1s for all

Philip Stanton

April 2023

1 Solution

The basic approach to solve the problem is to implement a dynamic programming solution. First, lets formulate the problem in a way that leads to this idea.

Lets say you want to compute the complexity for an integer n. We'll define $c(n)$ to be the function that computes the complexity of n

It follows that you want to find $a * b = n$, such that $c(a) + c(b)$ is minimized. We also need to consider the case of $a + b = n$ because there is no guarantee that the optimal solution comes from multiplication. We need to minimize $c(a) + c(b)$ in this case as well. After this, we can check to see if concatenating two integers leads to a better result for $c(n)$

This leads to the idea of a solving with a dynamic program, as the formulation implies that one will be solving many overlapping sub-problems when computing $c(n)$. If we know $c(a)$ and $c(b)$ $\forall a, b < n$, then when computing $c(n)$, we can efficiently check which a and b are optimal.

We now define our dynamic program:

Let $\mathit{opt}[i]$ = min number of ones to represent i using only +,*, and ()

The base cases are $\mathit{opt}[0]...\mathit{opt}[5]$ where the optimal solutions are themselves.

Now,

$$\begin{aligned}
\mathit{opt}[i] = \min\{ &\min\{\mathit{opt}[i/k] + \mathit{opt}[k] + \mathit{opt}[i\%k] : \forall 2 \leq k < i/2\}, \\
&\min\{\mathit{opt}[k] + \mathit{opt}[i-k] : \forall i/2 < k < i\}, \\
&\mathit{concat}(i)\}
\end{aligned}$$

and we construct $\mathit{opt}[n]$ using a bottom up approach. $\mathit{concat}(i)$ will simply split i by its digits and check if any concatenation is cheaper than the results we got from the first optimizations.

This gives an $O(n^2)$ algorithm, as when computing $\mathit{opt}[i]$, you must iterate through all $2 < k < i$, and we do this n times to get $\mathit{opt}[n]$. It should be noted that $\mathit{concat}(i)$ is completely dominated by this, as it runs in linear time with respect to number of digits for i