Sorting problem

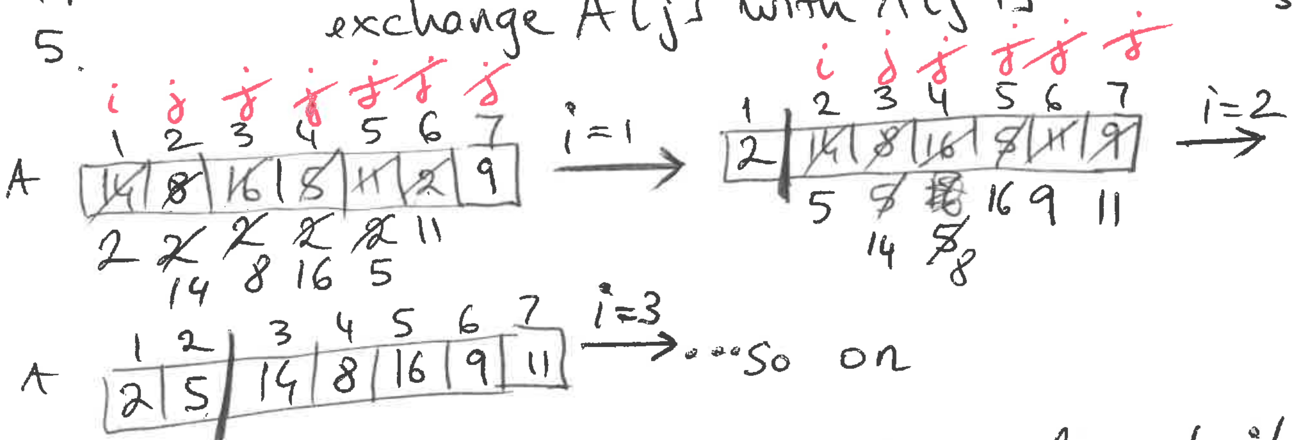
Input: a sequence of n numbers $A = \langle a_1, a_2, \ldots, a_n \rangle$

Output: rearrange the numbers in increasing order

$$A = \langle a_1', a_2', \ldots, a_n' \rangle \text{ s.t. } a_1' \le a_2' \le \ldots \le a_n'$$

Bubble-Sort (A, n)

Bubble-Sort (A, n)	cost	no. of times
1. **for** $i = 1$ to $n-1$	c_1	n
2. // move the smallest elm. in $A[i..n]$ to $A[i]$	0	$n-1$
3. **for** $j = n$ downto $i+1$	c_3	$\sum_{i=1}^{n-1}(n-i+1)$
4. **if** $A[j] < A[j-1]$	c_4	$\sum_{i=1}^{n-1}(n-i)$
5. exchange $A[j]$ with $A[j-1]$	c_5	$\sum_{i=1}^{n-1}(n-i)$



Loop Invariant (LI): At the start of each iteration i of the **for** loop (line 1), the subarray $A[1..i-1]$ contains the $(i-1)$ smallest elements in A, in sorted order.

Observation

For any loop (for, while loop) the instruction in the header of the loop executes one more time than the instructions in the body of the loop.

for $i = 2$ to 5	5 times
$x = x+2$	4 times
$y = y-1$	4 times

i takes values : $2, 3, 4, 5, \underline{6}$

RT analysis for Bubble-Sort

- use RAM model

$T(n)$ - running time for input size n

n = no. of elements to be sorted

$$\boxed{T(n) = \sum_{\text{all statements}} (\text{cost of statement})(\text{no. of times the statement is executed})}$$

j takes values $n, n-1, n-2, \ldots, i+1, \underline{i}$

$(n-i+1)$ times

$$T(n) = c_1 n + \cancel{0 \cdot (n-1)} + c_3 \cdot \sum_{i=1}^{n-1} (n-i+1) + (c_4 + c_5) \cdot \sum_{i=1}^{n-1} (n-i)$$

$$T(n) = c_1 n + c_3 \left(n + (n-1) + (n-2) + \ldots + 2\right) + (c_4 + c_5) \cdot$$
$$\left((n-1) + (n-2) + \ldots + 1\right)$$

Arithmetic series : $\boxed{1 + 2 + 3 + \ldots + n = \frac{n \cdot (n+1)}{2}}$

example : $1 + 2 + 3 + \ldots + 20 = \frac{20 \cdot 21}{2} = 210$

$$T(n) = c_1 n + c_3 \left(\frac{n \cdot (n+1)}{2} - 1\right) + (c_4 + c_5) \cdot \frac{(n-1) \cdot n}{2}$$

$$T(n) = \left(\frac{c_3}{2} + \frac{c_4}{2} + \frac{c_5}{2}\right) n^2 + \left(c_1 + \frac{c_3}{2} - \frac{c_4}{2} - \frac{c_5}{2}\right) n - c_3$$

$\boxed{T(n) = a n^2 + b n + c}$ - quadratic function

Order of growth

- drop the lower order terms
- ignore the constant in the leading term

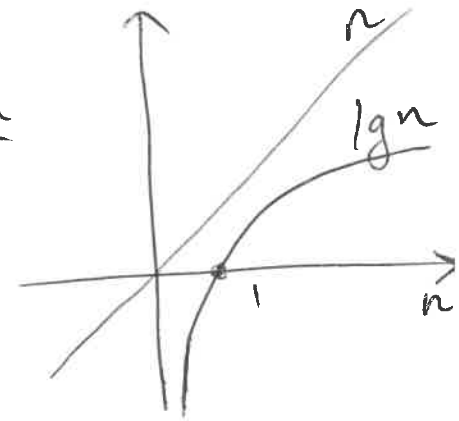
Bubble-Sort

$\Rightarrow a n^2$

$\Rightarrow n^2$

$\boxed{T(n) = \Theta(n^2)}$ — Bubble-sort has the order of growth n^2.

An alg. is more efficient than another algorithm if it has a smaller order of growth



Merge-Sort : $RT = \Theta(n \cdot \lg n)$ $\quad n \cdot \lg n$

Bubble-Sort : $RT = \Theta(n^2)$ $\quad n \cdot n$

Merge-Sort is more efficient than Bubble-Sort.

Examples

- Find the RT for the pseudocode below. Assume that n is the input size.

	Algorithm-Example (A, n)	no. of times
$\Theta(1)$	$\alpha = 2$	1
$\Theta(1)$	for $i = 2$ to 50	50
	$\quad \alpha = \alpha + 50$	49
$\Theta(n^3)$	for $i = 1$ to n	$n+1$
	$\quad$ for $j = 1$ to n^2	$n \cdot (n^2+1)$
	$\quad\quad A[j] = A[i] + \alpha + j$	$n \cdot n^2$
$\Theta(n)$	for $i = 1$ to n	$n+1$
	$\quad \alpha = \alpha + A[i]$	n

$\boxed{T(n) = \Theta(n^3)}$

$\Theta(1)$ – constant

$O(1)$ – constant

- Find Θ-notation for the number of times the statement "x=x+1" is executed, where n – input size.

```
for i=1 to n
  for j=1 to 1000
    for k=1 to n (log_2 n)^3
      x=x+1
```

Solution 1

$$n \cdot 1000 \cdot n \cdot (\log_2 n)^3 = \underline{1000 \cdot n^2 (\log_2 n)^3} \rightarrow$$

Notation

$$\lg n = \log_2 n$$

$$\lg^3 n = (\lg n)^3$$

$$\log_2^3 n = (\log_2 n)^3$$

$$\rightarrow 1000\, n^2 \lg^3 n \Rightarrow \boxed{\Theta(n^2 \lg^3 n)}$$

Solution 2

$$\sum_{i=1}^{n} \sum_{j=1}^{1000} \underbrace{\sum_{k=1}^{n \cdot \lg^3 n} 1} = \sum_{i=1}^{n} \underbrace{\sum_{j=1}^{1000} n \cdot \lg^3 n} = \sum_{i=1}^{n} 1000 \cdot n \lg^3 n =$$

$$= 1000 \sum_{i=1}^{n} n \lg^3 n = 1000 \cdot n \cdot n \lg^3 n = 1000\, n^2 \lg^3 n = \boxed{\Theta(n^2 \lg^3 n)}$$

- Same question for the pseudocode:

```
x=100
i=n
while i ≥ 1
  x=x+1
  i=i/2
```

Solution

let $n=16$. Then i takes values: $\underbrace{16, 8, 4, 2, 1}$, $\underline{0.5}$

"$x=x+1$" executes 5 times

General case:

i takes values: $\underbrace{\frac{n}{2^0}, \frac{n}{2}, \frac{n}{2^2}, \frac{n}{2^3}, \cdots, \frac{n}{2^k} \geq 1}_{(k+1) \text{ values}}$, $\underline{\frac{n}{2^{k+1}} < 1}$

"$x=x+1$" executes $(k+1)$ times

$$2^k \leq n < 2^{k+1}$$

$$\lg 2^k \leq \lg n < \lg 2^{k+1}$$

$$k \leq \lg n < k+1$$

$$k = \lfloor \lg n \rfloor$$

"$x=x+1$" executes $(\lfloor \lg n \rfloor + 1)$ times

$$\lfloor \lg n \rfloor = \Theta(\lg n)$$

"$x=x+1$" executes $\boxed{\Theta(\lg n)}$

- Find Θ-notation for the number of times the statement "$x=x+1$" is executed, when n is the input size:

```
i = n
while i >= 1
    for j = 1 to n
        x = x + 1
    i = i/3
```

Solution

while $\Rightarrow \Theta(\log_3 n)$

for $\Rightarrow \Theta(n)$

$\boxed{\text{total RT} = \Theta(n \cdot \log_3 n)}$

- Same question for the pseudocode:

```
for i = 1 to n^2
    for j = 1 to i
        x = x + 1
```

Solution

arithmetic series

$$\sum_{i=1}^{n^2} \underbrace{\sum_{j=1}^{i} 1}_{} = \sum_{i=1}^{n^2} i = 1+2+3+\dots+n^2 = \frac{n^2 \cdot (n^2+1)}{2} = \Theta(n^4)$$

$\boxed{RT = \Theta(n^4)}$

- Bubble-Sort algorithm:

```
for i = 1 to n-1
    for j = n down to i+1
        if A[j] < A[j-1]
            exchange A[j] with A[j-1]
```

arithmetic series

$$\sum_{i=1}^{n-1} \underbrace{\sum_{i+1}^{j=n} 1}_{} = \sum_{i=1}^{n-1} (n-i) = (n-1)+(n-2)+(n-3)+\dots+1 = \frac{(n-1)\cdot n}{2} = \Theta(n^2)$$

$n-(i+1)+1 = n-i$

$\boxed{\text{total RT} = \Theta(n^2)}$