Recurrences and Master Theorem

Lecture 4

Akshar Varma

10th July, 2023

CS3000 Algorithms and Data

1. Recurrences

Recurrences

- Sequences that refer to themselves (self-recur) $\implies$ self-**recur**+sequ**ences**: Recurrences.
- As an example, let $T(n)$ be the running time for binary search.
- Because of the recursive nature, we can write: $T(n) = T(n/2) + O(1); T(1) = O(1)$.

  Discard half, recurse on remaining half. All this needs $O(1)$ extra time.
- Closed form? Try $\lg n$, since discarding a fraction tends to lead to that.
- This is the guess and prove (via induction) approach.
- Another recurrence: $T'(n) = T'(n-1) + O(1); T(1) = O(1)$.
  Runtime of Drone dropping with 1 drone.
- Unravel and solve or guess and prove: $T'(n) = \Theta(n)$

2. Detour: Karatsuba Multiplication

Multiplication of n digits integers

- We saw that grade school multiplication took $\Theta(n^2)$ time.
- In 1960, Kolmogorov organized a seminar and conjectured that: multiplication requires $\Omega(n^2)$ elementary operations.
- Let x, y be n digit numbers in base B, for any $0 < m < n$, we have:

$$x = x_1 B^m + x_0$$
$$y = y_1 B^m + y_0$$
$$x_0, y_0 < B^m$$

Euclidean division, anyone?

- $xy = (x_1 B^m + x_0)(y_1 B^m + y_0) = z_2 B^{2m} + z_1 B^m + z_0$
- Here: $z_2 = x_1 y_1$, $z_1 = x_0 y_1 + x_1 y_0$ and $z_0 = x_0 y_0$.
- This is *four* multiplications of smaller numbers $x_i y_j$ for $i, j \in \{0, 1\}$.
- Within a week, Karatsuba (then a 23 year old student) found a $O(n^{\lg 3})$ algorithm.
- He managed to multiply using *three* multiplications of smaller numbers.
- This uses the divide-and-conquer paradigm that we will see more in upcoming lectures.

Karatsuba's Algorithm

- Consider $x = x_1 B^m + x_0$ and $y = y_1 B^m + y_0$ as before.
- We had: $xy = (x_1 B^m + x_0)(y_1 B^m + y_0) = z_2 B^{2m} + z_1 B^m + z_0$
- Where: $z_2 = x_1 \cdot y_1$, $z_1 = x_0 \cdot y_1 + x_1 \cdot y_0$ and $z_0 = x_0 \cdot y_0$.
- Karatsuba observed the following:

$$\begin{aligned} z_1 &= x_1 y_0 + x_0 y_1 \\ &= x_1 y_0 + x_0 y_1 + x_1 y_1 - x_1 y_1 + x_0 y_0 - x_0 y_0 \\ &= x_1 y_0 + x_0 y_0 + x_0 y_1 + x_1 y_1 - x_1 y_1 - x_0 y_0 \\ &= (x_1 + x_0) y_0 + (x_0 + x_1) y_1 - x_1 y_1 - x_0 y_0 \\ &= (x_1 + x_0)(y_0 + y_1) - x_1 y_1 - x_0 y_0 \\ &= (\mathbf{x_1} + \mathbf{x_0}) \cdot (\mathbf{y_1} + \mathbf{y_0}) - \mathbf{z_2} - \mathbf{z_0} \end{aligned}$$

- It requires more additions, but only three total multiplications are needed now.
- Next, note that this splitting into two parts can be done again and again.
- This gives rise to a time complexity that is represented using a *recurrence*.

Karatsuba Algorithm’s Time Complexity Recurrence

- We wish to find the time taken to multiple two n digit numbers.
- This is the same as the number of elementary operations required additions, subtractions, digit shifts (multiple by powers of B). [These are all linear in n.]
- Let $T(n)$ be the time it takes to multiply two n digit numbers.
- Let us set m to be $n/2$ so that the split is into equal sized smaller numbers.
- We have: $T(n) = 3T(n/2) + cn + d = 3T(n/2) + O(n); T(1) = O(1)$.
- The $3T(n/2)$ part is because we perform 3 multiplications of $n/2$ digits.
- The rest $O(n)$ is because of a constant number of elementary operations.
- Today: Understand how to solve such recurrences to get a closed-form solution.

3. Guess and prove: Induction

Guess and Prove

- We saw two simple examples where we guessed a solution and proved it.
- Since I told you the time complexity is $O(n^{\lg 3})$, we can try to "guess and prove".
- However, guessing is not always an easy task unless you have lots of experience.
- Still, proving via induction is a useful skill to learn.
- Practice for induction:
  Let $\phi = \frac{1+\sqrt{5}}{2}$
  and $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$.
  Note: $1 + \phi = \phi^2$ and $1 - \phi = -\frac{1}{\phi}$.
- Show that: $F_n = \frac{\phi^n - (-\phi)^{-n}}{\sqrt{5}}$

4. Recursion Trees

Recursion Trees

- Visualize the recurrence using a recursion tree: recursive call $\longrightarrow$ child node.
- Karatsuba: $T(n) = 3T(n/2) + O(n); T(1) = O(1)$
- Strassen (recursive matrix multiplication): $T(n) = 7T(n/2) + O(n^2); T(1) = O(1)$
- A common recurrence: $T(n) = 2T(n/2) + O(1); T(1) = O(1)$
- Another common recurrence: $T(n) = 2T(n/2) + O(n); T(1) = O(1)$

5. Master Theorem

"Three Rings for the Elven-kings under the sky,
Seven for the Dwarf-lords in their halls of stone,
Nine for Mortal Men, doomed to die,
One for the Dark Lord on his dark throne
In the Land of Mordor where the Shadows lie.
One Ring to rule them all, One Ring to find them,
One Ring to bring them all and in the darkness bind them.
In the Land of Mordor where the Shadows lie."

– J. R. R. Tolkien

One Theorem to Rule Them All

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + n^c \log^k n \qquad \xrightarrow{\text{set}} \qquad c_{crit} = \log_b a$$

The Cases of the Master Theorem

Recurse vs. Splitting	c vs. c_{crit}	Master Theorem
$a \cdot T\left(\frac{n}{b}\right) \gg n^c$	$c_{crit} > c$	$T(n) = \Theta(n^{c_{crit}})$
$a \cdot T\left(\frac{n}{b}\right) \approx n^c$	$c_{crit} = c, k > -1$	$T(n) = \Theta(n^{c_{crit}} \log^{k+1} n)$
$a \cdot T\left(\frac{n}{b}\right) \approx n^c$	$c_{crit} = c, k = -1$	$T(n) = \Theta(n^{c_{crit}} \log \log n)$
$a \cdot T\left(\frac{n}{b}\right) \approx n^c$	$c_{crit} = c, k < -1$	$T(n) = \Theta(n^{c_{crit}})$
$a \cdot T\left(\frac{n}{b}\right) \ll n^c$	$c_{crit} < c$	$T(n) = \Theta(n^c)$