Muharrem Ozan Yesiller 171044033

Data Structures and Algorithms

CSE222/505 Homework-2
(spring 2020)

---

(1)

```
somefunction(rows, cols)
{
    for(i=1; i<=rows; i++) ->
    {
        for(j=1; j<=cols; j++) ->
            Print(ø) ->
        Print(newline) ->
    }
}
```

Steps/exec	freq	total
1	(rows+1)	rows+1
1	(rows) x(cols+1)	rowsxcols +rows
1	(rowsxcols)	rowsxcols
1	rows	rows
2(rowsxcols)+3rows+1		

$f(rows, cols) = 2(rows \times cols) + 3rows + 1$

Assume that A and B reel number,
$rows = A.N \in R^+$
$cols = B.N \in R^+$
} To convert to one parameter for proof equation

$$\boxed{T(N) = (AB).2.(N^2) + A.3.N + 1 = O(N^2) = \Omega(N^2) = \Theta(N^2)}$$

(1) $T(N) = O(n^2)$

proof:

$T(N) \leq c.f(N)$ if $\exists c, n_0$ $\forall N \geq n_0$

- constant can be ignored
- lower order terms are ignored

$\Rightarrow T(N) = 2N^2 + 3N + 1$

$\Rightarrow 2N^2 + 3N + 1 \leq N^2.c$

$\Rightarrow 2N + 3 + 1/N \leq N.c$ $\left(\begin{matrix} c=3 \\ n_0=4 \end{matrix}\right)$

for $c \geq 3$ and $\forall N \geq 4$ equation provided

$\boxed{T(N) = O(N^2)}$

(2) $T(N) = \Omega(N^2)$

proof:

$T(N) \geq c.f(N)$ if $\exists c, n_0$ $\forall N \geq n_0$

(constant can be ignored lower order terms are ignored)

$\Rightarrow T(N) = 2N^2 + 3N + 1$

$\Rightarrow 2N^2 + 3N + 1 \geq N^2 c$

$\Rightarrow 2N + 3 + 1/N \geq N.c$

$\Rightarrow 2N + 3 + \frac{1}{N} \geq N.2$ $\left(\begin{matrix} c=2 \\ n_0=1 \end{matrix}\right)$

$\Rightarrow 3 + \frac{1}{N} \geq 0$

equation provided

$\boxed{T(N) = \Omega(N^2)}$

(3) $T(N) = \Theta(N^2)$

Proof:

$T(N) = \Theta(h(N))$ if and only if
$T(N) = O(h(N))$ &
$T(N) = \Omega(h(N))$

$\rightarrow h(N) = N^2$

$\Rightarrow T(N) = O(N^2)$ (1)

$\Rightarrow T(N) = \Omega(N^2)$ (2)

$O(N^2) \Leftrightarrow \Omega(N^2)$

$\Rightarrow$ True

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

(4) Best, worst and average case

Best Case
$\Omega(N^2)$

Worst Case
$O(N^2)$

Average Case
$\Theta(N^2)$

$\uparrow$
$T(N) = \Theta(h(N))$ if and only if $T(N) = O(h(N))$ AND $T(N) = \Omega(h(N))$

②

```
somefunction(a, b)
{
    if (b == 0)
        return 1

    answer = a
    increment = a

    for (i = 1 ; i < b ; i++)
    {
        for (j = 1 ; j < a ; j++)
        {
            answer += increment
        }
        increment = answer
    }
    return answer
}
```

$T_1(N)$: if (b == 0) return 1

$T_2(N)$: answer = a … return answer

steps / exec	freq	total
1	1	1
1	1	1
1	1	1
2	b	2b
2	(b−1) × a	(a.b − a) × 2
2	(b−1) × (a−1)	(ab − a − b + 1) × 2
1	b−1	b − 1
2(a.b) − 2a + b + 5		

$$\boxed{T(N) = T_1(N) + T_2(N) = 2(a.b) - 2a + b + 5}$$

Assume that A and B reel number
$a = A.N \quad b = B.N \ (\in R^+)$
To convert to one parameter for proof equation

$\boxed{T_1(N) = 2}$

$\boxed{T_2(N) = (AB).4.(N^2) - 4.A.(N) + 2B.(N) + 3}$

Best Case

We can get the best case when only $T_1(N)$ is working

$T_{Best}(N) = \min(T_1(N), T_2(N))$

$T_1(N) = 1 = O(1)$

$\Rightarrow T_1(N) \leq f(N).c$
(if $\exists$ c, n_0)
($\forall N \geq n_0$)

$\Rightarrow 1 \leq 1.c$

(N cannot be negative because it is a time-related variable)

$\boxed{\text{for } c \geq 1 \text{ equation provided}}$

$T_{Best}(N) = T_1(N) = O(1)$

Worst Case

we can get the worst case when $T_1(N) + T_2(N)$

$T_{worst}(N) = T_1(N) + T_2(N)$

$\Rightarrow \underbrace{(AB)}_{\text{ignored}} 4(N^2) - \underbrace{4A(N)}_{\text{ignored}} + \underbrace{2B(N) + 3}_{\text{ignored}}$

$\Rightarrow T(N) = (4AB)N^2 + (2B - 4A)N + 3$

$\Rightarrow T(N) = O(N^2)$

$T(N) \leq f(N).c$
(if $\exists$ c, n_0
$\forall N \geq n_0$)

$(4AB)N^2 + (2B - 4A)N + 3$
($c = 4AB + 2B - 4A + 3$
$n_0 = 0$)

$\Rightarrow (4AB)N^2 + (2B - 4A)N + 3 \leq (4AB + 2B - 4A + 3).N^2$ ✓

$T_{worst}(N) = T_1(N) + T_2(N) = O(N^2)$

Average Case

$T_{Best}(N) = O(1)$

$\boxed{T_{Best}(N) = \Omega(1)}$

$\boxed{T_{worst}(N) = O(N^2)}$

$T_{average}(N) \neq \Theta(N^2)$

③ somefunction(arr[], arr_len)

```
{
    val = 0
    for(i = 0 ; i < arr_len/2 ; i++)
        val = val + arr[i]

    for(i = arr_len/2 ; i < arr_len ; i++)
        val = val - arr[i]
    if (val >= 0)
        return 1
    else
        return -1
}
```

$T_1(N)$: if / return 1; $T_2(N)$: else / return -1

steps/exec	freq	total
1	1	1
1	$(\frac{n}{2})+1$	$\frac{n+2}{2}$
2	$n/2$	n
1	$\frac{n}{2}+1$	$\frac{n+2}{2}$
2	$n/2$	n
1	1	1
1	1	1
1	1	1
1	1	1
$n+2+n+5n = 3n+7$		

define: Assume that, arr_len = n

$\boxed{T(N) = 3n+7}$

$T(N) = O(n)$

$\Rightarrow T(N) \leq c.f(N)$ (if $\exists$ c, n_0 $\forall N \geq n_0$)

$\Rightarrow T(N) = 3n+7$

$\Rightarrow f(N) = n$

$\Rightarrow 3n+7 \leq c.n$

(Assume that c = 4 $n_0 = 7$)

$\Rightarrow 3n+7 \leq 4n$

$= 7 \leq n$

$\Rightarrow (N \geq n_0) = (N \geq 7)$

$(7 \leq n) = (N \geq 7)$ ✓

$\boxed{T(N) = O(N)}$

$\boxed{T_{\text{worst case}}(N) = O(N)}$

$T(N) = \Omega(N)$

$\Rightarrow T(N) \geq c.f(N)$ (if $\exists$ c, n_0 $\forall N \geq n_0$)

$\Rightarrow f(N) = O(g(N)) \leftrightarrow g(N) = \Omega(f(N))$

$\Rightarrow f(N) \leq c_1.g(N) \leftrightarrow g(N) \geq c_2.f(N)$

* (divide by c_1 left side)

$\Rightarrow \frac{1}{c_1} f(N) \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

* (if we choose c_2 as $1/c_1$ then theorem is provided)

(Assume that $c = 1/4$ $n_0 = 7$)

$T(N) \geq \frac{1}{4}.f(n)$ ($\forall n \geq n_0$)

$3n+7 \geq \frac{1}{4}.n$ ($\forall n \geq 7$)

$\Rightarrow \frac{11n}{4}+7 \geq 0$ ($\forall n \geq 7$) ✓

$\boxed{T(N) = \Omega(N)}$

$T_{\text{Best case}}(N) = \Omega(N)$

$T(N) = \Theta(N)$

Best Case → $\Omega(N)$

Worst case → $O(N)$

Theorem:

$\Rightarrow T(N) = \Theta(h(N))$ if and only if

$\Rightarrow T(N) = O(h(N))$ AND $T(N) = \Omega(h(N))$

$\Rightarrow$ Assume that $h(N) = N$

$T(N) = O(N)$ (Provided)

$T(N) = \Omega(N)$ (Provided)

then $\boxed{T(N) = \Theta(N)}$

$T_{\text{average}}(N) = \Theta(N)$

Scanned with CamScanner

(4) Somefunction(n)
{
c = 0
for(i = 1 to n*n)
for(j = 1 to n)
for(k = 1 to 2*j)
c = c+1
return c
}

Steps/exec	freq	total
1	1	1
2	(n^2+1)	$2n^2+2$
2	$n^2(n+1)$	$2n^3+2n^2$
2	$n^2 \cdot n \cdot (2j+1)$	$4jn^3+2n^3$
2	$n^2 \cdot n \cdot 2j$	$2n^3 2j$
1	1	1
$16n^3j+4n^3+4n^2+4$		

$T(N) = 16n^3j + 4n^3 + 4n^2 + 4$

Assume that
$= n*n = A.N^2$
$= 2*j = B.N$
$= n = C.N$

$T(N) = (ABC).8.N^4 + (AC).4N^3 + 4A.N^2 + 4$

ignored ignored ignored

$\boxed{T(N) = 8N^4 + 4N^3 + 4N^2 + 4}$

$T(N) = O(N^4)$

$\Rightarrow T(N) \le c.f(N)$ (if $\exists$ c, n_0 $\forall N \ge n_0$)

ignored (Advantages of O notation)

$\Rightarrow T(N) = 8N^4 + \cancel{4N^3} + \cancel{4N^2} + \cancel{4}$

$\Rightarrow f(N) = N^4$

(if don't be ignored then we should be assummed c=20)

$\Rightarrow T(N) \le c.f(N)$

$\Rightarrow 8N^4 \le c.N^4$ (Assume that c=8 $n_0=0$)

$\Rightarrow 8N^4 \le 8N^4$ ✓

$\boxed{T(N) = O(N^4)}$

$T_{worst\ case}(N) = O(N^4)$

$T(N) = \Omega(N^4)$

$\Rightarrow T(N) \ge c.f(N)$ (if $\exists$ c, n_0 $\forall N \ge n_0$)

$\Rightarrow f(N) = O(g(N)) \leftrightarrow g(N) = \Omega(f(N))$

$\Rightarrow f(N) \le c_1.g(N) \leftrightarrow g(N) \ge c_2.f(N)$

*(divide by c_1 left side)

$\Rightarrow \frac{1}{c_1} f(N) \le g(N) \leftrightarrow g(N) \ge c_2.f(N)$

*(if we choose c_2 as $\frac{1}{c_1}$ then theorem is provided)

(Assume that c=1/8 n_0=0)

$\Rightarrow T(N) \ge \frac{1}{8} f(N)$ ($\forall N \ge n_0$)

$\Rightarrow 8N^4 \ge \frac{1}{8}.N^4$ ($\forall N \ge 0$)

$\boxed{8 \ge 1/8}$ ✓

$\boxed{\Omega(N^4) \neq T(N)}$

$T_{Best\ case}(N) = \Omega(N^4)$

$T(N) = \Theta(N^4)$

Theorem:

$\Rightarrow T(N) = \Theta(h(N))$ if and only if

$\Rightarrow T(N) = O(h(N))$

AND

$T(N) = \Omega(h(N))$

$\Rightarrow$ Assume that $h(N) = N^4$

$T(N) = O(N^4)$ (provided)

$T(N) = \Omega(N^4)$ (provided)

then $\boxed{T(N) = \Theta(N^4)}$

$T_{Average}(N) = \Theta(N^4)$

Scanned with CamScanner

5

```
otherfunction(xp, yp)
{
  temp = xp
  xp = yp          5.1
  yp = temp
}
```

$T_1(N) = 1$

$(arr\_len-1) + (arr\_len-2) + \dots + 2 + 1 = \frac{(arr\_len-1)(arr\_len)}{2}$

```
Somefunction(arr[], arr_len)
{
  for(i = 0; i < arr_len - 1; i++)          -> arr_len
  {
    min_idx = i                              -> 1 x (arr_len - 1)
    for(j = i+1; j < arr_len; j++)           -> (arr_len-1)(arr_len)/2
      if(arr[j] < arr[min_idx])              1 x (arr_len-1)(arr_len)/2
        min_idx = j                          -> 1 x ?
    otherfunction(arr[min_idx], arr[i])      -> T1(N) = 1
  }
}
```

5.2

define: Assume that $arr\_len = n$

$\Rightarrow \boxed{\frac{n^2-n}{2} = T(N)}$

$T(N) = O(N^2)$

$\Rightarrow T(N) \le c \cdot f(N)$ $\left(\begin{array}{l}\text{if } \exists\, c, n_0 \\ \forall N \geq n_0\end{array}\right)$

$\Rightarrow T(N) = (n^2 - n)/2$

$\Rightarrow f(N) = n^2$

$\Rightarrow \frac{n^2-n}{2} \le c \cdot n^2$

(assume that $c = 1$, $n_0 = 2$)

$\Rightarrow n^2 - n \le 2n^2$

$-n \le n^2$

$\Rightarrow \boxed{1 \le n}$ $\boxed{n_0 = 2}$ ✓

$T(N) = O(N^2)$

$\boxed{T_{worst\,case}(N) = O(N^2)}$

$T(N) = \Omega(N)$

$\Rightarrow T(N) \geq c \cdot f(N)$ $\left(\begin{array}{l}\text{if } \exists\, c, n_0 \\ \forall N \geq n_0\end{array}\right)$

$\Rightarrow f(N) = O(g(N)) \leftrightarrow g(N) = \Omega(f(N))$

$\Rightarrow f(N) \le c_1 \cdot g(N) \leftrightarrow g(N) \geq f(N) \cdot c_2$

*(divide by c_1 leftside)

$\Rightarrow \frac{1}{c_1} f(N) \le g(N) \leftrightarrow g(N) \geq f(N)\, c_2$

*(if we choose c_2 as $1/c_1$ then theorem is provided)

(Assume that $c = 1$, $n_0 = 2$)

$\Rightarrow T(N) \geq 1 \cdot f(N)$ $(\forall N \geq n_0)$

$\Rightarrow n^2 - n \geq n$ $(\forall N \geq 2)$

$\Rightarrow n^2 \geq 2n$ $(\forall N \geq 2)$ ✓

$T(N) = \Omega(N)$

$\boxed{T_{Best\,case}(N) = \Omega(N)}$

$T(N) = \Theta(N^2)$

Theorem:

$\Rightarrow T(N) = \Theta(h(N))$

if and only if

$\Rightarrow T(N) = O(h(N))$

AND

$T(N) = \Omega(h(N))$

$\Rightarrow$ Assume that $h(N) = N^2$

$T(N) = O(N^2)$ (provided)

$T(N) = \Omega(N^2)$ (provided)

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

$T_{average}(N) = \Theta(N^2)$

⑥ otherfunction(a, b)

```
{
1 ← if b == 0:        } T1(N)
1 ←     return 1      }
1 ← answer = a
1 ← increment = a
b ← for i = 1 to b:
    {
(b-1)xa ←   for j = 1 to a:
(b-1)xax2 ←     answer += increment
1 ←         increment = answer
    }
1 ← return answer
}
```

$\}\ T_2(N)$

⇒ 6.1 **for $T_1(N)$**

$T_1(N) = 2$

Proof $O(1)$	$2 \leq 1 \cdot c$ (for $c \geq 2$)
Proof $\Omega(1)$	$2 \geq 1 \cdot c$ (for $c \leq 1/2$)

6.2 **for $T_2(N)$**

$T_2(N) = 3ab - 3a + b + 4$

Assume that
$a = A \cdot N$ ($A \in R^+$)
$b = B \cdot N$ ($A \in R^+$)

$T_2(N) = AB \cdot 3N^2 - 3AN + BN + 4$

6.3 Proof $O(N^2)$

$T_2(N) = 3AB\,N^2 + (B-3A)N + 4$

if $(\exists c, n_0)$, $(\forall N \geq n_0)$, ($A \in R^+$, $B \in R^+$)

$T_2(N) \leq c \cdot f(N)$

$3ABN^2 + (B-3A)N + 4 \leq c \cdot N^2$ (Assume that $n_0 = 0$, $\boxed{c = 3AB + B - 3A}$)

$(3AB)N^2 + (B-3A)N + 4 \leq (3AB + B - 3A)N^2$

$\boxed{T_2(N) = O(N^2)}$

6.4 Proof $\Omega(N^2)$

$f(N) = O(g(N)) \leftrightarrow g(N) = \Omega(f(N))$

$f(N) \leq c_1\, g(N) \leftrightarrow g(N) \geq c_2 \cdot f(N)$

$f(N) \cdot \frac{1}{c_1} \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

Assume that $c_2 = 1/c_1$ then theorem is provided

$3ABN^2 + (B-3A)N + 4 \geq \frac{1}{3AB + B - 3A} \cdot N^2$

$\boxed{T_2(N) = \Omega(N^2)}$

Somefunction(arr, arr_len)

```
{
T7(N) ① for i = 0 to arr_len:
T6(N) { ② for j = i to arr_len:
T4(N) {   ③ if otherfunction(n%i, 2) == arr[i]:
T5(N) {      ④ Print(arr[i])
}
```

$T_3(N)$

for $T_3(N)$

arr_len + (arr_len − 1) + (arr_len − 2) . . . + 2 + 1
= (arr_len)(arr_len + 1)/2

① → (arr_len + 1) times, ② → (arr_len)(arr_len + 1)/2 times, ③ → $(\text{arr\_len})^2(\text{arr\_len}+1)/2$ times

④ → time depends inputs

→ In "otherfunction(a, b)", b is called 2 throughout the entire loop. So, this method always works "worst-case".

→ In "somefunction(arr, arr_len)", In order for the worst case to happen, the if condition may need to work in all steps. At least it is a guarantee of possibilities.

$T_1(N) = O(1)$ $T_1(N) = \Omega(1)$ $T_2(N) = O(N^2)$ $T_2(N) = \Omega(N^2)$ $T_3(N) = O(N^4)$	$T_{3\,\text{worst}}(N) = T_7(N) \times T_6(N) \times T_{4\,\text{worst}}(N) \times T_5(N)$ (multiply nested) ⇒ (define: arr_len = n) $T_1(N) + T_2(N)$ $\Rightarrow n + \left(\frac{n \cdot n+1}{2}\right) \times O(N^2) \times 1$ $\Rightarrow \left(\frac{n^2 + 3n}{2}\right) \times O(N^2)$ (if these two complexities are nested) ⇒ upper term is n^4 $(n^2 \times n^2)$ $\boxed{T_{3\,\text{worst}}(N) = O(N^4)}$	$T_{3\,\text{Best}}(N) = T_7(N) \times T_6(N) \times T_4(N) \times T_5(N)$ (multiply because nested) ($T_4(N)$ works worst case because b calls 2 always. Best case is same of worst case) $\boxed{T_{3\,\text{Best}}(N) = \Omega(\cancel{N^4})(N^4)}$ $\boxed{T_{\text{average}}(N) = \Theta(N^4)}$ → $T(N) = \Theta(h(N))$ if and only if $T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$

Scanned with CamScanner

7) otherfunction(x, i)

```
{
    s = 0                    -> 1
    for (j = 1; j <= i; j = j*2)  -> (log(i)+1)
        s = s + x[j]         -> (2 x log(i))
    return s                 -> 1
}
```

$T_1(N)$

$T_2(N)$

```
somefunction(arr[], arr_len)
{
    for(i = 0; i <= arr_len-1; ++i)      -> (arr_len+1)
        A[i] = otherfunction(arr, i)/(i+1)   -> (arr_len) x (2+T1(N))
    return A   -> 1
}
```

7.1

j iteration

$1+2+4 \ldots$

$2^0+2^1+2^2 \ldots 2^n$

total iteration $= n+1$

$\boxed{i = 2^n \Rightarrow \log(i) = n}$

7.2 for $T_1(N)$

(Assume that $i = n$)

$T_1(N) = 1 + \log n + 1 + 2\log n + 1$

$\boxed{= 3\log n + 3}$

$T_1(N) = O(\log n)$ $\left(\begin{matrix}\text{if } \exists\, c, n_0 \\ \forall\, N \geq n_0\end{matrix}\right)$

$\Rightarrow T_1(N) \leq c.f(N)$

$\Rightarrow T_1(N) = 3\log n + 3$

$\Rightarrow f(N) = \log n$

$\Rightarrow T_1(N) \leq c.f(N)$

$\Rightarrow 3\log n + 3 \leq c.\log n$ $\left(\begin{matrix}\text{Assume that} \\ c=4 \quad n_0=8\end{matrix}\right)$

$\Rightarrow 3\log n + 3 \leq 4\log n$ $(\forall n \geq 8)$

$\boxed{3 \leq \log n}$ ✓

$\boxed{T_1(N) = O(\log n)}$

$T_1(N) = \Omega(\log n)$

$\Rightarrow f(N) = O(g(N)) \leftrightarrow g(N) = \Omega(f(N))$

$\Rightarrow f(N) \leq c_1.g(N) \leftrightarrow g(N) \geq c_2.f(N)$

$\Rightarrow \frac{1}{c_1} f(N) \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

(if we choose c_2 as $1/c_1$ then theorem is provided)

$\left(\begin{matrix}\text{Assume that} \\ c = 1/4 \quad n_0 = 8\end{matrix}\right)$

$\Rightarrow T_1(N) \geq 1/4\, f(N)$ $(\forall N \geq n_0)$

$\Rightarrow 3\log n + 3 \geq \frac{1}{4}\log n$ $(\forall N \geq 8)$

$\frac{11\log n}{4} + 3 \geq 0$ ✓

$\boxed{T_1(N) = \Omega(\log n)}$

$T_1(N) = \Theta(h(N)) \leftrightarrow (O(h(N))$ and $\Omega(h(N))$

$T_1(N) = O(\log n)$, $T_1(N) = \Omega(\log n)$

$\boxed{T_1(N) = \Theta(\log n)}$

7.3

for $T_2(N)$

$\boxed{\text{Assume that: } arr\_len = n}$

$T_2(N) = (n+1) + n(T_1(N) + 2) + 1$

$T_2(N) = (n+1) + (3n\log n + 3n + 2n) + 1$

$\boxed{= 3.n.\log n + 6n + 2}$

$T_2(N) = O(n.\log n)$

$T_2(N) \leq c.f(N)$

$T_2(N) \leq c.(n.\log n)$

$3.n.\log n + 6n + 2 \leq (c.n\log n)$ $\left(\begin{matrix}\text{Assume that} \\ c = 11 \quad n_0 = 4\end{matrix}\right)$

$6n + 2 \leq 8n\log n$ $(\forall N \geq 4)$ ✓

$\boxed{T_2(N) = O(n\log n)}$ → worst case

$T_2(N) = \Omega(n\log n)$ $\left(\begin{matrix}\text{if } \exists\, c, n_0 \\ \forall\, N \geq n_0\end{matrix}\right)$

$T_2(N) \geq c.f(N)$

$f(N) = O(g(N)) \leftrightarrow g(N) = \Omega f(N)$

$f(N) \leq c_1.g(N) \leftrightarrow g(N) = \Omega(f(N))$

$\Rightarrow \frac{1}{c_1}.f(N) \leq g(N) \leftrightarrow g(N) \geq f(N).c_2$

$\Rightarrow T_2(N) \geq 1/11 \cdot n\log n$ $(\forall n \geq 4)$ $\left(\begin{matrix}\text{Assume that} \\ c = 1/11 \quad n_0 = 4\end{matrix}\right)$

$\Rightarrow 3n\log n + 6n + 2 \leq 1/11\, n\log n$ ✓

$\boxed{T_2(N) = \Omega(n\log n)}$ → Best Case

$T(N) = \Theta(h(N)) \leftrightarrow (\underbrace{O(h(N))}_{n\log n} == \underbrace{\Omega(h(N))}_{n\log n})$

$\boxed{T(N) = \Theta(n\log n)}$ → Average case

⑧ somefunction(n)

```
{
①← res = 0
①← j = 1
①← if(n<10)
   ①← return n+10
⑨← for(i=9; i>1; i--)
     while(n%i == 0)
        n = n/i
        res = res+j*i
        j *= 10
①← if (n)10)
   ① ← return -1
   return res
}
```

$\left.\begin{matrix}res = 0\\ j = 1\\ if(n<10)\\ return\ n+10\end{matrix}\right\} T_1(N)$

$\left.\begin{matrix}for(i=9;\ i>1;\ i--)\\ \left.\begin{matrix}while(n\%i == 0)\\ n = n/i\\ res = res+j*i\\ j *= 10\end{matrix}\right\} T_3(N)\end{matrix}\right\} T_2(N)$

$\left.\begin{matrix}if\ (n)10)\\ return\ -1\end{matrix}\right\} T_4(N)$

$return\ res \} T_5(N)$

$T_1(N)$

$T_1(N) = 4$

$T_{1_{worst}}(N) = 4 = O(1)$

$T_{1_{Best}}(N) = 3 = O(1)$

$T_1(N) = O(1), \Omega(1), \Theta(1)$

$T_4(N)$

$T_4(N) = 2$

$T_{4_{worst}}(N) = 2 = O(1)$

$T_{4_{Best}}(N) = 1 = O(1)$

$T_4(N) = O(1), \Omega(1), \Theta(1)$

$T_2(N)$

This algorithm will return 9 times. Cause of running conditions in the loop.

$\boxed{T_2(N) = 9 \times T_3(N)}$

* For this complexity, it is necessary to solve the complexity of $T_3(N)$.

$T_3(N)$

This complexity shows an irregular increase / decrease. For example, if "n" is a 9 factorial, the loop turn several times, but if n is a prime number, it does not enter at all. It does not show a specific order according to N.

Best Case = $T_1(N)$ $T_1(N) = O(1), \Omega(1), \Theta(1)$ $\boxed{T_{Best}(N) = O(1)}$	Worst Case = The worst case cannot be found precisely because T3 cannot be found fully. →	Average Case =

Scanned with CamScanner

PART 2

2.1

```
Struct Point2D
{
    x
    y
}

closestPoint (arr, len, givenPoint)      (array of Point2D)
{
    closest                                             → 1 time
    dif_x = (arr[0].x - givenPoint.x)                   → 3 time
    dif_y = (arr[0].y - givenPoint.y)                   → 3 time
    dif                                                 → 1 time
    temp_dif = sqrt( pow(dif_x, 2) + pow(dif_y, 2))     → logn time
    for i=0 to len              → (Assume that len = n) → n+1 time
    {
        dif_x = (arr[i].x - givenPoint.x)               → 3n time
        dif_y = (arr[i].y - givenPoint.y)               → 3n time
        dif = sqrt( pow(dif_x, 2) + pow(dif_y, 2))      → 3n logn time
        if (temp_dif ≥ dif)                             → n time
        {
            temp_dif = dif
            closest = arr[i]                            → ? time => O(1)
        }
    }
    return closest                                      → 1 time
}
```

$T(N) = 3n.\log n + 8n + \log n + 10$

$T(N) = O(N.\log N)$

$T(N) \leq c.f(N)$

$3n\log n + 8n + \log n + 10 \leq c.n\log n$

if ($c = 16$ $n_0 = 10$)

$\boxed{8n + \log n + 10 \leq 13n\log n}$ ✓

$T(N) = \Omega(N.\log N)$

$T(N) \geq c.f(N)$

$\left.\begin{array}{l} 3n\log n + 8n + \log n + 10 \geq \frac{1}{16}.n\log n \\ \text{if}(c = 1/16 \quad n_0 = 10) \end{array}\right\}$ ✓

$T(N) = \Theta(N.\log N)$

$T(N) = \Theta(h(N))$
if and only if
$T(N) = O(h(N))$ AND
$T(N) = \Omega(h(N))$

$T(N) = O(n\log n) \rightarrow$ worst case
$T(N) = \Omega(n\log n) \rightarrow$ Best case
$T(N) = \Theta(n\log n) \rightarrow$ average case

Scanned with CamScanner

2.2.1

```
        findLocalMin (arr, len)
        {
1 time      ← min
(len-1) times ← for i=1 to len-1 )
(len-2) x 3 times ← if((arr[i] <= arr[i+1]) && (arr[i] <= arr[i-1]) && arr[i] < min)
                min = arr[i] → ? times but O(1)
            return min → 1 times
        }
```

define: len = n

$\boxed{T(N) \cong 4n-5}$

$T(N) = O(N)$

$T(N) \leq c.f(N)$ (assume that $c=4$, $n_0=0$)

$4n-5 \leq c.N$

$\boxed{-5 \leq 0}$ ✓

$T(N) = \Omega(N)$

$f(N) \leq c_1\, g(N) \leftrightarrow g(N) = \Omega(f(N))$

$1/c_1\, f(N) \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

$\boxed{\frac{1}{c_1} = c_2}$

$T(N) \geq c.f(N)$

$\boxed{4n-5 \geq 1/4\, N}$ ✓

$T(N) = \Theta(N)$

$T(N) = \Theta(h(N))$
if and only if

$T(N) = O(h(N))$ AND
$T(N) = \Omega(h(N))$

worst $T(N) = O(N)$
best $T(N) = \Omega(N)$
$T(N) = \Theta(N)$ average

2.2.2

```
    allLocalMin (arr, len)
    {
        all[len] → 1 times
        count=0 → 1 time
        for(i=1 ; i < len-1 ; ++i) → (len-2) times
         if ( (arr[i] <= arr[i+1]) && (arr[i] <= arr[i-1]) ) → (len-1)x3 times
         {
            all[count] = arr[i] }
            count += 1          } ? times but O(1)
         }
        return all → 1 times
    }
```

define: len = N

$\boxed{T(N) \cong 4n-2}$

worst case
$T(N) = O(N)$

$T(N) \leq c\, f(N)$ $\left(\begin{matrix} c=4 \\ n_0=0 \end{matrix}\right)$

$4n-2 \leq c.N$

$4n-2 \leq 4n$ $(\forall n \geq 0)$

$\boxed{-2 \leq 0}$ ✓

Best case
$T(N) = \Omega(N)$

$f(N) \leq c_1\, g(N) \leftrightarrow g(N) = \Omega(f(N))$

$1/c_1\, f(N) \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

$1/c_1 = c_2$

$T(N) \geq c.f(N)$ $\left(\begin{matrix} c = 1/4 \\ n_0 = 0 \end{matrix} \text{ or } \begin{matrix} c = 3 \\ n_0 = 2 \end{matrix}\right)$

$\boxed{4n-2 \geq 1/4 . N}$ ✓

$\boxed{4n-2 \geq 3n\ (\forall n \geq 2)}$ ✓

average case
$T(N) = \Theta(N)$

$T(N) = \Theta(h(N))$.
if and only if
$T(N) = O(h(N))$ &
$T(N) = \Omega(h(N))$

$\boxed{h(N) = N}$

$\boxed{T(N) = O(N) \text{ and } \Omega(N)}$

$\boxed{T(N) = \Theta(N)}$

Scanned with CamScanner

2.3

```
isValid(a[], len, b)
{
  for i=0 to len                 → (len+1) time
   for j=0 to len                → (len)(len+1) time
     if(a[i]+a[j]==b)            → (len)(len) * 2 time
       return true               → (len)(len) x1 time
}  return false                  → 1 time
```

define: len = n

$T(N) = 4n^2 + n + 2$

$T(N) = O(N^2)$

$T(N) \leq c.f(N)$

$4n^2+n+2 \leq c.N^2$ ($c=5$, $n_0=2$)

$4n^2+n+2 \leq 5n^2$

$\boxed{n+2 \leq n^2}$ ✓

$O(N^2) = T(N)$ worst case

$T(N) = \Omega(N^2)$

$f(N) \leq c_1 g(N) \leftrightarrow g(N) \geq c_2 f(N)$

$1/c_1 f(N) \leq g(N) \leftrightarrow g(N) \geq c_2 f(N)$

$1/c_1 = c_2$

$T(N) \geq c.f(N)$

$\boxed{4n^2+n+2 \geq 1/5\, n^2}$ ✓ ($c=1/5$, $n_0=2$ or $c=4$, $n_0=0$)

$\boxed{4n^2+n+2 \geq 4n^2}$ ✓

$n+2 \geq 0$

$\Omega(N^2) = T(N)$ best case

$T(N) = \Theta(N^2)$

$T(N) = \Theta(h(N))$

If and only if

$T(N) = O(h(N))$ & $T(N) = \Omega(h(N))$

$h(N) = N$

$T(N) = O(N)$ and $\Omega(N)$

$\boxed{T(N) = \Theta(N)}$

$T(N) = O(N^2)$ worst case

$T(N) = \Omega(N^2)$ best case

$T(N) = \Theta(N^2)$ average case

2.4

```
isOrder (a[], len)
{
  for (i = 0 to len-1)
    if (a[i] > a[i+1])
      return false
  return true
}
```

$O(N) = \Omega(N) = \Theta(N)$

```
isValidAll (a[], len)
{
  if (!isOrder)
    return false          } T1(N)
  k = len - 1             -> 1 time
  for x = 1 to k+1)       -> (k+1) time
k time { if (!isValid(a, x, a[x]))
           return false
  return true             -> 1 time
}
```

third part algorithm

Best → $\Omega(N^2)$

worst → $O(N^2)$

Average → $\Theta(N^2)$

$$T(N) = T_1(N) + 2 + k\left(T_{\substack{\text{third}\\\text{part}}}(N)\right) + k + 1$$

$\boxed{\text{define: } k = n}$

$T_{Best}(N)$ = if the array is sorted then only "$T_1(N)$" works.

$$T_{Best}(N) = T_1(N) = \Omega(N)$$

$\boxed{= \Omega(N)}$. $\boxed{T_{Best}(N) = \Omega(N)}$

$$T_{worst}(N) = O(N) + n(O(N)) + n + 3$$

$$= O(N) + O(N).O(N) + O(N) + O(1)$$

$$= \underbrace{2O(N)}_{\text{ignored}} + O(N^2) + \underbrace{O(1)}_{\text{ignored}}$$

$\boxed{= O(N^2)}$ $\boxed{T_{worst}(N) = O(N^2)}$

$T_{average}(N)$ =

$T(N) = O(N^2)$

$T(N) = \Omega(N)$

$T(N) = \Theta(N^2)$

$\boxed{T_{average}(N) = \Theta(N^2)}$