Lecture 2

- You have reviewed merge sort from the video
- This lecture, we will cover:
  - How to prove correctness recursive algorithms
  - How to analyze the run time recursive algorithms

How to prove correctness of recursive algorithm?

- Mergesort solves the sorting problem recursively
- How do we prove its correct?

Correctness of Merge sort: proof by induction

- For array size = 1, it is already sorted, Merge_sort correctly outputs sorted array
- Inductive assumption: Assume that merge_sort correctly sorts arrays of size $1, \ldots, k$
- Inductive step: For array A size $k + 1$
  - For any $k \geq 1$, we have $\frac{\lceil k+1 \rceil}{2} \leq k$
  - Inductive assumption implies that the two halfs will be sorted correctly by merge sort, since we know that the merge procedure will maintain the correct order, we see that merge_sort correctly sort A of size $k + 1$

Proof by induction

- Theorem: $p(n)$ is true for every positive integer n
- Template for proof:
  - Base case: **Prove** the most basic case(s)
  - Inductive hypothesis: **Assume** statement is true for some k, or for all numbers $\leq k$
  - Inductive step: **Prove** the statement true for $k+1$

Example: Making Postage

- Statement: Any postage $\geq$ 20 cents can be made with 4 and 5 cents stamps.
- **Base case**: 20 cents can be made with 4 fives
- **Inductive hypothesis**: assume we can make postage 20,..,k
- **Inductive step**: show that we can make postage k+1

What is missing?

Complete proof

- Base cases:
  - n=20: 4 x5; n=21: 4x4+5; n=22: 3x4+2x5; n=23: 2x4+3x5
- Inductive assumption: assume that any postage amount $20, \ldots, k$ can be made with 4, 5 cents stamps
- Inductive step: For k+1, for $k \geq 24$, we have $20 \leq k - 4 \leq k$, so we can make $k - 4$ and then use one more 4c postage to make $k + 1$

QED

Induction can be viewed as a form of Recursion

- Postage(n)

  if n = 20 return “four 5c stamps”

  else if n= 21 return …

  else if n = 22 return …

  else if n= 23 return …

  else return Postage(n-4)

What is run time $T(n)$ for making postage n?

Recurrence relation

$$T(n) = T(n-4) + c$$

$T(n-4) = T(n-8) + c$

$T(n) = T(n-8) + 2c$

$T(n) = T(n-4k) + kc$

How many layers of recursion ?
When do we hit the base case? $n - 4k \approx 23$

$$k_{max} \approx \frac{n-23}{4} = O(n)$$
$$T(n) = O(n)$$

Recurrence relation for merge_sort?

$T(n)$: the runtime for an input array of size n

Runtime:

1. Break A into two half sized arrays - c
2. Sort the two half size arrays – $2T\left(\frac{n}{2}\right)$
3. Merge the two sorted half arrays – cn

Recurrence relation:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

Solving recurrence relation using recursion tree

What if we break the array into 3 parts?

- Recurrence relation?
- Recursion tree?

More generally…

$T(n) = aT(\frac{n}{b}) + cn^d$ for some constants a, b, c, d

Recursion tree:

$$T(n) = \left(1 + \left(\frac{a}{b^d}\right)^2 + \cdots + \left(\frac{a}{b^d}\right)^k\right) cn^d$$

- If $\frac{a}{b^d} < 1, T(n) = O\left(n^d\right)$
- If $\frac{a}{b^d} = 1, T(n) = O\left(n^d \log n\right)$
- If $\frac{a}{b^d} > 1, T(n) = O\left(\left(\frac{a}{b^d}\right)^k cn^d\right) = O(n^{\log_b a})$

$$\left(\frac{a}{b^d}\right)^{\log_b n} = n^{\log_b \frac{a}{b^d}} = n^{\log_b a - \log_b b^d} = n^{\log_b a - \mathrm{d}}$$

Master Theorem

$$T(n) = aT(\frac{n}{b}) + cn^d$$

Case 1: $a < \mathrm{b}^{\mathrm{d}}$, or equiv. $d > \log_b a$, $T(n) = O\left(n^d\right)$

Case 2: $a = b^d$, or equiv. $d = \log_b a$, $T(n) = O\left(n^d \log n\right)$

Case 3: $a > b^d$, or equiv. $d < \log_b a$, $T(n) = O\left(n^{\log_b a}\right)$