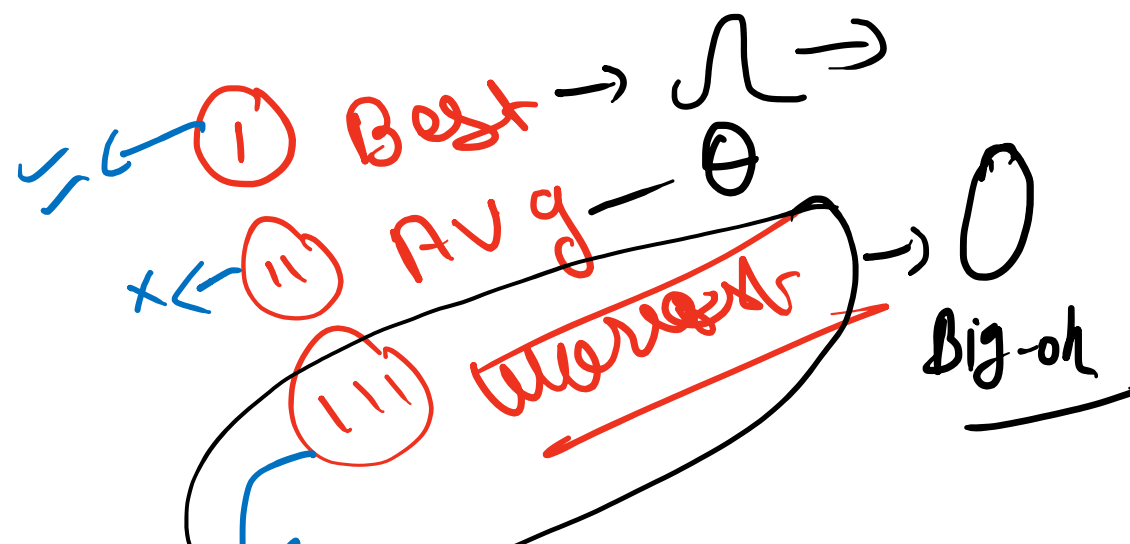
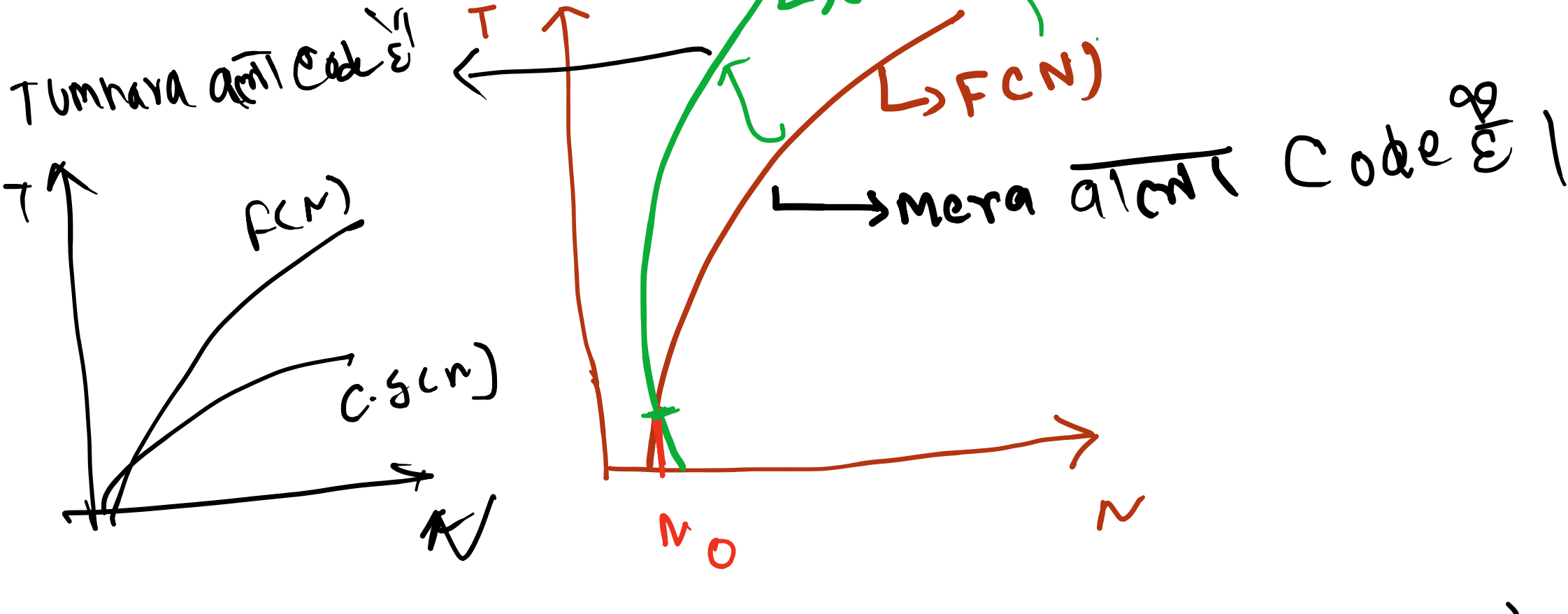


Time & Space Complexity

asymptotic notation vs Experimental Way



input size vs time

$$f(N) \leq c.g.c(N) \quad N \geq N_0$$
  
$$\rightarrow O(g(N))$$

$$f(N) > c.g.c(N) \rightarrow \Theta(N)$$

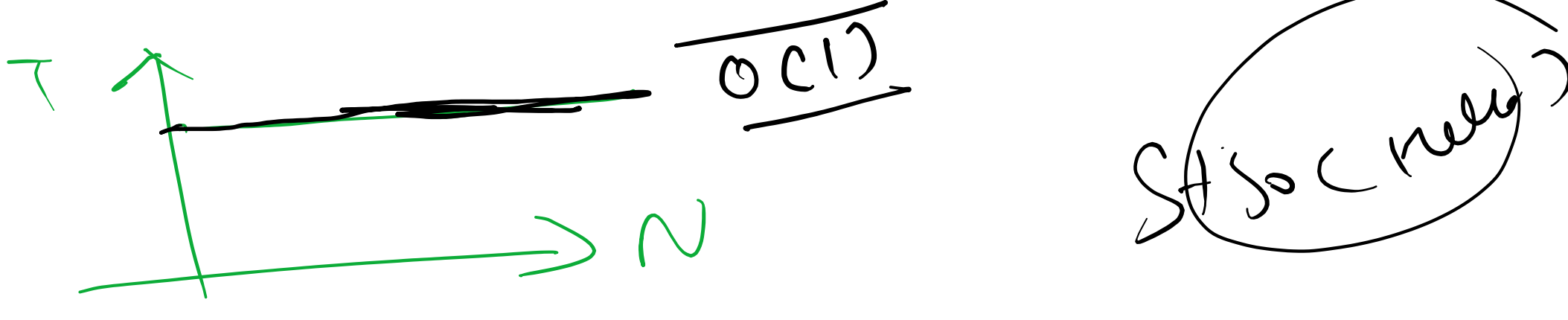
#  $T(N) = 3n^2 + 2n + 5 \rightarrow O(n^2)$

$$\lim_{n \rightarrow \infty} \frac{2^n}{n} = \infty$$
  
$$2^n \log n$$

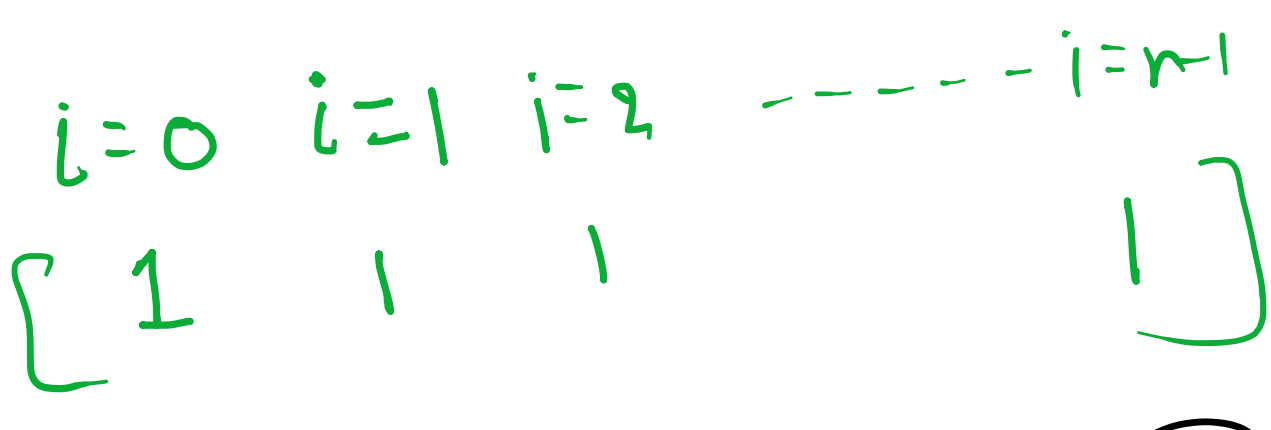
$$T(N) = 2^n + 7n + 7(2^n \log n)^2 + 5n^2 + 7$$

$$T(N) = 4e^N + n^2 + 7 \rightarrow O(e^N)$$

constant time

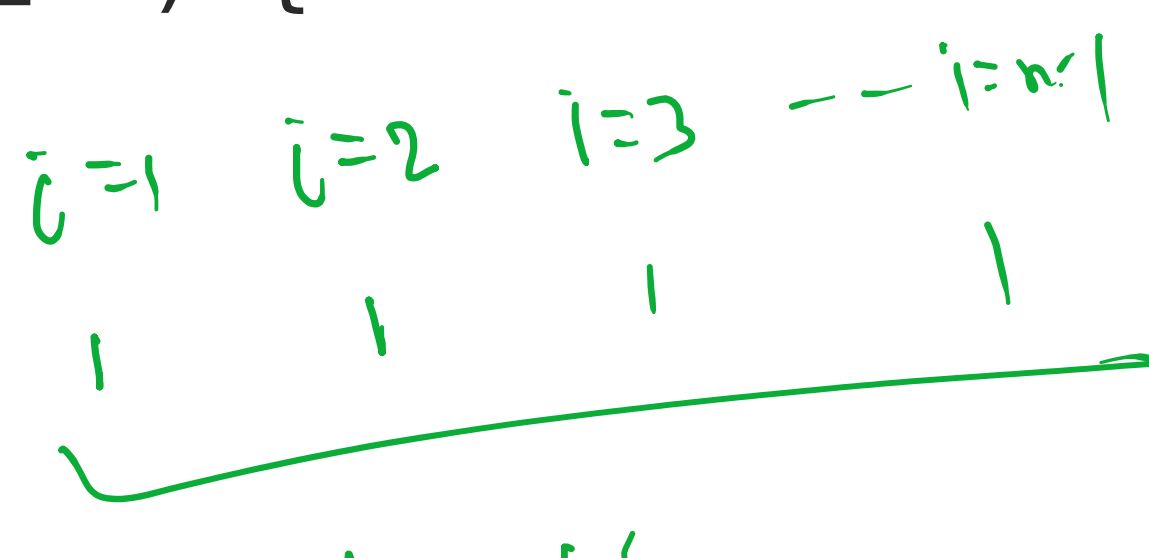


```
public static int Search(int[] arr, int item) {  
    for (int i = 0; i < arr.length; i++) {  
        if (arr[i] == item) {  
            return i;  
        }  
    }  
    return -1;  
}
```



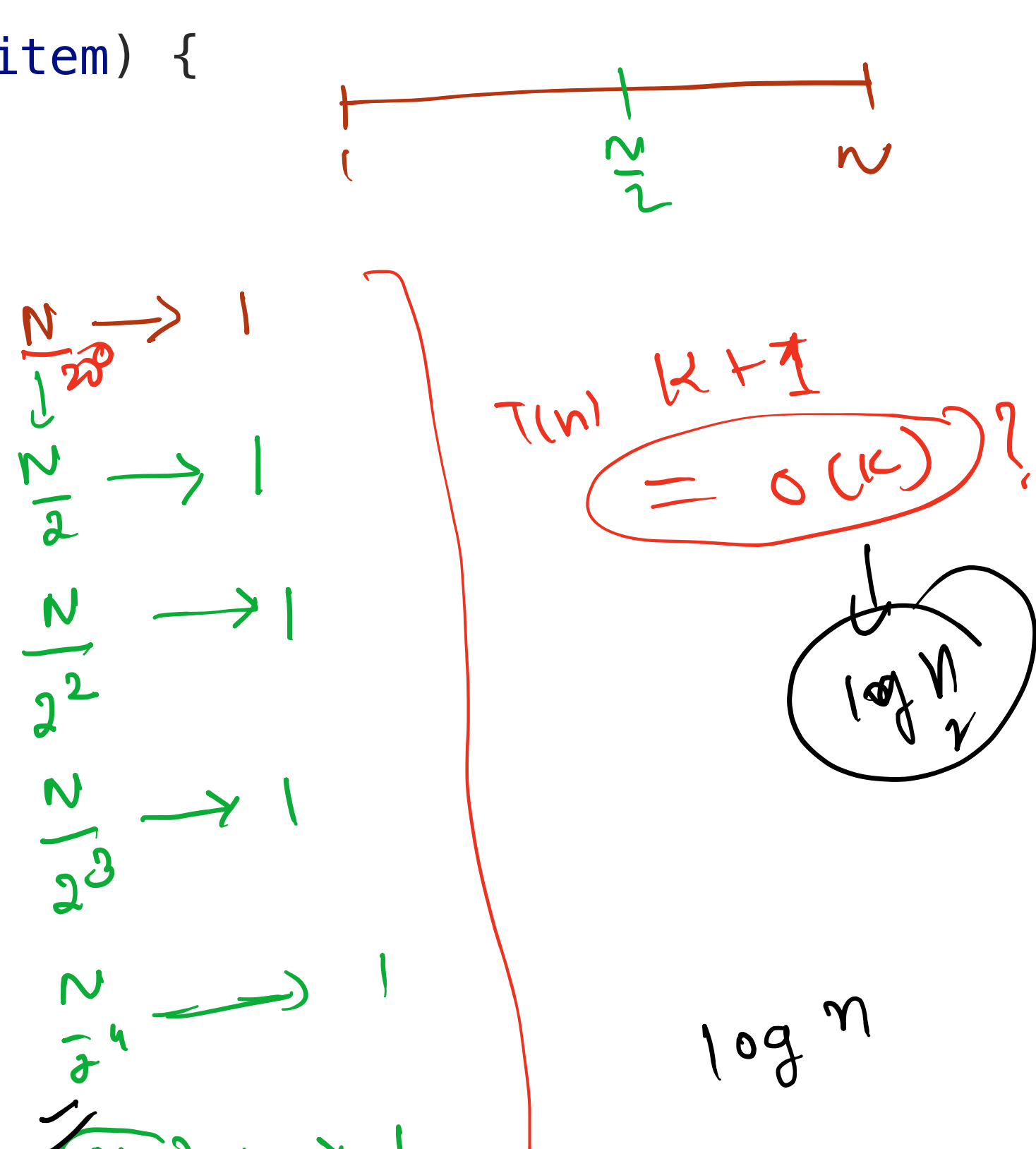
$$T(N) = N \quad O(N) \quad T(N) = 3n^2 \quad O(n^2)$$

```
public static int Maximum(int[] arr) {  
    int max = arr[0];  
    for (int i = 1; i < arr.length; i++) {  
        if (arr[i] > max) {  
            max = arr[i];  
        }  
    }  
    return max;  
}
```

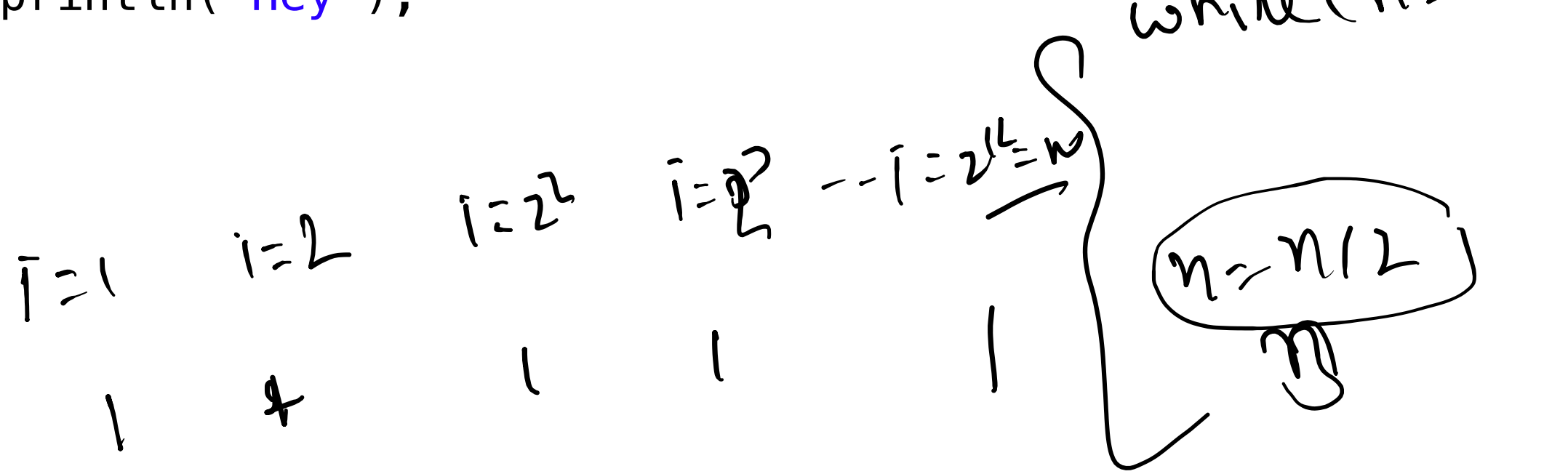


$$T(N) = (n-1) + 1 + 1 = O(n)$$
  
$$T(N) = n$$
  
$$T(N) = n-1$$

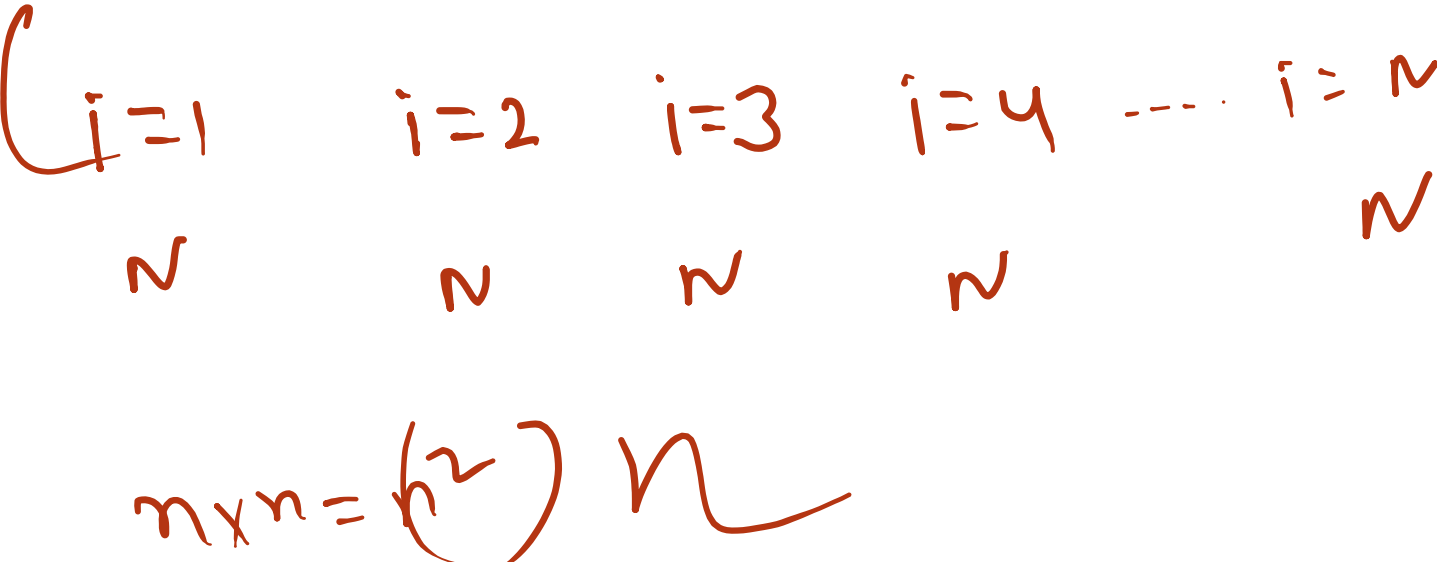
```
public static int Search(int[] arr, int item) {  
    int lo=0;  
    int hi=arr.length-1;  
    while (lo <= hi) {  
        int mid = (lo+hi)/2;  
        if (arr[mid] == item) {  
            return mid;  
        } else if (arr[mid] > item) {  
            hi = mid-1;  
        } else {  
            lo = mid+1;  
        }  
    }  
    return -1;  
}
```



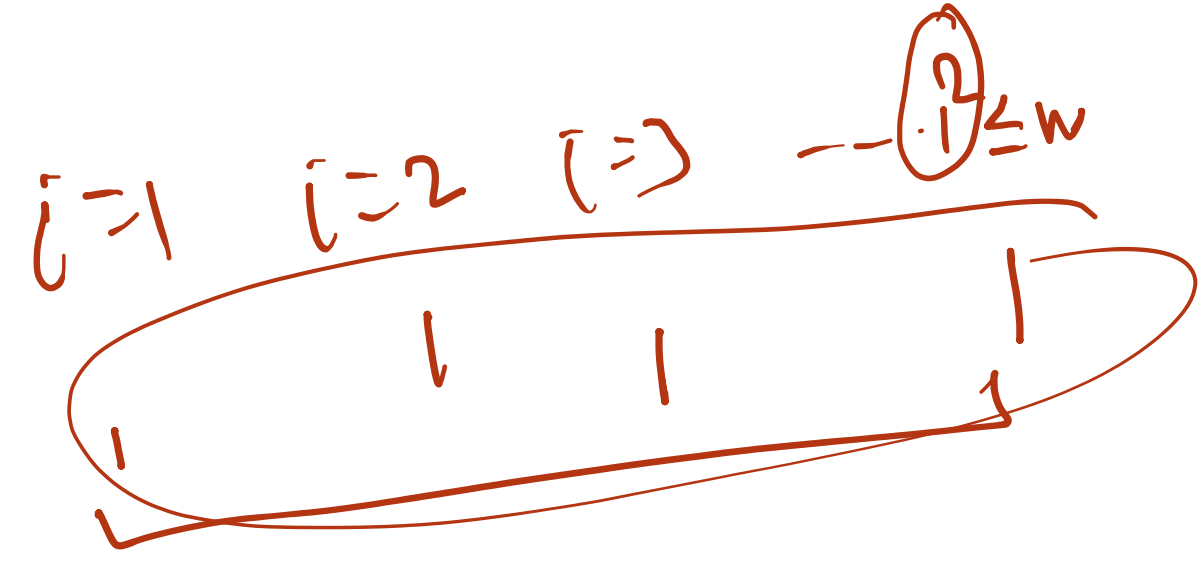
```
while (i < n) {  
    System.out.println("Hey");  
    i *= 2;  
}
```



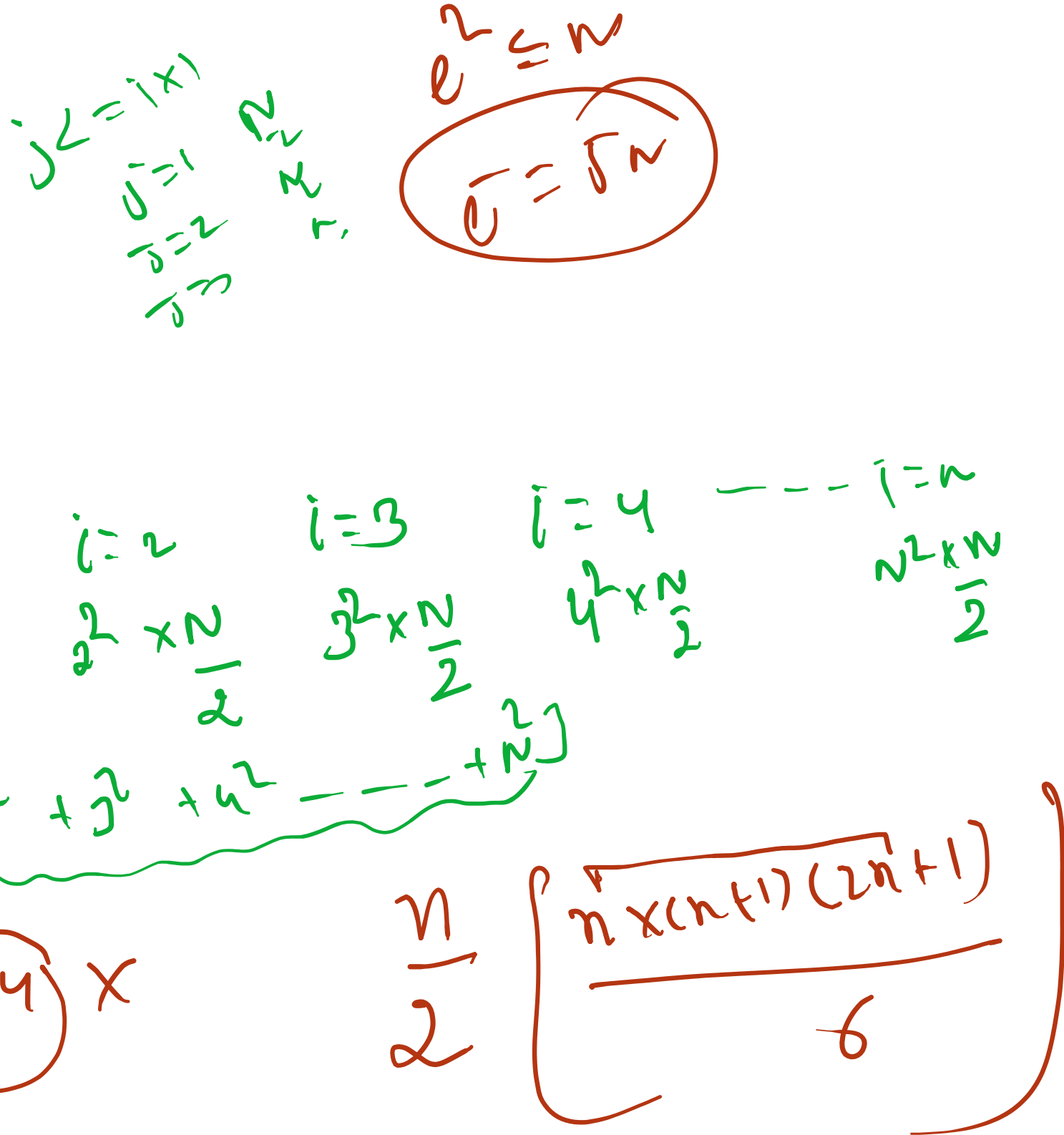
```
for (i = 1; i <= n; i++) {  
    for (int j = 1; j <= n; j++) {  
        System.out.println("hey");  
    }  
}
```



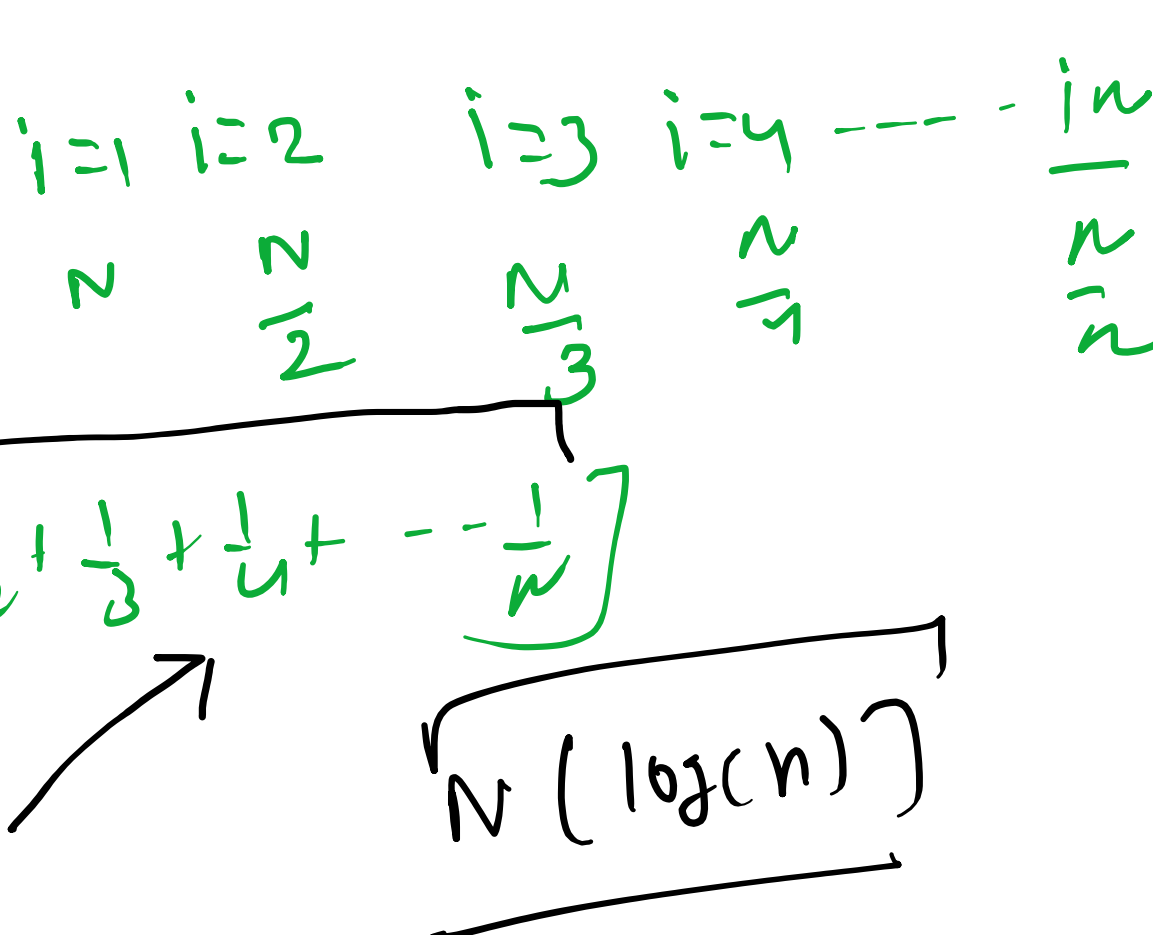
```
for (i = 1; i * i <= n; i++) {  
    System.out.println("hey");  
}
```



```
for (i = 1; i <= n; i++) {  
    for (int j = 1; j <= i * i; j++) {  
        for (k = 1; k <= n / 2; k++) {  
            System.out.println("hey");  
        }  
    }  
}
```

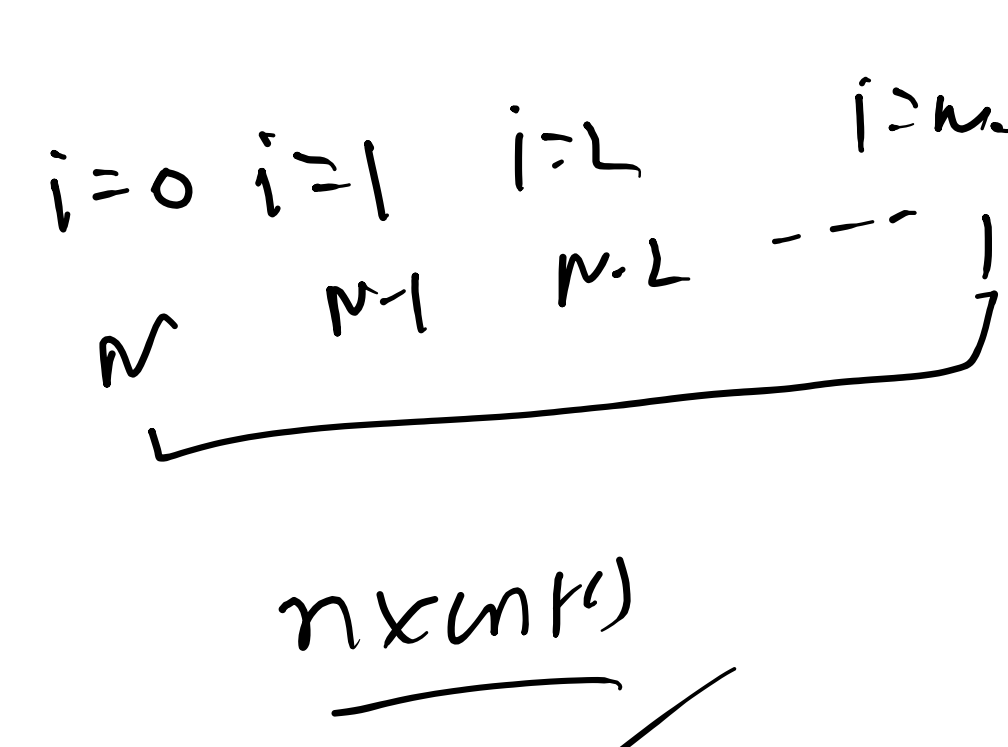


```
for (i = 1; i <= n; i++) {  
    for (int j = 1; j <= n; j += i) {  
        System.out.println("hey");  
    }  
}
```



1 2 3 4 5 ... n

```
public static int Maximum(int[] arr) {  
    int ans = Integer.MIN_VALUE;  
    for (int i = 0; i < arr.length; i++) {  
        int sum = 0;  
        for (int j = i; j < arr.length; j++) {  
            sum += arr[j];  
            ans = Math.max(ans, sum);  
        }  
    }  
    return ans;  
}
```



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99

