Lab 2

1. Determine the asymtotic running time:

```
int [] arrays(int n)
{
    int[] arr = new int[n];          ⟶ 1
    For(int i=0; i<n; ++i)           ⟶ n+3
        {arr[i] = 1;                 ⟶ 1
    }
For(int i=0; i<n; ++i)
    {
        For(int j = i; j<n; ++j)
            {arr[i] += arr[j] + i+j;
    }                                ⟶ O(n²)
    }
    return arr;                      ⟶ 1
}
```

Total running time = $O(n) + O(n^2) = O(n^2)$

2. pseudo code for Merge Sort:

Algoritum to merge(ML, M2)

Input: Two sorted array ML & M2.

Output: Sorted Array after merging ML & M2.

```
p ← new array
i ← 0
j ← 0
k ← 0
```

~~While (m1.length~~

While (i < m1.length && j < m2.length) do

if (m1[i] < m2[i] then

p[k] = m1[i]

i ← i+1

else

p[k] = m2[j]

j ← j+1

k ← k+1

Copy array m1 to new array p until index m1.length

Copy array m2 to new array p until index m2.length - j.

return new array p.

B. Asymtotic running time = O(n)

3 Assume the running time $T(n)$ for a particular algorithm satisfies the following recurrence relation.

$T(1) = a$

$T(2) = b$

$T(n) = T(n-1) + T(n-1) + T(n-2) + c$

where, a, b and c are constants (for some $a, b, c > 0$)

$T(n) = 2T(n-1) + T(n-2) + c$

$\geq 2T(n-2) + T(n-2) + c$

$\geq 3T(n-2)$

Suppose, $T(1) = a$, $T(2) = b$, and $T(n) \geq 3T(n-2)$.

Define a recurrence $s(1) = a$, $s(2) = b$, $s(n) = 3s(n-2)$.

Then for all n, $T(n) \geq s(n)$

This is obvious, for $n = 0$ or 1. Assume, $T(k) \geq s(k)$ when $k < n$. Then,

$T(n) \geq 3T(n-2) \geq 3s(n-2) = s(n)$

So, $s(n)$ is $\Theta(g(n))$, then $T(n)$ is $\Omega(g(n))$.

given that, $s(1) = a$, $s(n) = 3s(n-2)$?

$s(1) = a$

$s(3) = 3 * s(1) = 3 * a$

$s(5) = 3 * s(3) = 3 * 3 * a = 3^2 a$

$s(7) = 3 * s(5) = 3 * 3 * 3 * a = 3^3 a$

$s(9) = 3 * s(7) = 3 * 3 * 3 * 3 * a = 3^4 a$

$s(n) = 3^{n/2} * a = (\sqrt{3})^n * a$, which is $\Theta((\sqrt{3})^n)$

④ ⇒ Power set Algorithm.

⑤ ⇒ Devise an iterative algorithm for computing the Fibonacci numbers and compute its running time.

Algorithm to find Fibonacci numbers iteratively.

Input: Any number.
output: Fibonacci number

~~Set void~~
$i \leftarrow 0$
$a \leftarrow 0$
$b \leftarrow 1$
~~$c \leftarrow 0$~~
~~while ($i < n$)~~
While $i < n$ do
    $a = b$
    $b = c$
    $c = a + b$

return c

Running time for above algorithm is

$O(n)$

(6) ⇒ Find the asymptotic running time using the master formula.

$T(n) = T(n/2) + n;\ T(1) = 1$

we have,

$$T(n) = \begin{cases} d & \text{if } n = 1 \\ aT(\lceil n/b \rceil) + cn^k & \text{otherwise.} \end{cases}$$

So, in our problem,

$a = 1,\ b = 2,$ ~~$c = 1,\ k = 1$~~

n^1 means $k = 1$

Then,

if $a < b^k$ then $\Theta(n^k)$

$1 < 2^1$ then $\Theta(n^1)$ true

$\therefore \Theta(n)$ is asymptotic running time.