CSE421: Design and Analysis of Algorithms Feb 07, 2018

Lecture 15 Integer Multiplication

Lecturer: Shayan Oveis Gharan Scribe:

We didn't discuss the details of the correctness proof of Divide and Conquer algorithms in class. This is mainly because we didn't have enough time in class.

So, here I will explain a correctness proof for the integer multiplication algorithm. For divide and Conquer problems, it is typically convenient to use induction to prove correctness; this is mainly due to a recursive structure of these problems. Furthermore, since we typically reduce a problem instance of size n to several instances of size much smaller than n we are indeed using *strong* induction.

Remember that strong induction is not much different from ordinary induction. The main difference is that we assume $P(k)$ holds for all $k < n$ and we use them to prove $P(n)$. So, here is a general structure of a structural induction:

Base Case: $P(k)$ holds for all $k \leq c$ where c is an absolute constant like 10.

IH: For some $n > c$, for all $1 \leq k < n$, $P(k)$ holds true.

IS: Goal: We prove $P(n)$.

Note that in most applications of the Divide and Conquer approach there is a trivial slow algorithm and we typically use this method to speed up that algorithm. It is more convenient to use the trivial slow algorithm when the input size is less than a large constant c. Note that essentially all algorithms take $O(1)$ time if the input size is an absolute constant c. Therefore, we can take c to be large enough in order to avoid edge cases when we use induction. Correspondingly, the base case of our strong induction would be that $P(j)$ is true for all $j \leq c$.

Now, let us use this to rigorously prove the correctness of our integer multiplication algorithm. Here is the algorithm:

---

function MULT(x, y)
  Let n be the number of bits of x, y
  if $n \leq 10$ **then**
    Use the simple $O(n^2)$-time algorithm to multiply x, y and return the product
  end if
  Let x_1 be the left $\lfloor \frac{n}{2} \rfloor$ and x_0 be the right $\lceil \frac{n}{2} \rceil$ bits of x, i.e., $x = 2^{\lceil n/2 \rceil} x_1 + x_0$.
  Similarly, let y_1, y_0 be the left $\lfloor \frac{n}{2} \rfloor$ and the right $\lceil \frac{n}{2} \rceil$ bits of y, i.e., $y = 2^{\lceil n/2 \rceil} y_1 + y_0$.
  $A \leftarrow \text{MULT}(x_1, y_1)$.
  $B \leftarrow \text{MULT}(x_0, y_0)$.
  $\alpha \leftarrow x_0 + x_1$.
  $\beta \leftarrow y_0 + y_1$.
  $C \leftarrow \text{MULT}(\alpha, \beta)$.
  Return $2^{2\lceil n/2 \rceil} A + 2^{\lceil n/2 \rceil}(C - A - B) + B$
end function

Now, that we have an algorithm we use strong induction to prove its correctness. Let $P(n)$ be "for any two n bit integers x, y, MULT(x, y) computes the product of x, y correctly.

Base Case: $P(j)$ is true for all $j \leq 10$. This is obviously true because the algorithm uses the naiive $O(n^2)$ time algorithm that we learn in elementary school for $j \leq 10$.

IH: Suppose for some $n > 10$, $P(j)$ is true for all $1 \leq j \leq n-1$.

IS: We prove $P(n)$ is true. Fix two n bit integers x, y. Our goal is to show that MULT(x, y) correctly computes the product of x, y. Firstly of all, since $n \geq 2$, $\lceil n/2 \rceil \leq n-1$. So, since x_1, y_1, x_0, y_0 have at most $\lceil n/2 \rceil$ bits, by IH, we correctly compute A, B. Since we can compute addition in $O(n)$ time we can correctly compute α, β. Observe that α, β have at most $\lceil n/2 \rceil + 1$ bits which is at most $n-1$ for $n \geq 10$. Therefore, by IH we correctly compute C. Now observe that $x_1y_0 + x_0y_1 = C - A - B$. Therefore, $xy = 2^{2\lceil n/2 \rceil}A + 2^{\lceil n/2 \rceil}(C - A - B) + B$ which is correctly computed in our algorithm. This completes the proof of induction.

Complexity Note that all operations we do in MULT function is $O(n)$: We do 4 addition of at most n bit integers, 2 subtraction of at most n bit integers and two shifting operations. We also call MULT function 3 times on inputs of size at most $\lceil n/2 \rceil + 1$. Therefore, we can write the following recurrence relation for the running time:

$$T(n) = 3T(\lceil n/2 \rceil + 1) + O(n)$$

where $T(n) = O(1)$ for all $n \leq 10$. It follows by Master theorem that $T(n) \leq n^{\log_2 3}$.