

# ECS 122A: Algorithm Design and Analysis

## Week 1 Discussion

Ji Wang

Spring 2020

# Outline

- ▶ Mathematical Induction
- ▶ Insertion Sort and Time Analysis
- ▶ Merge Sort and Recurrence Relation
- ▶ Asymptotic Notation

# Mathematical Induction: Approach

**When** applicable: Prove that a property  $P(n)$  holds for every natural number  $n$ .

**How** it works:

1. Show that a property  $P(n)$  holds for the basis case, usually when  $n = 0$  or  $1$ .
2. Assume that  $P(n)$  is true for  $n = k$  where  $k$  is greater than the basis case.
3. Prove that  $P(n)$  is true for  $n = k + 1$ . In this step we will use the assumption above.
4. Then, by the principle of induction, we can conclude that  $P(n)$  is true for every natural number that is greater than or equal to the basis case.

# Mathematical Induction: Example

**Problem Statement:** Prove the following statement  $P(n)$  is true

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

**Build a skeleton:**

- ▶ Basis Step:
- ▶ Inductive Hypothesis:
- ▶ Inductive Step:
- ▶ Thus, by the principle of induction, we can conclude that the statement above is true for every natural number  $n$ .

# Mathematical Induction: Example

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

► Basis Step:

Verify  $P(n)$  is true when  $n = 1$ .  $1^3 = \left(\frac{1 \cdot (1+1)}{2}\right)^2$ .

# Mathematical Induction: Example

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

► Basis Step:

Verify  $P(n)$  is true when  $n = 1$ .  $1^3 = \left(\frac{1 \cdot (1+1)}{2}\right)^2$ .

► Inductive Hypothesis:

Assume  $P(n)$  is true when  $n = k$ . Then, we have

$$1^3 + 2^3 + \cdots + k^3 = \left(\frac{k(k+1)}{2}\right)^2.$$

## Mathematical Induction: Example

$$1^3 + 2^3 + \cdots + n^3 = \left(\frac{n(n+1)}{2}\right)^2$$

- Inductive Step: We need to show  $P(n)$  is true when  $n = k + 1$ .

$$\begin{aligned} 1^3 + 2^3 + \cdots + (k+1)^3 &= 1^3 + 2^3 + \cdots + k^3 + (k+1)^3 \\ &= \left(\frac{k(k+1)}{2}\right)^2 + (k+1)^3 \\ &= \frac{k^2(k+1)^2}{4} + \frac{4(k+1)^3}{4} \\ &= \frac{(k+1)^2(k^2 + 4(k+1))}{4} \\ &= \frac{(k+1)^2(k+2)^2}{4} \\ &= \left(\frac{(k+1)((k+1)+1)}{2}\right)^2 \end{aligned}$$

# Insertion Sort: Example

**Problem Statement:** Given a list of numbers, sort them in non-decreasing order.

**Algorithm:** Partition the list into sorted part and unsorted part. Insert the head of unsorted one into the sorted one. Repeat until no unsorted part remaining.

**Example:** 8 3 2 7 4 1 6

8	3	2	7	4	1	6	
8	3		2	7	4	1	6
3	8		2	7	4	1	6
3	8	2		7	4	1	6
2	3	8		7	4	1	6

2	3	8	7		4	1	6
2	3	7	8		4	1	6
2	3	7	8	4		1	6
2	3	4	7	8		1	6
2	3	4	7	8	1		6

1	2	3	4	7	8		6
1	2	3	4	7	8	6	
1	2	3	4	6	7	8	



# Insertion Sort: Pseudocode and Time Analysis

INSERTION-SORT( $A$ )

```
1  for  $j = 2$  to  $A.length$ 
2       $key = A[j]$ 
3      // Insert key into the sorted list  $A[1 \cdots j - 1]$ .
4       $i = j - 1$ 
5      while  $i > 0$  and  $A[i] > key$ 
6           $A[i + 1] = A[i]$ 
7           $i = i - 1$ 
8       $A[i + 1] = key$ 
```

See Section 2.1 on textbook for pseudocode convention.

- ▶ Worst case scenario: reversed order, e.g. 5 4 3 2 1.
- ▶ Running time in this case:  $\sum_{j=2}^n \sum_{i=1}^{j-1} = O(n^2)$
- ▶ Best case scenario: already sorted
- ▶ Running time in this case:  $\sum_{j=2}^n = O(n)$

# Merge Sort: Divide-and-Conquer Approach

1. **Divide** the problem into a number of subproblems that are smaller instances of the **same** problem.
2. **Conquer** by solving the subproblems **recursively**.
3. **Combine** the solutions to the subproblems to produce the solution to the original problem.

MERGE-SORT( $A, p, r$ )

```
1  if  $p < r$                                 // Input list:  $A[p \cdots r]$ 
2       $q = \lfloor (p + r)/2 \rfloor$                 // Divide
3      MERGE-SORT( $A, p, q$ )                    // Conquer
4      MERGE-SORT( $A, q + 1, r$ )                // Conquer
5      MERGE( $A, p, q, r$ )                      // Combine
```

## Merge Sort: How to combine

$L_1 \ L_2 \ L_3 \ L_4 \ \infty$

$R_1 \ R_2 \ R_3 \ R_4 \ \infty$

$A_1 \ A_2 \ A_3 \ A_4 \ A_5 \ A_6 \ A_7 \ A_8$

Comparison part of Merge function:

```
1   $i = 1$ 
2   $j = 1$ 
3  for  $k = 1$  to  $A.length$ 
4      if  $L[i] \leq R[j]$ 
5           $A[k] = L[i]$ 
6           $i = i + 1$ 
7      else
8           $A[k] = R[j]$ 
9           $j = j + 1$ 
```

For full version, refer to the textbook.

# Merge Sort: Recurrence Relation

MERGE-SORT( $A, p, r$ )

1	<b>if</b> $p < r$	// Input list: $A[p \cdots r]$
2	$q = \lfloor (p + r) / 2 \rfloor$	// Divide
3	MERGE-SORT( $A, p, q$ )	// Conquer
4	MERGE-SORT( $A, q + 1, r$ )	// Conquer
5	MERGE( $A, p, q, r$ )	// Combine

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

- ▶ Where does  $n$  come from? # of comparisons in Merge function.
- ▶ Why  $n$ ? As long as a comparison is done, we assign a value to an element in the merged array.

# Asymptotic Notation

By applying Limit Lemma Theorem, we can determine the growth relation between functions.

Frequently used functions	Order of Growth	Example
constant	$O(1)$	$f(n) = 10$
logarithm	$O(\log n)$	$f(n) = 4 \log(n + 2)$
linear	$O(n)$	$f(n) = 2n + 1$
linearithmic	$O(n \log n)$	$f(n) = 3n \log 4n + 1$
quadratic	$O(n^2)$	$f(n) = 2n^2 + 3n + 1$
cubic	$O(n^3)$	$f(n) = n^3 + 1$
exponential	$O(a^n), a \geq 2$	$f(n) = 3^n$

More details on Auxiliary Notes.