MODULE 1 / UNIT 2

INSERTION SORT

Today

- What is an algorithm?
- Understanding time complexities
- Sorting algorithms
- Insertion sort

A **sorting** problem

What would be your algorithm?

Algorithm

- **Webster's defintion:**
  - A **procedure** for **solving** a mathematical **problem** (as of finding the greatest common divisor) in a finite number of **steps** that frequently involves repetition of an operation;
  - *broadly* : a **step-by-step procedure for solving a problem** or accomplishing some end especially by a computer

- **An algorithm should be described..**
  - Unequivocally
  - Concisely
  - With a sufficient level of details for the target reader.
    - e.g. "Pick the smallest book" may or may not be enough:

Some high level ideas for sorting:

- Seek the smallest book ➜ place in the left ➜ Seek the next smallest (excluding the leftmost) ➜ place in the second left ➜ ....

- Keep the first k elements sorted ➜ Place the k+1-th element in the increasing order ➜ (repeat)

- Keep switching between adjacent elements whenever left is greater than right. ➜ Repeat the procedure n times

What if you have thousands of books?

Some high level ideas for sorting:

- Seek the smallest book ➜ place in the left ➜ Seek the next smallest (excluding the leftmost) ➜ place in the second left ➜ ....
  : Selection sort

- Keep the first k elements sorted ➜ Place the k+1-th element in the increasing order ➜ (repeat)
  : Insertion sort

- Keep switching between adjacent elements whenever left is greater than right. ➜ Repeat the procedure n times
  : Bubble sort

What if you have thousands of books?

Visual illustration of insertion sort

https://www.toptal.com/developers/sorting-algorithms/insertion-sort

Insertion sort

- **Key idea : Make sure that the first k elements being sorted, and increase k stepwise.**
- **Procedure**
  - When k = 1, the first 1 element is automatically sorted.
  - When first k elements are sorted,
    the (k+1)-th elements can be placed in one of the (k+1) spots.
    : Keep switching the element with the previous element as long as the previous element is larger.
  - Then the first (k+1) elements are sorted.
  - By induction, repeating the procedure n times will sort the entire list.

Computer-friendly description via pseudocodes

Data: An unsorted list $A[1 \cdots n]$
Result: The list $A[1 \cdots n]$ is sorted

```
for j = 2 to n do
    key = A[j];
    i = j − 1;
    while i > 0 and A[i] > key do
        A[i + 1] = A[i];
        i = i − 1;
    end
    A[i + 1] = key;
end
```

Jupyter notebook:

insertion_sort.ipynb

How long does an insertion sort take?

for $j = 2$ **to** n	$c_1 n$
do	
$\quad key = A[j]$;	$c_2(n-1)$
$\quad i = j - 1$;	$c_3(n-1)$
$\quad$ **while** $i > 0$ *and* $A[i] > key$	$c_4 \sum_{j=2}^{n} j$
$\quad$ **do**	
$\qquad A[i+1] = A[i]$;	$c_5 \sum_{j=2}^{n} (j-1)$
$\qquad i = i - 1$;	$c_6 \sum_{j=2}^{n} (j-1)$
$\quad$ **end**	
$\quad A[i+1] = key$;	$c_7(n-1)$
end	

How long does an insertion sort take?

$$T(n) = (c_4 + c_5 + c_6)n^2 + \frac{2(c_1 + c_2 + c_3 + c_7) + c_4 - c_5 - c_6}{2}n - (c_2 + c_3 + c_4 + c_7)$$

end

$A[i+1] = key$; $\quad c_7(n-1)$

end

How long does an insertion sort take?

$$
\begin{aligned}
T(n) &= (c_4 + c_5 + c_6)n^2 \\
&\quad + \frac{2(c_1 + c_2 + c_3 + c_7) + c_4 - c_5 - c_6}{2} n \\
&\quad - (c_2 + c_3 + c_4 + c_7) \\
&= \Theta(n^2)
\end{aligned}
$$

$A[i+1] = key$; $c_7(n-1)$

end

Time complexity : Notations

- **$O(\cdot)$: Time complexity is AT MOST $c(\cdot)$ as n increases**
  - Formally, $T(n) = O(f(n))$ if and only if there exists $n_0 > 0$ and $M > 0$ such that $T(n) \leqslant Mf(n)$ for all $n > n_0$

- **$\Omega(\cdot)$: Time complexity is AT LEAST $c(\cdot)$ as n increases.**
  - Formally, $T(n) = \Omega(f(n))$ if and only if there exists $n_0 > 0$ and $M > 0$ such that $T(n) \geqslant Mf(n)$ for all $n > n_0$.

- **$\Theta(\cdot)$: Time complexity is EXACTLY $c(\cdot)$ as n increase.**
  - $T(n) = \Theta(f(n))$ if and only if $T(n) = O(f(n))$ *and* $T(n) = \Omega(f(n))$

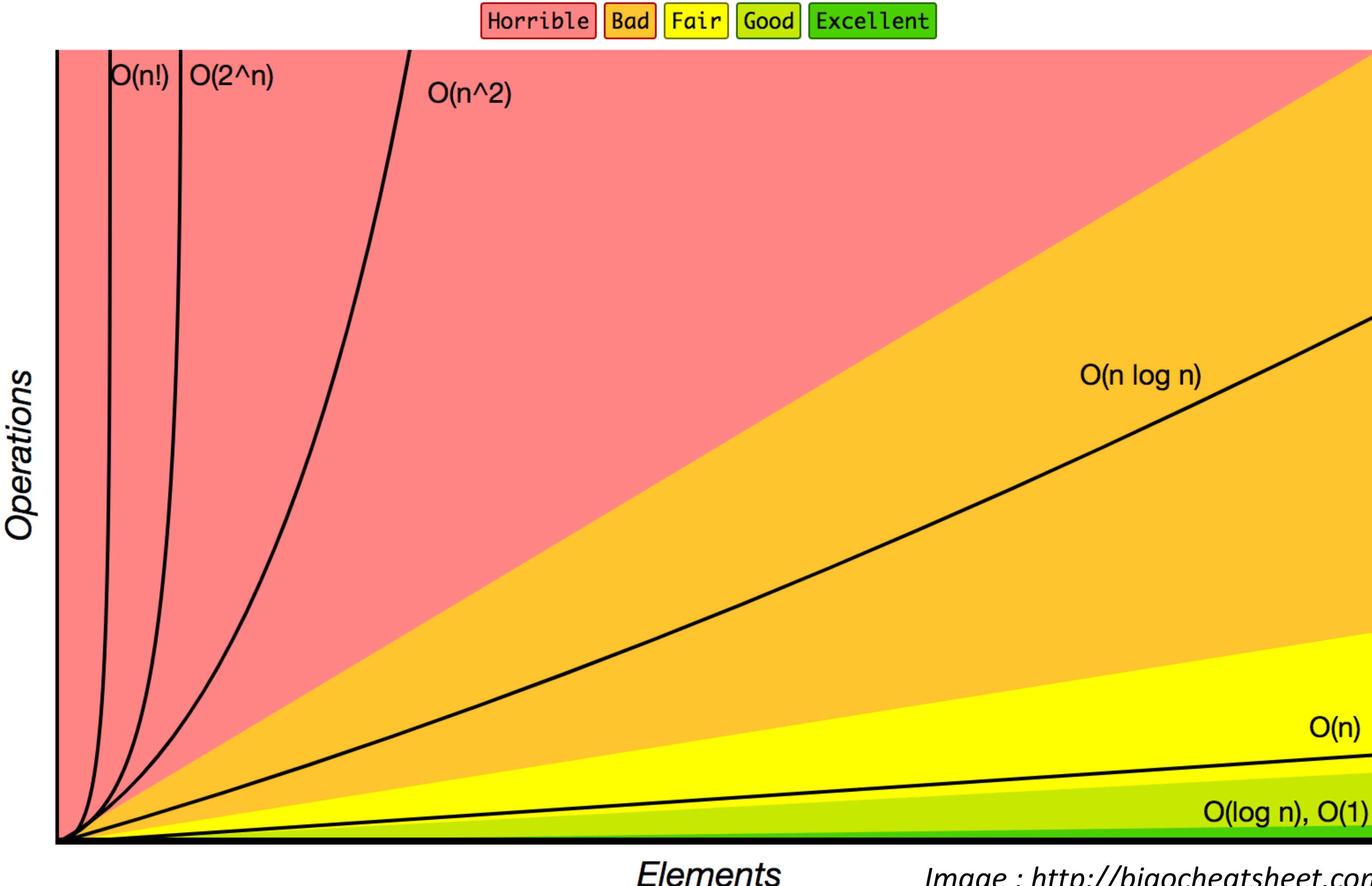

Image : http://bigocheatsheet.com

Q: What would be time complexity of...

- Finding an element from an array?
- Finding an element from a sorted array?
- Inserting an element to an array?
- Inserting an element to a sorted array?
- Finding median value from a sorted array?

Summary : Simple Sorting

- **Sorting algorithms**
  - Selection sort
  - Insertion sort
  - Bubble sort

- **Time complexity**
  - Insertion sort is $\Theta(n^2)$
  - Scalable algorithms should be at most $\Theta(n \log n)$

- **Would it be possible to sort faster than $\Theta(n^2)$?**

Reading List

- CLRS I.1 – I.3 (pp. 5-65)