# 2383 A2

1) a) $n^3+8n^2+6n+2$

$n^3+8n^2+6n+2 \leq c \cdot n^3$

$\leq n^3+8n^3+6n^3+2n^3$

$\leq 17n^3$

Let $c=17$, $n_0=1$

We have $n^3+8n^2+6n+2 \leq c \cdot n^3$, for all $n \geq n_0=1$

$\therefore n^3+8n^2+6n+2$ is $O(n^3)$

$n^3+8n^2+6n+2 \geq c \cdot n^3$

$\geq n^3$ (removed +ve terms)

Let $c=1$, $n_0=1$

We have $n^3+8n^2+6n+2 \geq c \cdot n^3$, $n \geq n_0=1$

$\therefore n^3+8n^2+6n+2$ is $\Omega(n^3)$

$\therefore n^3+8n^2+6n+2$ is $\Theta(n^3)$

---

b) $8n^2-6n+2$ Check

$8n^2-6n+2 \leq c \cdot n^2$

$\leq 8n^2+2$

$\leq 8n^2+2n^2$

$\leq 10n^2$

Let $c=10$, $n_0=1$

We have $8n^2-6n+2 \leq c \cdot n^2$, for all $n \geq n_0=1$

$\therefore 8n^2-6n+2$ is $O(n^2)$

$8n^2-6n+2 \geq c \cdot n^2$

$\geq 8n^2-6n+2$

$\geq n^2+(7n^2-6n+2)$

Let $c=1$, $n_0=1$

We have $8n^2-6n+2 \geq c \cdot n^2$, $n \geq n_0=1$

$\therefore 8n^2-6n+2$ is $\Omega(n^2)$

$\therefore 8n^2 - 6n + 2$ is $\Theta(n^2)$

---

c) $2n^2 - 3n + 50$ Check

$2n^2 - 3n + 50 \leq c \cdot n^2$

$\leq 2n^2 + 50$

$\leq 2n^2 + 50n^2$

$\leq 52n^2$

Let $c = 52, n_0 = 1$

We have $2n^2 - 3n + 50 \leq c \cdot n^2$, for all $n \geq n_0 = 1$

$\therefore 2n^2 - 3n + 50$ is $O(n^2)$

$2n^2 - 3n + 5 \geq c \cdot n^2$

$\geq 2n^2 - 3n + 2$

$\geq n^2 + n^2 - 3n$

$\geq n^2$

Let $c = 1, n_0 = 1$

We have $2n^2 - 3n + 50 \geq cn^2, n \geq n_0 = 1$

$\therefore 2n^2 - 3n + 5$ is $\Omega(n^2)$

$\therefore 2n^2 - 3n + 5$ is $\Theta(n^2)$

---

2) a)

```
p ← 1                          1
for i ← 1 to n² do             n² + 1
    p ← p x i                  n²
return p                       1
```

Total: $2n^2+3$

$O(n^2)$

---

b)

```
s ← 0                          1
for i ← 1 to n do              n
    For j ← i to n do          n - i
        s ← s+1
return s                       1
```

it runs $n+(n-1)+(n-2) \ldots\ldots 1$

Total: $\frac{n(n+1)}{2}$

$O(n^2)$

---

c) $x \leftarrow 0$ — 1
$j \leftarrow 1$ — 1

while $(j^3 \leq n)$ { — $\log \log n$
$x \leftarrow x+1;$ — $2 \log \log n$
$j \leftarrow j+1;$ — $2 \log \log n$
}

Total: $5 \log \log n + 2$

$O(\log \log n)$

---

d) $x \leftarrow 0$ — 1
$j \leftarrow n$ — 1

while $(j \geq 1)$ { — $\log n$
$x \leftarrow x+1$ — $\log n$
$j \leftarrow 2j/3$ — $\log n$
}

Total: $3 \log n + 2$

$O(\log n)$

e) $x \leftarrow 0$ | 1
$j \leftarrow 2$ | 1
while ($j <= n$) { | $\log\log n$
$x \leftarrow x+1$ | $\log\log n$
$j \leftarrow j^3$ | $\log\log n$
}

| iteration : | 1st | 2nd | 3rd | 4th | kth |
|---|---|---|---|---|---|
| j : | 2 | $2^3$ | $(2^3)^3$ $2^{3^2}$ | $(2^{3^2})^3$ $2^{3^3}$ | $2^{k^{\log_2}(\log(n))}$ |

$O(\log\log n)$

---

F) $x \leftarrow 0$ | 1
$j \leftarrow n$ | 1
while ($j \geq 1$) | n
For $i \leftarrow 1$ to $j$ do | n
$x \leftarrow x+1$ | n
$j \leftarrow j-2$ | n
return $x$ | 1
$4n+3$

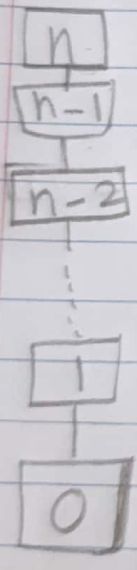
$O(n)$

---

3) a)
```
if (n=0)
    return m;
else
    return fun1(n-1, n+m);
```

check

This algorithm calculates the sum of numbers from n to 0 inclusive then it adds m to that sum and returns the result of the last addition

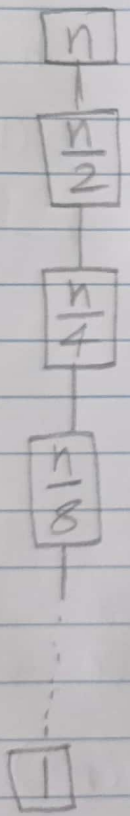



| | RT |
|---|---|
| Fun1(n, m) | n |
| fun1(n-1, m) | n-1 |
| fun1(n-2, m) | n-2 |
| | |
| fun1(1, m) | 1 |
| Fun1(0, m) | 1 |

O(n)

b) Fun2(n)
if(n=1)
    return 0
else
    return 1 + Fun2($\frac{n}{2}$)

This program returns the number of times we recur (by dividing n by 2) until n = 1




$2^k$

$2^{k-1}$

$2^{k-2}$

$2^{k-4}$

$2^0$

$n = 2^k$

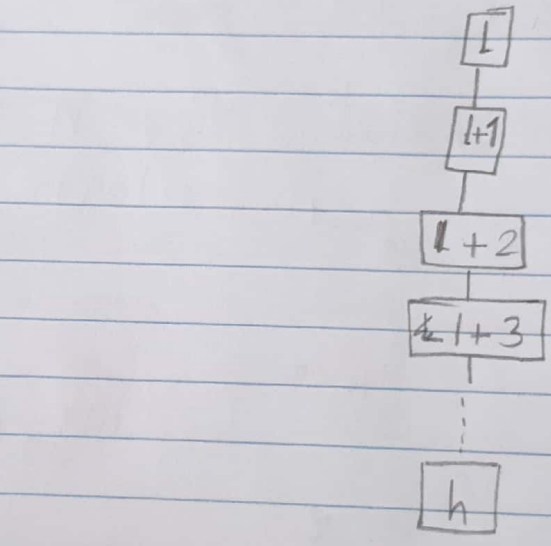
$\log n = \log 2^k$

$= k \log 2$

$\log_2 n = k$

$O(\log n)$

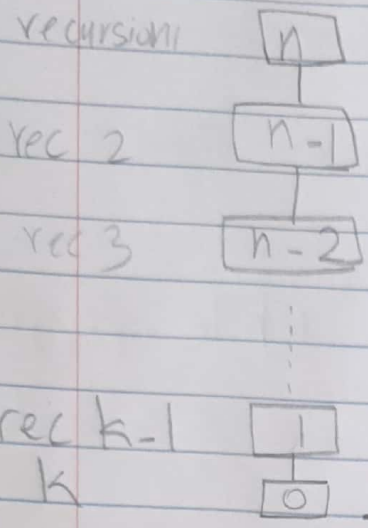
c) The algorithm takes in array A and l and h integers. The base case for this program is $l \geq h$. If the base case is true we stop recurring. If not, l is assigned to minindex (integer) [int] and A[l] is assigned to minivalue. We then go in a loop to check if minivalue is the smallest number in the array. If not, minivalue & minindex are altered. After the loop, we swap A[l] and A[minindex] and recur with the same array and int h while incrementing l by 1.




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

$$O(n^2)$$

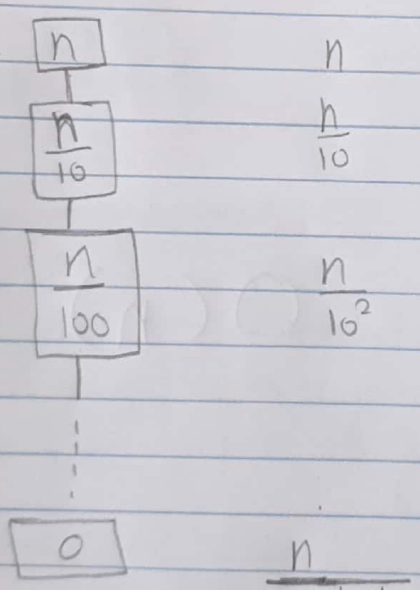
4) Time complexity




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

$$O(n^2)$$

5) Reverse algorithm




$$\log 1 = \log n + (k-1)\log \frac{1}{10}$$

$$k-1 = \frac{\log 1 - \log n}{\log \frac{1}{10}}$$

$$k = \frac{\log 1 - \log n}{\log \frac{1}{10}} + 1$$

$$O(\log n)$$

palindrome Algorithm

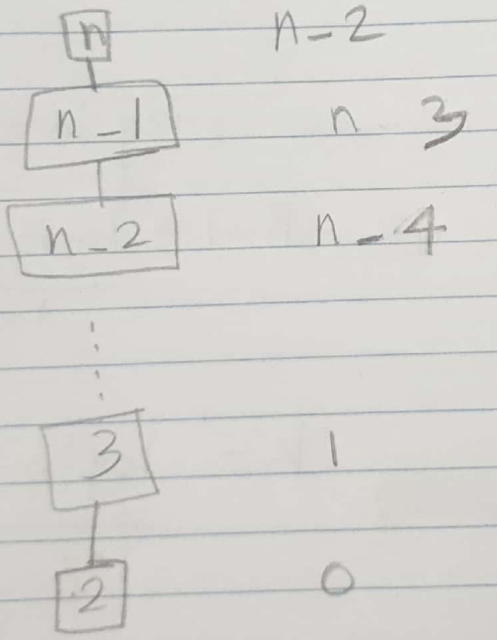
O(1)

Time complexity of the whole program

O(log n)

---

6)




$O(n^2)$

$$\frac{n(n+1)}{2}$$

4) Algorithm: sortStack(s) / input: stack s / output: sorted stack

```
~~sortStack(s)~~
if(!s.isEmpty())
    temp ← s.pop();
    sortStack(s);
    sortNInsert(s, temp);
return s;
```

```
Algorithm: sortNInsert(s, temp)
input: Stack s, int temp

if(s.isEmpty() || temp < s.peek())
    s.push(temp);
else
    temp2 ← s.pop();
    sortNInsert(s, temp);
    s.push(temp2);
```

5) Algorithm: reverse(n, rev) / input: int n, int rev
output: reversed n

```
if(n=0)
    return rev;
rev ← rev*10 + (n%10);
rev ← reverse(n/10, rev);
return rev;
```

Algorithm palindrome(w) / input: int w
output: "palindrome" or "Not palindrome"

```
rw ← 0;

rw ← reverse(w, rw);
if (w = rw)
    return "palindrome";
return "Not palindrome";
```

6) Algorithm isRecursion(A[],n) / input: int A[], int n
output: sorted array using insertion sort.

```
if (n<2)
    return;
isRecursion(A, n-1);
last ← A[n-1];
i ← n-2;
for(i ← n; i ≥ 0 && A[i] > last)
    A[i+1] = A[i];
    {i--}  (implied)
A[i+1] = last;
```