1. if implicit stack space does not count, only need to do this for each level separately, each time use dfs to visit the top i levels of the tree. time $\sum_{i=1}^{\log n} 2^i = O(n)$. space $O(1)$ (without stack).
2. build the next of the $(i+1)$-th level from the next of the i-th level, by traversing from left to right on the i-th level. $O(n)$ time, $O(1)$ extra space (the actual space complexity is $O(n)$, because we're using the stored next).

note. we can use $O(\log \log n)$ space to simulate the dfs stack and dfs the whole tree in $O(n)$ time. only store a pointer at the 1.5^i-th level from the bottom of the dfs stack, if we need to go up during the dfs, instead go down from the nearest stored pointer above (the dfs path can be stored by $O(\log n)$ bits, i.e. 1 word, indicating going left/right at each level). the running time is $\sum_{i=1}^{\log n} n \cdot (\frac{1.5}{2})^i = O(n)$.
update. we can use $O(\log^{(c)} n)$ (iterative log) space and $O(n)$ time to simulate dfs, where c is any constant. there are $\frac{n}{\log n}$ points at the $\log \log n$-level from the bottom, we go down from the root to reach them, each using $O(\log n)$ time, in total $O(n)$. recursively do this, there are $\frac{n}{\log \log n}$ points at the $\log \log \log n$-level from the bottom, we go down from the stored pointer at the $\log \log n$-level from the bottom. recursively do this c times.
we can further get $O(\log^* n)$ space and $O(n)$ time, by performing $c = O(\log^* n)$ recursions, and let the total time of the i-th recursion be $n/2^{c-i}$ (e.g. the first recursion is at the $(\log \log n + c)$-level from the bottom, and the running time for that recursion is $O(2^{\log n - \log \log n - c} \cdot \log n) = O(\frac{n}{2^c})$), which is a geometric series. The recursion analysis will not change much.

References