Algorithms & Analysis

Kuan-Yu Chen (陳冠宇)

2019/02/20 @ TR-310-1, NTUST

Algorithm

- **Algorithm:** Any well-defined computation procedure that takes some value, or set of values, as input and produces some value, or set of values, as output
  - Tool for solving well specific computational problem

  - Sorting
    - Input: A sequence of n numbers $\{a_1, a_2, \ldots, a_n\}$
    - Output: A permutation of the input sequence such that $\{a'_1, a'_2, \ldots, a'_n\}$

- An algorithm is said to be ***correct*** if for every input, it halts with the correct output
  - A correct algorithm ***solves*** the given computational problem

Sorting

- Sorting means arranging the elements of an array so that they are placed in some relevant order which may be either **ascending** or **descending**

- A sorting algorithm is defined as an algorithm that puts the elements of a list in a certain order, which can be either **numerical** order, **lexicographical** order, or **any user-defined** order
  - Bubble, Insertion, Selection, Tree
  - Merge, Quick, Radix, Heap, Shell

Insertion Sort.

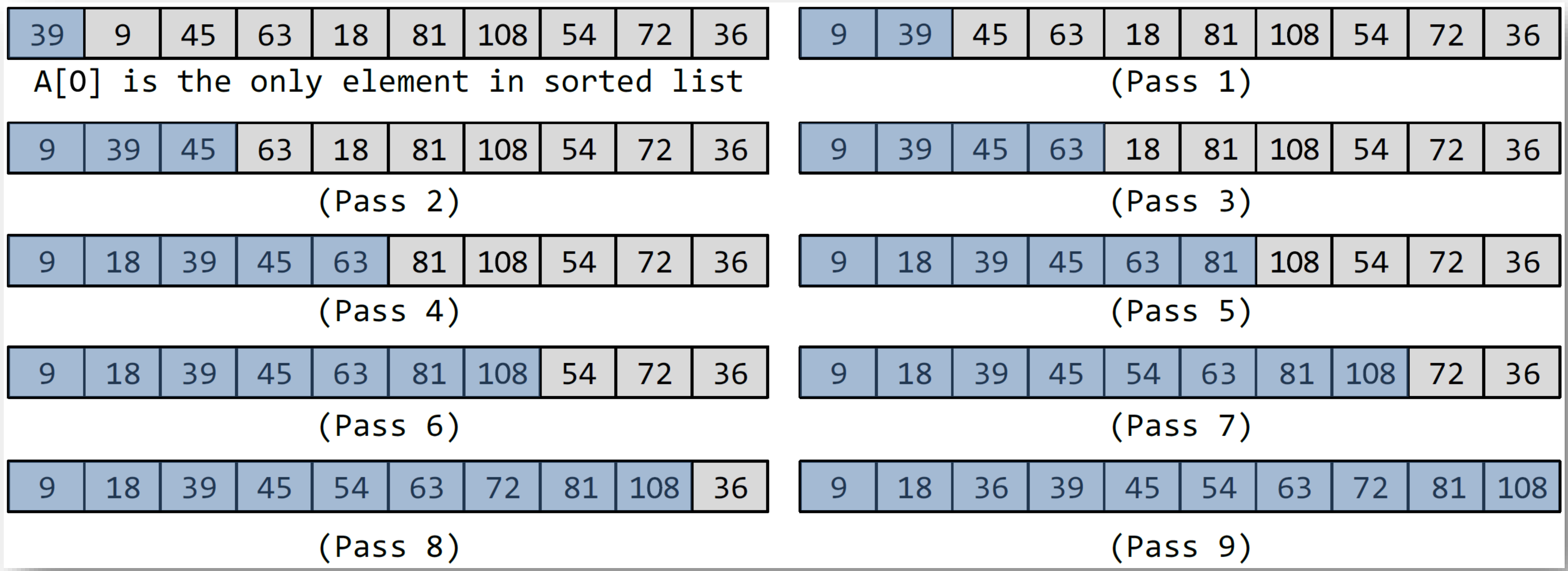
- Insertion sort is a very simple sorting algorithm in which the sorted array (or list) is built one element at a time

- The procedure!
  - The array of values to be sorted is divided into two sets
    - One stores sorted values
    - Another contains unsorted values
  - The sorting algorithm will proceed until there are no elements in the unsorted set

Example

- Please sort a given data array by using insertion sort

39	9	45	63	18	81	108	54	72	36



39	9	45	63	18	81	108	54	72	36

A[0] is the only element in sorted list

9	39	45	63	18	81	108	54	72	36

(Pass 1)

9	39	45	63	18	81	108	54	72	36

(Pass 2)

9	39	45	63	18	81	108	54	72	36

(Pass 3)

9	18	39	45	63	81	108	54	72	36

(Pass 4)

9	18	39	45	63	81	108	54	72	36

(Pass 5)

9	18	39	45	63	81	108	54	72	36

(Pass 6)

9	18	39	45	54	63	81	108	72	36

(Pass 7)

9	18	39	45	54	63	72	81	108	36

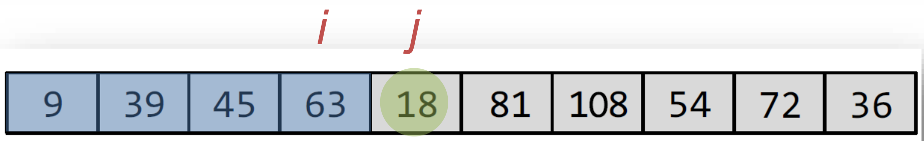
(Pass 8)

9	18	36	39	45	54	63	72	81	108

(Pass 9)

Insertion Sort..

```
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 $A[i] > key$
        $A[i+1] = A[i]$
        $i = i - 1$
    $A[i+1] = key$
```



(Pass 3)

Analyzing Algorithms

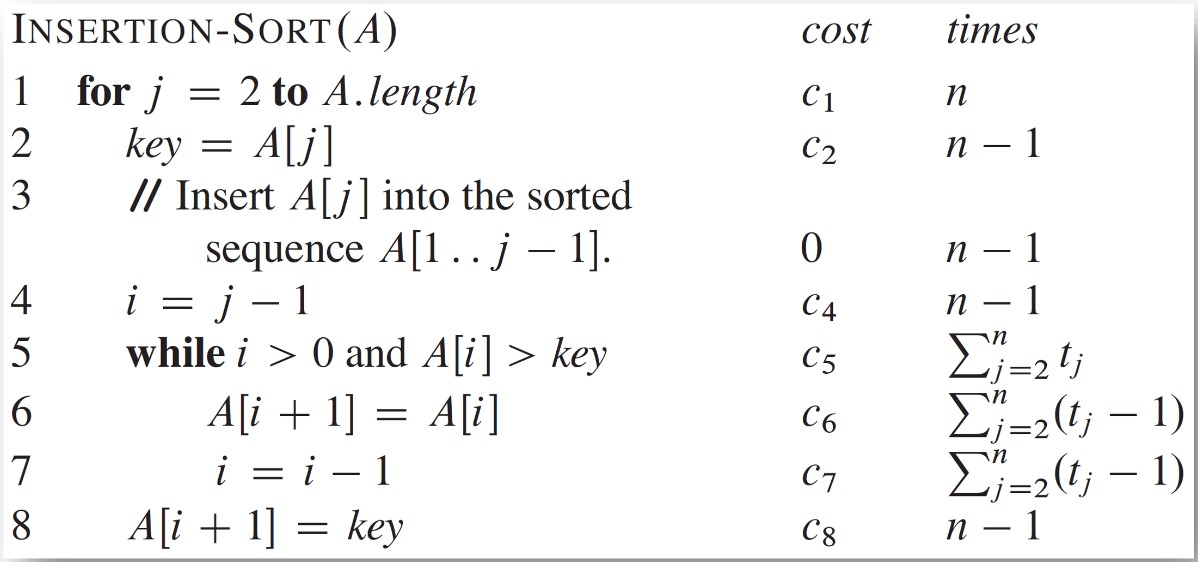
- Analyzing an algorithm has come to mean predicting the **resources** that the algorithm requires
  - Memory, communication bandwidth, or computer hardware are of primary concern
  - Most often we want to measure the computational time

- The **running time** of an algorithm on a particular input is the number of primitive operations or steps executed
  - It is convenient to define the notion of step so that it is as machine-independent as possible
    - One line may take a different amount of time than another line
    - We shall assume that each execution of the i^{th} line takes time c_i, where c_i is a constant

Running Time of Insertion Sort.

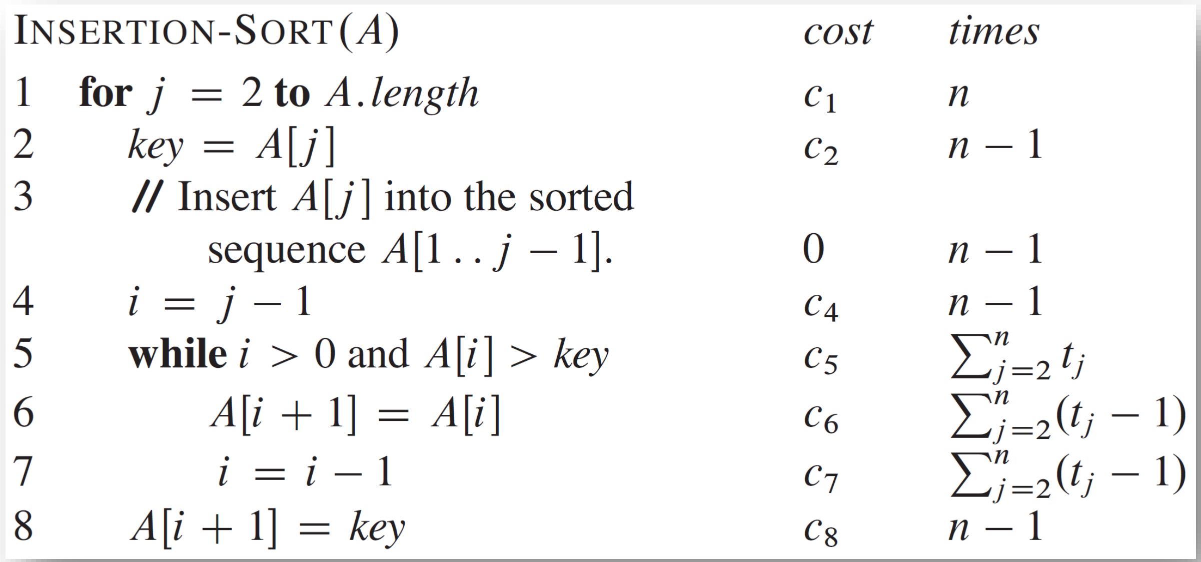
- The running time of the algorithm is the sum of running times for each statement executed
  - A statement that takes c_i to execute and executes n times will contribute $c_i \times n$ to the total running time

- To compute $T(n)$, the running time of insertion sort on an input of n values, we sum the products of the [illegible]and [illegible] [illegible] for each statement



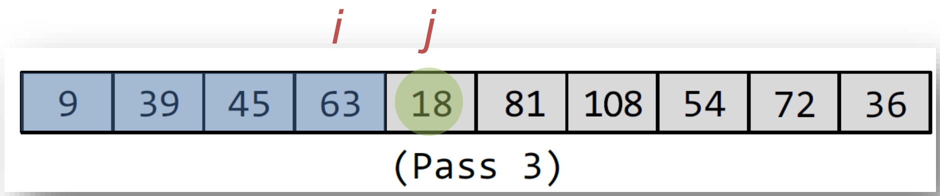
	INSERTION-SORT(A)	*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$

Running Time of Insertion Sort..

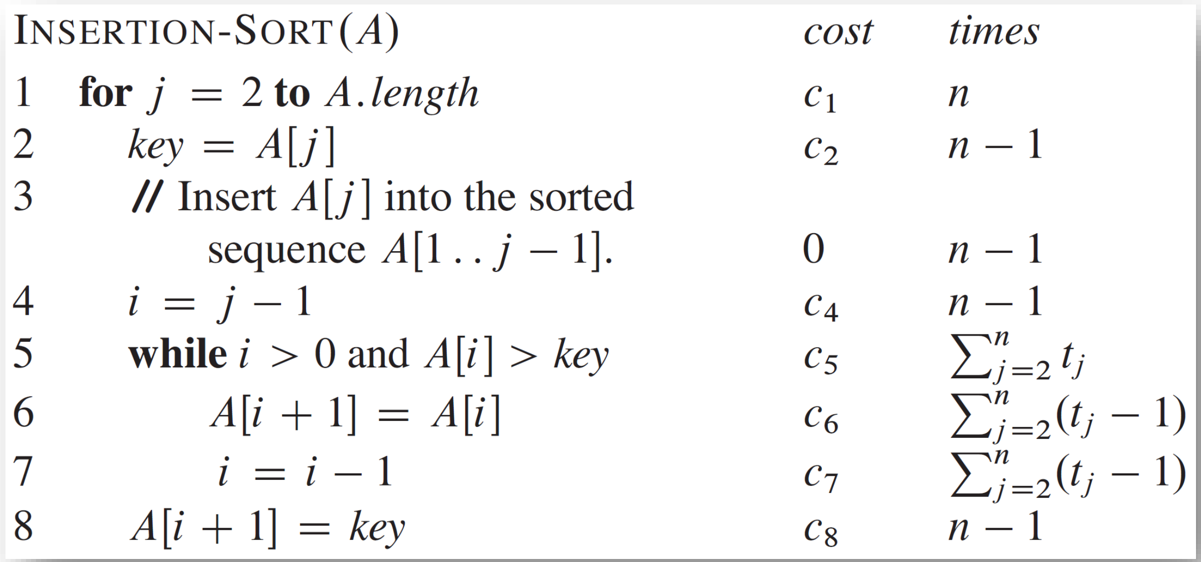


INSERTION-SORT(A)		cost	times
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1 \,.\,.\, j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$

- $n = A.length$
- t_j denote the number of times the **while** loop test in line 5 is executed for that value of j



Running Time of Insertion Sort…

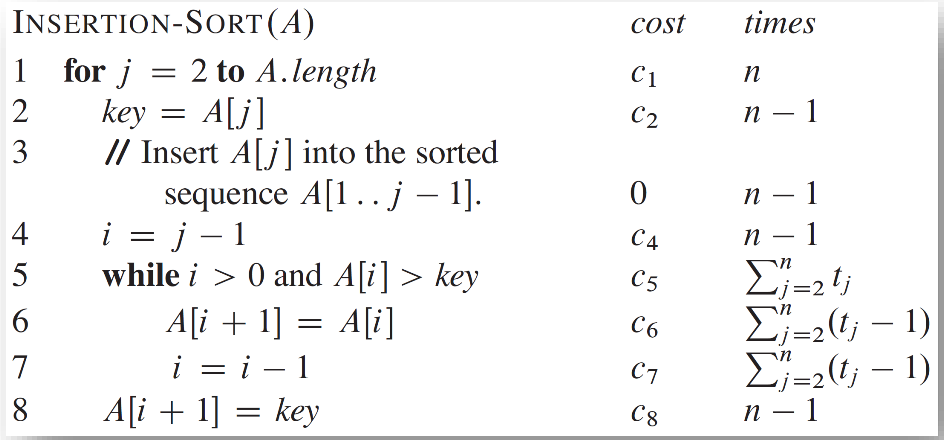


```
INSERTION-SORT(A)                                      cost     times
1  for $j = 2$ to $A.length$                           $c_1$    $n$
2      $key = A[j]$                                    $c_2$    $n-1$
3      // Insert $A[j]$ into the sorted
              sequence $A[1..j-1]$.             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

- Consequently, we can obtain:

$$T(n) = c_1 \times n + c_2 \times (n-1) + c_4 \times (n-1)$$
$$+ c_5 \times \sum_{j=2}^{n} t_j + c_6 \times \sum_{j=2}^{n} (t_j - 1)$$
$$+ c_7 \times \sum_{j=2}^{n} (t_j - 1) + c_8 \times (n-1)$$

Running Time of Insertion Sort….



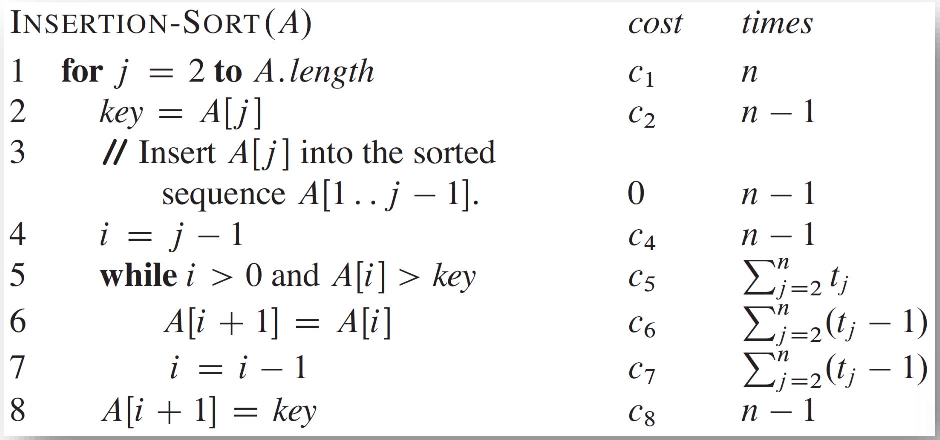
```
INSERTION-SORT(A)                                  cost   times
1  for j = 2 to A.length                           c_1    n
2      key = A[j]                                  c_2    n − 1
3      // Insert A[j] into the sorted
           sequence A[1 .. j − 1].                 0      n − 1
4      i = j − 1                                   c_4    n − 1
5      while i > 0 and A[i] > key                  c_5    Σ_{j=2}^{n} t_j
6          A[i + 1] = A[i]                         c_6    Σ_{j=2}^{n} (t_j − 1)
7          i = i − 1                               c_7    Σ_{j=2}^{n} (t_j − 1)
8      A[i + 1] = key                              c_8    n − 1
```

$$T(n) = c_1 \times n + c_2 \times (n-1) + c_4 \times (n-1) + c_5 \times \sum_{j=2}^{n} t_j + c_6 \times \sum_{j=2}^{n} (t_j - 1) + c_7 \times \sum_{j=2}^{n} (t_j - 1) + c_8 \times (n-1)$$

- The **best case** occurs if the array is already sorted
  - For each $j = 2, 3, \dots, n$, we then find that $A[i] \le key$ in line 5 when i has its initial value of $j - 1$
  - Thus, $t_j = 1$ for $j = 2, 3, \dots, n$
  - The running time is:

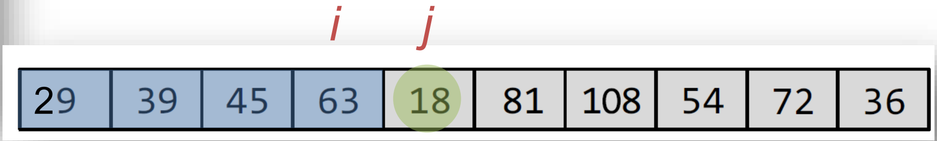
$$\begin{aligned} T(n) &= c_1 \times n + c_2 \times (n-1) + c_4 \times (n-1) + c_5 \times (n-1) + c_8 \times (n-1) \\ &= (c_1 + c_2 + c_4 + c_5 + c_8) \times n - (c_2 + c_4 + c_5 + c_8) \end{aligned}$$

Running Time of Insertion Sort.....



INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

$$T(n) = c_1 \times n + c_2 \times (n-1) + c_4 \times (n-1)$$
$$+ c_5 \times \sum_{j=2}^{n} t_j + c_6 \times \sum_{j=2}^{n} (t_j - 1)$$
$$+ c_7 \times \sum_{j=2}^{n} (t_j - 1) + c_8 \times (n-1)$$



- The **worst case** occurs if the array is in reverse sorted order
  - We must compare each element $A[j]$ with each element in the entire subarray $A[1, \dots, j-1]$, and so $t_j = j$ for $j = 2, 3, \dots, n$
  - The running time is:

$$\sum_{j=2}^{n} j = \frac{n \times (n+1)}{2} - 1$$
$$\sum_{j=2}^{n} (j-1) = \frac{n \times (n-1)}{2}$$

$$T(n) = c_1 \times n + c_2 \times (n-1) + c_4 \times (n-1) + c_5 \times \left(\frac{n \times (n+1)}{2} - 1\right)$$
$$+ c_6 \times \left(\frac{n \times (n-1)}{2}\right) + c_7 \times \left(\frac{n \times (n-1)}{2}\right) + c_8 \times (n-1)$$
$$= \left(\frac{c_5 + c_6 + c_7}{2}\right) \times n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5 - c_6 - c_7}{2} + c_8\right) \times n - (c_2 + c_4 + c_5 + c_8)$$

Running Time of Insertion Sort……

- To sum up,
  - For the best case

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

    - We can express this running time as $a \times n + b$ for constants a and b that depend on the statement costs c_i
    - It is a ***linear function*** of n

  - For the worst case

$$T(n) = \left(\frac{c_5 + c_6 + c_7}{2}\right) \times n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5 - c_6 - c_7}{2} + c_8\right) \times n - (c_2 + c_4 + c_5 + c_8)$$

    - We can express this running time as $a \times n^2 + b \times n + c$ for constants a, b and c that depend on the statement costs c_i
    - It is a ***quadratic function*** of n

Questions?

kychen@mail.ntust.edu.tw