Complexity Analysis

Only two methods in the algorithm are worth mentioning: the `isDivisible`- and `expand`-methods. Appending and removing digits, checking the total number of digits, and checking if some digit has already been used are all constant time operations.

Considering addition, multiplication, and computation of a remainder as constant time operations, the `isDivisible`-method uses a constant number of operations for each digit of the input. That is, the method has complexity $\mathcal{O}(k)$, where k is the 'length' of the input.

To analyse the recursive `expand`-method, note that it corresponds to a depth-first search of a tree containing the possible strings of digits. See for instance figure 1, where the left tree shows the strings of digits when 1 is expanded, and the right tree shows the order in which the nodes are visited.

For ease of notation, the root of the tree is considered to be the first level of the tree, rather than the zeroth. For each subsequent level k, notice that each node at level $k-1$ will expand to $n-k$ nodes at level k. This implies that the total number of nodes at level k, where $2 \le k \le n-1$, is given by

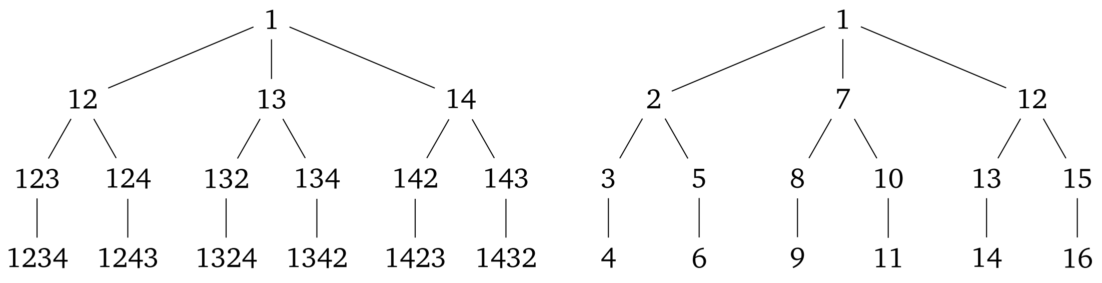
$$(n-2)(n-3)\cdots(n-k) = \frac{(n-2)!}{(n-k-1)!}.$$

Each node at level k has a computation cost of $\mathcal{O}(k)$ from the `isDivisible`-method, which dominates the constant complexity of appending and removing digits. Thus, the total computation cost for the entire tree is bounded by

$$\sum_{k=2}^{n-1} \frac{(n-2)!}{(n-k-1)!} k \le (n-2)! \sum_{k=2}^{n-1} k = (n-2)! \left(\frac{(n-1)n}{2} - 1 \right) = \mathcal{O}(n!).$$

As if that is not enough, such a tree must be computed for each digit, raising the total computation time used by the algorithm to $\mathcal{O}(n!n)$.

It must be remarked, however, that the entire tree will not be searched in practice. At the second level of the tree, the resulting integer must be divisible by 2; i.e. the integer must be even. Depending on the starting digit, this rules out all digits of a certain parity, and the corresponding branches of the tree will never be searched. Taking the situation in figure 1 as an example, only the branch corresponding to $(13)_5$ is searched, since $(12)_5 = (7)_{10}$ and $(14)_5 = (9)_{10}$ are both odd numbers. More generally, this argument applies to any power of 2.



Figur 1: Tree structure when expanding 1 in base-5