B. K. DHARANEESH
CSE – B II
230701072

7-24 DAA

ASSIGNMENT – I

1. To prove: $100n+5 \in O(n^2)$

w.k.t, if $f(n) \in O(g(n))$,

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

Here, let $f(n) = 100n+5$; $g(n) = n^2$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} \Rightarrow \lim_{n \to \infty} \frac{100n+5}{n^2}$$

Using L. Hospital rule,

$$\lim_{n \to \infty} \frac{100}{2n} \Rightarrow \frac{100}{\infty} = 0.$$

$f(n)$ has smaller order of growth than $g(n)$. Hence $f(n) \in O(g(n))$ // Hence proved.

2. MatrixMultiplication(A[0,...n-1, 0...n-1], B[0...n-1, 0...n-1]){

```
for i ← 0 to n-1 do:
    for j ← 0 to n-1 do
        C[i,j] ← 0.0
    for k ← 0 to n-1 do
        C[i,j] ← C[i,j] + A[i,k] * B[k,j]
```

return C.

input size : n^2

basic operation : $C[i,j] \leftarrow C[i,j] + A[i,k] * B[k,j]$

iterations : 0 to $(n-1)^3$ (or) 1 to $(n)^3$

$$\Rightarrow \sum_{i=1}^{n^3} 1 \Rightarrow (n^3 - 1) + 1 = n^3$$

$$T(n) \in O(n^3)$$

3. Binary(n) {
count ← 1
while n > 1 do
count ← count + 1
n ← ⌊n/2⌋
return count
}

input size: n (integer)
basic operation: n > 1
iterations: 1 ~~n times~~ to n/2

$$\Rightarrow \sum_{i=1}^{n/2} 1 \Rightarrow \left(\frac{n}{2} - 1\right) + 1 \Rightarrow \frac{n}{2} = \frac{1}{2}n$$

$T(n) \in O(n)$

4. Secret(A[0..n-1]) {
~~for~~
minval ← A[0];
maxval ← A[0];
for i ← 1 to n-1 do
if A[i] < minval
minval ← A[i]
if A[i] > maxval
maxval ← A[i]
return maxval - minval
}

input size: n
basic operation: A[i] < minval
iterations: 1 to n-1

$$\Rightarrow \sum_{i=1}^{n-1} 1 \Rightarrow (n-1-1) + 1 = n-1$$

$T(n) \in O(n)$

a) This algorithm compute the difference between maximum and minimum value of an array

b) the basic operation is

A[i] < minval (or) A[i] > maxval

because these repeat the most and prioritise above assignment statements

c) basic operation is executed n times

d) Efficiency class of this algorithm is O(n)

e) there are no such Alternative methods that has better efficiency class than this algorithm because to find min and max element of an array we have 2 methods

i) sort the array and take the first and last elements (time (least): O(nlogn))

ii) use bruteforce approach (time : O(n))

since nlogn > n, efficient method is brute force approach, which is followed in the given algorithm.