Within this case, f(x) can be represented in a decision tree. At the heart of its behavior, decision trees are able to represent and emulate any logic operation such as a boolean function. Now, considering the 2^d inputs, which is a significant amount of sampled data, we would still be able to build a decision tree recursively by splitting the sampled data into subsets of data and so on and so forth. This gives rise to the series of decisions that the algorithm itself will make upon the test data cascading down the nodes of the tree arriving at a specific result. The essence of it is to understand the mere behavior of the function given whatever input it is allowed to take it.

An example to illuminate the characteristics of a decision tree is the AND operation, where there are two inputs, A and B, that determine the result of the function. We begin at the root node, which observes the value of A (0, 1), and based on that value, the tree will branch to a new node that observes the value of B (0, 1). At each node a decision is made based on a condition, to effectively check the data value at a given node to dictate where the branch will go. So, if A==1 then branch to node2 containing if B is 0 or 1, after seeing that B == 1 the output will result in True.