Analysis

1 Introduction

The problem statement in short is:

We have a tree in which every vertex has a weight. We want to decompose the tree into vertical chains, such that every vertex belongs to exactly one chain and the sum of "values" of the chains is maximum. The value of a chain is defined as $W_{max} - W_{min}$, where W_{max} is the maximal weight of a vertex in the chain and a W_{min} is the minimal weight.

We want our solution to be O(N log N) or O(N) in time, where N is the number of vertices in the tree.

2 Observations

To solve the problem we first need to make some observations. The most important one is that at least one of the following is true for every chain:

- The chain should start from the vertex with the largest weight and finish on the vertex with the smallest weight from this path.
- The chain should start from the vertex with the smallest weight and finish on the vertex with the largest weight from this path.

This is true, because if we have a path not satisfying one of the above conditions we can divide this path into several smaller ones without decreasing the initial value.

The second observation is that there exists an optimal decomposition into chains such that the weights of the vertices in each chain are either increasing or decreasing. This observation is again true because if there was a chain not satisfying this, we could have divided it into several smaller ones without decreasing its initial values.

3 Main part

Using the above observations we can solve the problem in linear time. The author's solution uses only the first observation. Also there exists a solution which is easier to be implemented but it uses both observations. Both solutions are linear in time.

To get partial points one could have implemented a brute force for 10 points. To get 40 points one could implement a solution quadratic in time.

3.1 Solution for 40 points

We will solve the problem with dynamic programming.

Our state will be dp_u which is the maximal sum of values of chains in a decomposition of the subtree rooted at vertex u. We know that there is a chain starting at vertex u and finishing in a vertex in its subtree. Lets denote this vertex as v and the set of vertices which belong to the path from u to v as S. Also lets denote by C_i the sum of the dp values of the children of u ($C_i = \sum_{x \in children(i)} dp_x$). From here we get that for the chain starting at u and finishing at v, the value of dp_u will be $\sum_{x \in S} C_x - \sum_{x \in S / \{u\}} dp_x + max_{x \in S} w_x - min_{x \in S} w_x$. And so we can find dp_u by checking this value for all vertices v in the subtree of u and then by getting the maximal value. This solution is $O(N^2)$. If implemented well it can get even 50 points.

3.2 Author's Solution

We will use dynamic programming to solve the problem. With each vertex we will associate 3 values:

- $dp_{u,0}$ – maximum sum of values to cover the subtree of u.
- $dp_{u,1}$ – maximum sum of values to cover the subtree of u, if we have an unfinished path, which includes the root and a fixed maximum (the value of the maximum has been added in advance).
- $dp_{u,2}$ - maximum sum of values to cover the subtree of u, if we have an unfinished path, which includes the root and a fixed minimum (the value of the minimum has been substracted in advance).
- The answer to the problem will be $dp_{1,0}$.

We will use DFS, starting from the root (vertex number 1). After going through all of the children of a given vertex, we will calculate the three values for the vertex. Let us firstly calculate $dp_{u,1}$. The evaluation of $dp_{u,2}$ will use an analogical approach as the one for $dp_{u,1}$.

When we calculate $dp_{u,1}$, we have two possibilities:

- A new path starts from vertex u.
- A path, which until now finished in one of the children of u, continues through u.

In the first case, the value will be $W_u + \sum_{v \in children(u)} dp_{v,0}$.

In the second case, we need to choose a child of u, the path from which we want to continue. Let $S = \sum_{v \in children(u)} dp_{v,0}$. If the chosen child of u is p, the value of $dp_{u,1}$ will be $S - dp_{p,0} + dp_{p,1}$.

This way we can find the maximum value for $dp_{u,1}$ and $dp_{u,2}$. Now in order to calculate $dp_{u,0}$, we once again have two possibilities:

- To finish a path, where u is the vertex with maximum weight.
- To finish a path, where u is the vertex with minimum weight.

So we get $dp_{u,0} = \max(dp_{u,1} - W_u, dp_{u,2} + W_u)$.

This solution has a complexity of O(N) if it's implemented by the described above way.

3.3 Second Solution (Encho Mishinev)

We will once again use dynamic programming. However, we will use the second observation – that an optimal decomposition into paths exists, such that the values in each path are either only increasing or only decreasing. If we decompose the tree in such fashion, we can say, that the weight of a path is the sum of the absolute differences between each two consecutive vertices. For increasing or decreasing paths this definition is equivalent to $W_{max} - W_{min}$ With each vertex we will associate 2 values:

- $F_{v,0}$ – maximum sum of values for the covering of the subtree of u, if the path starting from u is with decreasing weights.
- $F_{v,1}$ - maximum sum of values for the covering of the subtree of u, if the path starting from u is with increasing weights.

The answer to the problem will then be $\max(F_{1,0}, F_{1,1})$

The calculation of $F_{v,0}$ and $F_{v,1}$ is analogical, so we will focus only on the calculation of $F_{v,0}$:

Let $S = \sum_{x \in children(v)} \max(F_{x,0}, F_{x,1})$.

Because a decreasing path starts from v, we have top chose to which child it continues. If this child is u ($W_u \leq W_v$), the value will be $S - \max(F_{u,0}, F_{u,1}) + (W_v - W_u) + F_{u,0}$.

We have to remember the case, where v is a path, consisting only of one vertex and thus the value is simply S.

This way with $O(1)$ we can evaluate the value for a fixed child, which allows us to calculate all values in the tree with a complexity of $O(N)$.

4 Additional information

You can try to solve a more difficult version of the problem. In it once again you need to maximize the sum of the paths, where the value of a path is once again the difference between the largest and smallest weight in the path, however we have one more condition – each path has to have length ≤ K, where K is a given integer.

This version of the problem can be solved with a complexity of $O(N \log N)$, using a segment tree, or with a complexity of $O(N \log^2 N)$, using the technique, described here: http://codeforces.com/blog/entry/44351.