=max1(v), то в max2(v) оказывается длина второго по длине пути, начинающегося в вершине v; если же max3(v)>max1(v), то мы и max1(v) заменяем на max3(v), и это не приведёт в дальнейшем к ошибкам – в этом случае самый длинный путь из вершины v проходит через вершину u (т.е. начинается «вверх») и значение max2(v) при обработке сыновей вершины v нам не понадобится.

В результате в max1(v) хранится интересующая нас величина – длина самого длинного пути, начинающегося в вершине v, а мы можем переходить к обработке потомков v.

Для хранения дерева использовались связные списки, реализованные на массивах. Такой способ хранения деревьев (и других подобных структур) очень удобен и эффективен. Имеется масса описаний и примеров такого хранения, поэтому я не буду составлять ещё одно описание, а просто выложу в учебных материалах главу из книги

Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов, 536 стр. М.: Мир, 1979.

В этой главе разбирается очень изяшный способ решения задачи об объединении непересекающихся множеств. Всячески советую прочитать эту небольшую (5 страничек) статью – очень красиво. А сцепляемые непересекающиеся множества – это ведь, фактически дерево (если взглянуть на дерево, как на множество разрозненных вершин, которые мы постепенно соединяем путями-дугами, не допуская появления циклов). Вл всяклм случае, хранение данных очень похожее.

Впрочем, я обещал быть кратким ☺ ...