RECAP

- Analysis of Algorithms.
    - Input size
    - Running time

Input Size: depends on the Problem being studied.

The best measure of the input size is "the total number of bits needed to represent the input in ordinary binary notation"

- For sorting – Array Size
- multiplying two integers – total # of bits needed to represent integers
- For graphs – Number of vertices & Number of edges

Running time

The running time of an algorithm on a particular input is the number of operations or "steps" executed.

(RAM model)

Warmup - Examples

A has n elements

Pseudo code

```
FIND-MAX(A)
    max = A[1]
    for j = 2 to A.length
        if max < A[j]
            max = A[j]
    return max
```

Q: What is the running time?

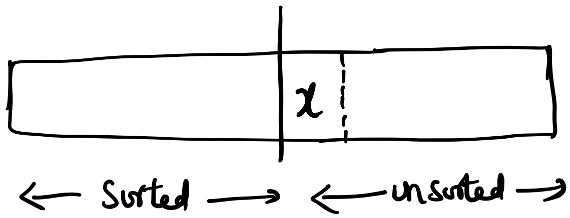
Sorting Problem [Recap]

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.

Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \le a'_2 \le \cdots \le a'_n$.

Insertion Sort [Reference : Coreman - Chapter 2]

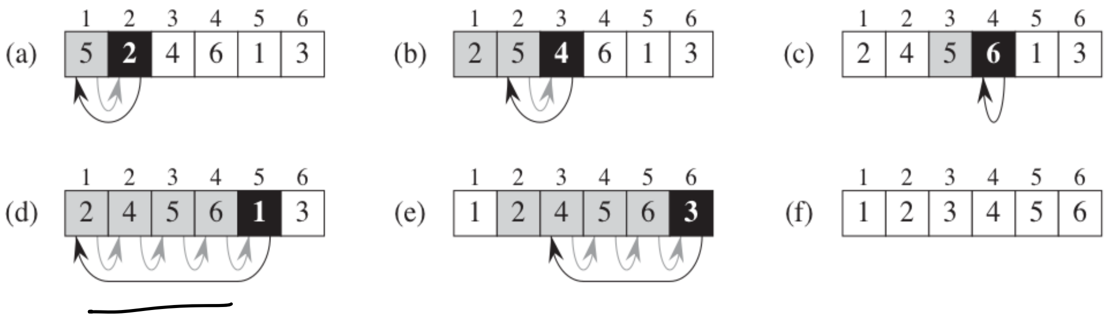
Given
Array of n elements.



Example

Input array:

5	2	4	6	1	3



Run insertion sort on

a

9	7	6	5	3	2

Next :

- Analysis of Insertion sort

Pseudocode

INSERTION-SORT(A)

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 . . j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

$T(n)$: the running time of INSERTION-SORT on an input of n values

INSERTION-SORT(A)	*cost*	*times*
for $j = 2$ **to** $A.length$	c_1	n
$key = A[j]$	c_2	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
$i = j-1$	c_4	$n-1$
while $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] = key$	c_8	$n-1$

$$
\begin{aligned}
T(n) \quad = \quad & c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) \\
& + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) \; .
\end{aligned}
$$

BEST CASE

The best case occurs if the array is already sorted.

In this case $t_j = 1$ for $j = 2, 3, \ldots n$.

$\therefore$ the best case running time is

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\
&= (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8) \,. \\
&= an + b
\end{aligned}
$$

WORST-CASE

- The worst case occurs if the array is in reverse sorted order (decreasing order)

- In this case $t_j = j$ for $j = 2, 3, \ldots n$

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) \\
&\quad + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) \,.
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5 \left(\frac{n(n+1)}{2} - 1 \right) \\
&\quad + c_6 \left(\frac{n(n-1)}{2} \right) + c_7 \left(\frac{n(n-1)}{2} \right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2} \right) n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8 \right) n \\
&\quad - (c_2 + c_4 + c_5 + c_8) \,.
\end{aligned}
$$

$$T(n) = an^2 + bn + c$$

For the remainder of this course, we shall usually concentrate on finding only the worst-case running time.

$\downarrow$

longest running for any input of size n.

[Don't disclose the answer]

Puzzle

Businessman

Son - 1

$W(t) = 10^{8} t + 400$

$W(t)$ ↓ wealth at time t (in years)

Son - 2

$W(t) = 10^{-6} t^{2} + 400$

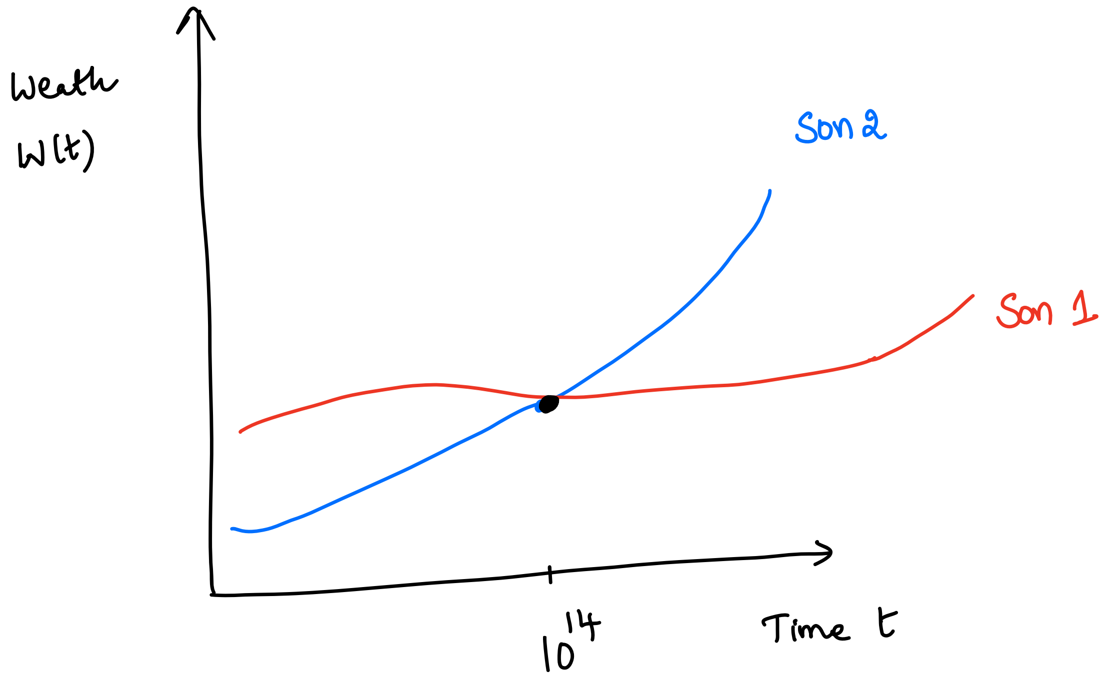
[Don't disclose the answer]

Puzzle

$$
\begin{tikzcd}
 & \text{Businessman} \arrow[dl] \arrow[dr] & \\
\text{Son-1} \arrow[d] & & \text{Son-2} \arrow[d] \\
W(t) = 10^{8}\, t + 400 & & W(t) = 10^{-6}\, t^{2} + 400
\end{tikzcd}
$$

$W(t)$ → wealth at time t (in years)

Q Who is doing better in their business Son-1 or Son-2?


Weath
W(t)
Son 2
Son 1
10^{14}
Time t

How Business/wealth is related to Algorithms?

Wealth (maximized)

Wealth as fun of time

Running time (minimized)

Running time as fun of input size.

Q:
$$W(t) = 10^{8} t + 400$$

$$W(t) = 10^{-6} t^{2} + 400$$

Does 400, 10^{8} or 10^{-6} play any role here?

$$W(t) = 10^{8} t + 400 = \Theta(t)$$

$$W(t) = 10^{-6} t^{2} + 400 = \Theta(t^{2})$$

Next lecture.

Order of growth (or) rate of growth

Order of growth of an algorithm means how the Computation time increase when we increase the input size.

We consider only the leading term as lower order terms are insignificant for large size inputs.

We ignore the leading term's constant coefficient.

$\boxed{\text{Imp}}$: Asymptotic efficiency of algoritms:

We are only interested in how the running time of an algorithm increases with the size of the input in the limit, as the size of the input increases without bound.

" usually an algorithm that is asymptotically more efficient will be the best choice for all but very small i/ps "

We usually consider one algorithm to be more efficient than another if its worst case running time has a lower order of growth.

* Due to constant factors and lower order terms, an algorithm whose running time has a higher order of growth might take less time for Small inputs than an algorithm whose running time has a lower order of growth.