Deep Reverse Recurrence Relation

The recurrence relation for deep reverse relies on the following ideas: define \( n \) to be maximum length of any list, and \( d \) to be the depth of the nested lists. Let the list that we need to deep-reverse be called \( L \). In the worst case, any sublist nested within \( L \) has length exactly equal to \( n \) and the depth from each element is maximal. To analyze the run time of deep-reverse, we first need to know the run time of simply reversing a list of elements. Depending on implementation, reverse can run in either \( O(n^2) \) or \( O(n) \) time, so we will show both. Now, we observe that if we increase the maximum number of elements in any list, then the runtime will increase because there are more elements to consider. Also, if the depth increases, then we also must do more computation. This means that our recurrence relation should depend on both \( n \) and \( d \) and be of the form \( T(n,d) \). Now, when we recursively call a sublist of the top layer, we observe that the worst case is that the sublist has \( n \) elements as well, but the depth is now \( d - 1 \). There are at most \( n \) of these subcalls made from the top layer. Lastly, the top layer of elements also needs to be reversed, which after making the recursive subcalls, takes an addition reverse step. For our base case, if we have a list of \( n \) elements with no sublists, then we are in a list of depth 1. So, our base case becomes \( T(n,1) \), which just takes as much time as a single reverse. So, let us assume that reverse is implemented in \( O(n) \) time, and we get the following recurrence relation, which we then solve:

\[
T(n, d) = nT(n, d - 1) + O(n), T(n, 1) = O(n)
\]

\[
T(n, d - 1) = nT(n, d - 2) + O(n)
\]

\[
\Rightarrow T(n, d) = n(nT(n, d - 2) + O(n)) + O(n)
\]

\[
T(n, d) = n^2T(n, d - 2) + O(n^2) + O(n)
\]

Now we see a pattern, so consider the recurrence relation after \( k \) steps:

\[
T(n, d) = n^kT(n, d - k) + O(n^k) + O(n^{k-1}) + ... O(n)
\]

Now, to reach our base case, let \( k = d - 1 \):

\[
T(n, d) = n^{d-1}T(n, 1) + O(n^{d-1}) + O(n^{d-2}) + ... + O(n)
\]

\[
T(n, 1) = O(n) \Rightarrow T(n, d) = n^d + O(n^{d-1}) + O(n^{d-2}) + ... + O(n)
\]

\[
T(n, d) = O(n^d)
\]

So, deep-reverse runs in \( O(n^d) \) time, assuming reverse is implemented in \( O(n) \) time. For completeness, the derivation for the case where reverse in \( O(n^2) \) time is also presented. The changes are the base case and the extra time taken per step are both taken up a power to \( O(n^2) \).

\[
T(n, d) = nT(n, d - 1) + O(n^2), T(n, 1) = O(n^2)
\]

\[
T(n, d - 1) = nT(n, d - 2) + O(n^2)
\]

\[
\Rightarrow T(n, d) = n(nT(n, d - 2) + O(n^2)) + O(n^2)
\]

\[
T(n, d) = n^2T(n, d - 2) + O(n^3) + O(n^2)
\]

...\[
T(n, d) = n^kT(n, d - k) + O(n^{k+1}) + O(n^k) + ... O(n^2)
\]

\[
k = d - 1 \Rightarrow T(n, d) = n^{d-1}T(n, 1) + O(n^d) + O(n^{d-1}) + ... + O(n^2)
\]

\[
T(n, 1) = O(n^2) \Rightarrow T(n, d) = n^{d+1} + O(n^d) + O(n^{d-1}) + ... + O(n^2)
\]

\[
T(n, d) = O(n^{d+1})
\]

This shows the difference in the derivations for both cases and we are done.