3. INTEGER MULTIPLICATION I

Raveen de Silva, r.desilva@unsw.edu.au
office: K17 202
Course Admin: Anahita Namvar, cs3121@cse.unsw.edu.au

School of Computer Science and Engineering
UNSW Sydney

Term 3, 2021

Table of Contents

Table of Contents

Master Theorem

Setup

Let:

- $a \geq 1$ be an integer and and $b > 1$ be a real number;
- $f(n) > 0$ be a non-decreasing function defined on the positive integers;
- $T(n)$ be the solution of the recurrence

$$T(n) = a\,T(n/b) + f(n).$$

Define the *critical exponent* $c^* = \log_b a$ and the *critical polynomial* n^{c^*}.

Master Theorem

Theorem

1. If $f(n) = O(n^{c^*-\varepsilon})$ for some $\varepsilon > 0$,then $T(n) = \Theta(n^{c^*})$;

2. If $f(n) = \Theta(n^{c^*})$,then $T(n) = \Theta(n^{c^*} \log n)$;

3. If $f(n) = \Omega(n^{c^*+\varepsilon})$ for some $\varepsilon > 0$,**and** for some $c < 1$ and some n_0,
$$a\, f\,(n/b) \leq c\, f(n) \tag{1}$$
holds for all $n > n_0$,then $T(n) = \Theta(f(n))$;

Exercise

Prove that $f(n) = \Omega(n^{c^*+\varepsilon})$ is a consequence of (1).

Master Theorem

Theorem (continued)

4. If none of these conditions hold, the Master Theorem is NOT applicable.

   Often, the proof of the Master Theorem can be tweaked to (asymptotically) solve such recurrences anyway! An example is $T(n) = 2T(n/2) + n \log n$.

Master Theorem

Remark

- Recall that for $a, b > 1$, $\log_a n = \Theta(\log_b n)$, so we can omit the base and simply write statements of the form $f(n) = \Theta(g(n) \log n)$.

- However, $n^{\log_a x}$ is not interchangeable with $n^{\log_b x}$ - the base must be specified in such expressions.

Table of Contents

Master Theorem

Example 1

Let $T(n) = 4\,T(n/2) + n$.

Then the critical exponent is $c^* = \log_b a = \log_2 4 = 2$, so the critical polynomial is n^2.

Now, $f(n) = n = O(n^{2-\varepsilon})$ for small ε (e.g. 0.1).

This satisfies the condition for case 1, so $T(n) = \Theta(n^2)$.

Master Theorem

Example 2

Let $T(n) = 2\,T(n/2) + 5\,n$.

Then the critical exponent is $c^* = \log_b a = \log_2 2 = 1$, so the critical polynomial is n.

Now, $f(n) = 5\,n = \Theta(n)$.

This satisfies the condition for case 2, so $T(n) = \Theta(n \log n)$.

Master Theorem

Example 3

Let $T(n) = 3\,T(n/4) + n$.

Then the critical exponent is $c^* = \log_4 3 \approx 0.7925$, so the critical polynomial is $n^{\log_4 3}$.

Now, $f(n) = n = \Omega(n^{\log_4 3+\varepsilon})$ for small ε (e.g. 0.1).

Also, $af(n/b) = 3f(n/4) = 3/4\ n\ < c\,n = cf(n)$ for $c = .9 < 1$.

This satisfies the condition for case 3, so $T(n) = \Theta(f(n)) = \Theta(n)$.

Master Theorem

Example 4

Let $T(n) = 2\,T(n/2) + n\log_2 n$.

Then the critical exponent is $c^* = \log_2 2 = 1$, so the critical polynomial is n.

Now, $f(n) = n\log_2 n = \omega(n)$, so the conditions for case 1 and 2 do not apply.

However,

$$f(n) \neq \Omega(n^{1+\varepsilon}), \tag{2}$$

no matter how small we choose $\varepsilon > 0$.

Therefore the Master Theorem does **not** apply!

Master Theorem

Exercise

Prove (2), that is, for all $\varepsilon > 0$, $c > 0$ and $N > 0$ there is some $n > N$ such that

$$\log_2 n < c \cdot n^{\varepsilon}.$$

Hint

Use *L'Hôpital's rule* to show that

$$\frac{\log n}{n^{\varepsilon}} \to 0.$$

Table of Contents

Master Theorem

Suppose $T(n)$ satisfies the recurrence

$$T(n) = a\left[T\left(\frac{n}{b}\right)\right] + f(n) \tag{3}$$

However, the $T(n/b)$ term can itself be reduced using the recurrence as follows:

$$T\left(\frac{n}{b}\right) = a\,T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)$$

Substituting into (3) and simplifying gives

$$\begin{aligned} T(n) &= a\left[aT\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)\right] + f(n) \\ &= a^2\,T\left(\frac{n}{b^2}\right) + a\,f\left(\frac{n}{b}\right) + f(n). \end{aligned}$$

Master Theorem

We have now established

$$T(n) = a^2 \left[T\left(\frac{n}{b^2}\right) \right] + a\, f\left(\frac{n}{b}\right) + f(n). \tag{4}$$

But why stop there? We can now reduce the $T\left(n/b^2\right)$ term, again using (3):

$$T\left(\frac{n}{b^2}\right) = a\, T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right).$$

We now substitute this into (4) and simplify to get

$$\begin{aligned} T(n) &= a^2 \left[a T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right) \right] + a\, f\left(\frac{n}{b}\right) + f(n) \\ &= a^3\, T\left(\frac{n}{b^3}\right) + a^2\, f\left(\frac{n}{b^2}\right) + a\, f\left(\frac{n}{b}\right) + f(n). \end{aligned}$$

We can see a pattern emerging!

Master Theorem

Continuing in this way, we find that

$$\begin{aligned} T(n) &= a^k T\left(\frac{n}{b^k}\right) + a^{k-1} f\left(\frac{n}{b^{k-1}}\right) + \ldots + a f\left(\frac{n}{b}\right) + f(n) \\ &= a^k T\left(\frac{n}{b^k}\right) + \sum_{i=0}^{k-1} a^i f\left(\frac{n}{b^i}\right). \end{aligned}$$

We stop when $k = \lfloor \log_b n \rfloor$, since this gives $n/b^k \approx 1$.

$$T(n) \approx a^{\log_b n} T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right).$$

Master Theorem

Now we have

$$T(n) \approx a^{\log_b n}\, T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i\, f\left(\frac{n}{b^i}\right).$$

We can use the identity $a^{\log_b n} = n^{\log_b a}$ to get:

$$T(n) \approx n^{\log_b a}\, T(1) + \underbrace{\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i\, f\left(\frac{n}{b^i}\right)}_{S}. \tag{5}$$

Importantly, we have not assumed anything about $f(n)$ yet! We will now analyse the sum S in the simplest case of the Master Theorem, namely Case 2.

Master Theorem: Case 2

Suppose $f(n) = \Theta\left(n^{\log_b a}\right)$. Then

$$
\begin{aligned}
S &= \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) \\
&= \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \, \Theta\left(\left(\frac{n}{b^i}\right)^{\log_b a}\right) \\
&= \Theta\left(\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a}\right),
\end{aligned}
$$

using the sum property and scalar multiple property.

Master Theorem: Case 2

$$S = \Theta\left(\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a}\right)$$

$$= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \left(\frac{1}{b^i}\right)^{\log_b a}\right)$$

as $n^{\log_b a}$ is common to every term of the sum,

$$= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{b^{\log_b a}}\right)^i\right).$$

Master Theorem: Case 2

$$S = \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{b^{\log_b a}}\right)^i\right)$$

$$= \Theta^{\log_b a} left(\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{a}\right)^i$$

as $b^{\log_b a} = a$,

$$= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} 1\right)$$

$$= \Theta\left(n^{\log_b a} \lfloor \log_b n \rfloor\right).$$

Master Theorem: Case 2

Finally, we return to (5). Substituting our result for S gives

$$\begin{aligned} T(n) &\approx n^{\log_b a}\, T(1) + \Theta\left(n^{\log_b a} \lfloor \log_b n \rfloor\right) \\ &= \Theta\left(n^{\log_b a} \log n\right) \\ &\qquad\qquad \text{as logarithms of any base are equivalent,} \end{aligned}$$

completing the proof.

Master Theorem: Other cases

- The proof of Case 1 is very similar to the above. The main difference is that $\sum 1$ is replaced by $\sum (b^{\varepsilon})^i$, forming a geometric series, which can be summed using the identity

$$1 + r + r^2 + \ldots + r^{k-1} = \frac{r^k - 1}{r - 1}.$$

- In Case 3, we need to prove that $T(n) = \Theta(f(n))$, that is, both:
  - $T(n) = \Omega(f(n))$, which follows directly from the recurrence $T(n) = a\,T(n/b) + f(n)$, and
  - $T(n) = O(f(n))$ (not as obvious).

Master Theorem: Case 3

Exercise

Prove that in Case 3, $T(n) = O(f(n))$.

Hint

You will need to bound

$$S = \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)$$

from above. Try using the inequality

$$a\, f\left(\frac{n}{b}\right) \leq c\, f(n)$$

to relate each term of S to $f(n)$.

Table of Contents

Basics revisited: how do we add two integers?

```
 C C C C C     carry
   X X X X X    first integer
+  X X X X X    second integer
  -----------
 X X X X X X    result
```

- Adding 3 bits can be done in constant time.
- It follows that the whole algorithm runs in linear time i.e., $O(n)$ many steps.

Basics revisited: how do we add two integers?

Question

Can we add two n-bit numbers in faster than in linear time?

Answer

No! There is no asymptotically faster algorithm because we have to read every bit of the input, which takes $O(n)$ time.

Basics revisited: how do we multiply two integers?

```
        X X X X  <- first input integer
      * X X X X  <- second input integer
        -------
        X X X X  \
      X X X X     \ O(n^2) intermediate operations:
    X X X X       / O(n^2) elementary  multiplications
  X X X X        /    + O(n^2) elementary additions
---------------
X X X X X X X X  <-  result of length 2n
```

- We assume that two `X`'s can be multiplied in $O(1)$ time (each `X` could be a bit or a digit in some other base).
- Thus the above procedure runs in time $O(n^2)$.

Basics revisited: how do we multiply two integers?

Question

Can we multiply two n-bit numbers in linear time, like addition?

Answer

No one knows! "Simple" problems can actually turn out to be difficult!

Question

Can we do it in faster than quadratic time? Let's try divide and conquer.

Table of Contents

Multiplying large integers by divide-and-conquer

- Split the two input numbers A and B into halves:
  - A_0, B_0 - the least significant $n/2$ bits;
  - A_1, B_1 - the most significant $n/2$ bits.

$$A = A_1 2^{\frac{n}{2}} + A_0 \qquad \underbrace{XX\ldots X}_{\frac{n}{2}}\,\underbrace{XX\ldots X}_{\frac{n}{2}}$$

$$B = B_1 2^{\frac{n}{2}} + B_0$$

- AB can now be calculated recursively using the following equation:

$$AB = A_1 B_1 2^n + (A_1 B_0 + B_1 A_0) 2^{\frac{n}{2}} + A_0 B_0.$$

Multiplying large integers by divide-and-conquer

```
function MULT(A, B)
    if |A| = |B| = 1 then return AB
    else
        A_1 ← MoreSignificantPart(A);
        A_0 ← LessSignificantPart(A);
        B_1 ← MoreSignificantPart(B);
        B_0 ← LessSignificantPart(B);
        X ← MULT(A_0, B_0);
        Y ← MULT(A_0, B_1);
        Z ← MULT(A_1, B_0);
        W ← MULT(A_1, B_1);
        return W 2^n + (Y + Z) 2^{n/2} + X
    end if
end function
```

Multiplying large integers by divide-and-conquer

How many steps does this algorithm take?

Each multiplication of two n digit numbers is replaced by four multiplications of $n/2$ digit numbers: $A_1 B_1$, $A_1 B_0$, $B_1 A_0$, $A_0 B_0$, plus we have a **linear** overhead to shift and add:

$$T(n) = 4\,T\left(\frac{n}{2}\right) + c\,n.$$

Let's use the Master Theorem!

Multiplying large integers by divide-and-conquer

$$T(n) = 4\,T\left(\frac{n}{2}\right) + c\,n$$

The critical exponent is $c^* = \log_2 4 = 2$, so the critical polynomial is n^2.

Then $f(n) = c\,n = O(n^{2-0.1})$, so Case 1 applies.

We conclude that $T(n) = \Theta(n^{c^*}) = \Theta(n^2)$, i.e., we gained **nothing** with our divide-and-conquer!

Multiplying large integers by divide-and-conquer

Question

Is there a smarter multiplication algorithm taking less than $O(n^2)$ many steps?

Answer

Remarkably, there is!

Table of Contents

History

- In 1952, one of the most famous mathematicians of the 20^{th} century, Andrey Kolmogorov, conjectured that you cannot multiply in less than quadratic many elementary operations.

- In 1960, Anatoly Karatsuba, then a 23-year-old student, found an algorithm (later it was called "divide-and-conquer") that multiplies two n-digit numbers in $\Theta\left(n^{\log_2 3}\right) \approx \Theta(n^{1.58\ldots})$ elementary steps, thus disproving the conjecture!! Kolmogorov was shocked!

The Karatsuba trick

- Once again we split each of our two input numbers A and B into halves:

$$A = A_1 2^{\frac{n}{2}} + A_0 \qquad \underbrace{XX\dots X}_{\frac{n}{2}}\underbrace{XX\dots X}_{\frac{n}{2}}.$$

$$B = B_1 2^{\frac{n}{2}} + B_0$$

- Previously we saw that

$$AB = A_1 B_1 2^n + (A_1 B_0 + A_0 B_1) 2^{\frac{n}{2}} + A_0 B_0,$$

but rearranging the bracketed expression gives

$$AB = A_1 B_1 2^n + ((A_1 + A_0)(B_1 + B_0) - A_1 B_1 - A_0 B_0) 2^{\frac{n}{2}} + A_0 B_0,$$

saving one multiplication at each round of the recursion!

The Karatsuba trick

```
function MULT(A, B)
    if |A| = |B| = 1 then return AB
    else
        A_1 ← MoreSignificantPart(A);
        A_0 ← LessSignificantPart(A);
        B_1 ← MoreSignificantPart(B);
        B_0 ← LessSignificantPart(B);
        U ← A_1 + A_0;
        V ← B_1 + B_0;
        X ← MULT(A_0, B_0);
        W ← MULT(A_1, B_1);
        Y ← MULT(U, V);
        return W 2^n + (Y − X − W) 2^{n/2} + X
    end if
end function
```

The Karatsuba trick

- How fast is this algorithm?
- Addition takes linear time, so we are only concerned with the number of multiplications.
- We need $A_1 B_1$, $A_0 B_0$ and $(A_1 + A_0)(B_1 + B_0)$; thus

$$T(n) = 3\,T\left(\frac{n}{2}\right) + c\,n.$$

The Karatsuba trick

Clearly, the run time $T(n)$ satisfies the recurrence

$$T(n) = 3\left[T\left(\frac{n}{2}\right)\right] + c\,n \tag{6}$$

Now the critical exponent is $c^* = \log_2 3$. Once again, we are in Case 1 of the Master Theorem, but this time

$$\begin{aligned} T(n) &= \Theta\left(n^{\log_2 3}\right) \\ &= \Theta\left(n^{1.58\ldots}\right) \\ &= o(n^2), \end{aligned}$$

disproving Kolmogorov's conjecture.

Next time: can we do even better?

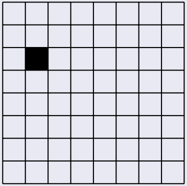
Table of Contents

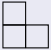
PUZZLE!

Problem

You are given a $2^n \times 2^n$ board with one of its cells missing (i.e., the board has a hole). The position of the missing cell can be arbitrary.

You are also given a supply of "trominoes", each of which can cover three cells as below.





PUZZLE!

Problem (continued)

Your task is to design an algorithm which covers the entire board (except for the hole) with these “trominoes”.

Hint

Do a divide-and-conquer recursion!

That's All, Folks!!