Extrema of piecewise linear functions

One possible approach to the solution is outlined here.

Representation

First, a function representation is needed. The representation must be such that it is easy to derive from it all local extrema of the function. Moreover, the representation should be general, that is, not only the input function but any function of the same type can be represented in the same way. The type of the function(s) is piecewise linear function, therefore it is natural to think about this kind of function as a sequence of segments in the plane which form the graph of the function. The endpoints of the segments are precisely those points of the functions in which the derivative is undefined. For example, the graph of the function depicted in the Image 1 is made of four segments. As the right endpoint of each segment coincide with the left endpoint of the next segment, it is also possible to represent the graph by the sequence of the endpoints together with the slope of the first and last (infinite) segments. When we have this or other similar representation it is obviously trivial to extract from it, in one pass, all local extrema of the function.

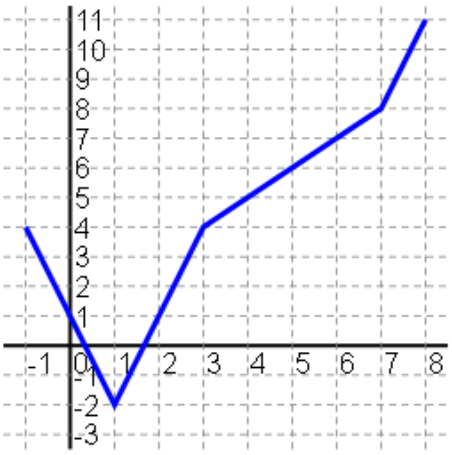

Image 1. Graph of the function defined by the term $||x-3|-2\cdot x| + |x-7| + x-9$.

Recursive nature of the problem

We can observe that there are only two "elementary" piecewise linear functions:
$f_1(x) = x$, $f_2(x) = constant$.
Any other more complex piecewise linear functions can be recursively constructed from these by repeated application of just three operations:
A. Sum of two functions.
B. Multiplying a function by an elementary constant function.
C. Absolute value of the function.
All these operations are unary or binary. We can associate with any given piecewise linear function a binary tree which reflects the way in which the function is constructed form the elementary functions using operations A, B, C. The idea is illustrated in the image 2. The leaves correspond to the elementary functions. Each internal node corresponds to some more complex function which is produced by the operation A or B or C (denoted at the node) applied to the function(s) corresponding to the child/children of the node. The root corresponds to the whole given function.

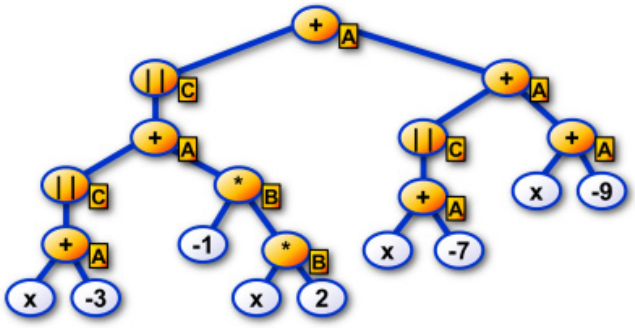

Image 2. Tree associated with the function $f(x) = ||x-3|-2\cdot x| + |x-7| + x-9$.

Building the solution

The representation of any tree leaf is very simple, it is just one (infinite) segment. For each of the three operations A, B, C we need a method which builds a function representation corresponding to a particular node from the representation(s) corresponding to the child/children of the node. For example, method dealing with operation B is very simple:
Let $[x_1, y_1], ..., [x_k, y_k], s_0, s_k$ be the representation of a function corresponding to one child of the node.
Let c_0 be the representation of the constant function corresponding to other child of the node.
Then the representation of function corresponding to the given node is
$[c_0 \cdot x_1, c_0 \cdot y_1], ..., [c_0 \cdot x_k, c_0 \cdot y_k], c_0 \cdot s_0, c_0 \cdot s_k$.
(The values $[x_1, y_1], ..., [x_k, y_k]$ are the endpoints of the segments of the graph, the values s_0, s_k are the slopes of the leftmost and rightmost (infinite) segment of the function graph.)
The methods dealing with the remaining two operations, A and C, should be only slightly more involved.

The whole solution may run in a few phases:
1. Parse the input term and build the tree associated with the function defined by the input term.
2. Create representations (single infinite segments) for all leaves in the tree.
3. Recursively create representations for all nodes in the tree.
4. Use the root representation (which corresponds to the input function) to find the extrema of the input function.

Notes

1. The solution strategy presented here does not pretend to be the only one or the best one. You are encouraged to think of other ways of solution too.
2. There might be different, equally suitable or better, function representations.
3. The tree is not defined unambiguously, you can build it in a slightly different way. Also, you may consider slightly different set of operations (depending on the order of operands etc.), elementary functions etc.
4. The tree associated with the function needs not to be physically built in the memory. It is also possible to merge parsing process with the tree traversal and computing the functions representations. In such case it might be handy to use an appropriately configured stack (stack of function representations or graph representations etc.)