1.11.2017

Sorting problem

Input: a sequence of n numbers $\langle a_1, a_2, \dots, a_n \rangle$

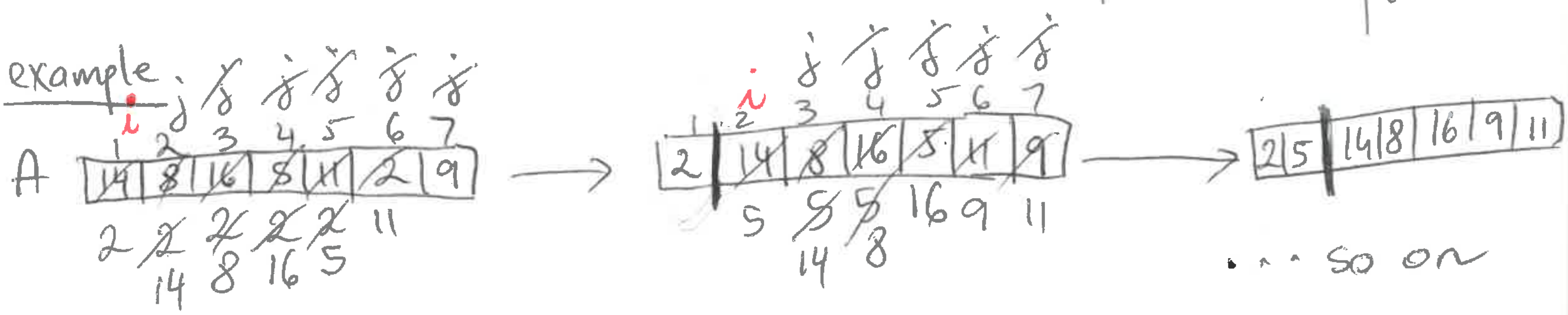
Output: arrange the numbers in increasing order $\langle a_1', a_2', \dots, a_n' \rangle$ s.t. $a_1' \le a_2' \le \dots \le a_n'$

BUBBLE-SORT(A, n)

```
1. for i = 1 to n-1
2.    // move the smalles elm. in A[i,..n] to A[i]
3.    for j = n downto i+1
4.       if A[j] < A[j-1]
5.          exchange A[j] with A[j-1]
```

line	cost	number of times
1	c_1	n
2	0	$n-1$
3	c_3	$\sum_{i=1}^{n-1}(n-i+1)$
4	c_4	$\sum_{i=1}^{n-1}(n-i)$
5	c_5	$\sum_{i=1}^{n-1}(n-i)$

example



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 elms. in the array in sorted order.

Observation

- for any loop (for, while) 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

$y = y + 3$ 4 times

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

RT analysis

RAM model

input size $= n$ = no. of elms. to be sorted

$T(n)$ = RT

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

j takes values: $n, n-1, \ldots, i+1, \underline{i}$ $\Rightarrow (n-i+1)$ times

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

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

Arithmetic Series

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$$

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

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

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

a, b, c - const

$\Rightarrow$ RT is a quadratic function of input size n

Order of growth

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

our case

$\Rightarrow an^2$

$\Rightarrow n^2$

$$\boxed{T(n) = \Theta(n^2)}$$

An algorithm is more efficient than another alg. if the worst-case running time has a smaller order of growth.

Merge Sort: $T(n) = \Theta(n \lg n)$

Bubble Sort: $T(n) = \Theta(n^2)$

$\Rightarrow$ Merge Sort is more efficient than Bubble Sort.

① Assume n-input size. Compute the RT:

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

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

$\Theta(1)$ – constant

---

② 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 (log2 n)^3
            x = x + 1
```

$n \cdot 1000 \cdot n(\log_2 n)^3 =$

$= \cancel{1000}\, n^2 (\log_2 n)^3$

$\Theta\left(n^2 (\log_2 n)^3\right)$

$\boxed{\Theta(n^2 \lg^3 n)}$

Notations

$$\lg n = \log_2 n$$

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

Solution 2 :

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

$$= n\, 1000\, n \lg^3 n = 1000\, n^2 \lg^3 n = \Theta(n^2 \lg^3 n)$$

---

(3) Same question for:

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