Triangle Task

ID	NAME
20220025	احمد علاء احمد محمد مصطفى (TeamLeader)
20220042	احمد محمود محمد عبدالحميد
20220020	احمد حمدي صالح عبدالقوي
20220057	اسامة احمد فوزي محمد
20220051	ادهم باسم كمال محمود
20220442	محمود صبحي عيد محمود
20220150	رامز عماد عبدالرحمن مهدي فرج

BRUTEFORCE PSEUDOCODE

```
Algoritm isTriangular(A[], N) {
    if (N < 3)
        return 0;

    for i = 0 to N - 2 step 1 do  ---> N-1
    {
        for  j = i + 1 to N - 1 step 1 do ---> N-i-1
        {
            for k = j + 1 to N step 1 do  ---> N-j
                {
                if (A[i] + A[j] > A[k] && A[j] + A[k] >
A[i] && A[k] + A[i] > A[j]) then ---> 1
                {
                    return 1;
                }
            }
        }
}
    return 0; ---> 1
}

The WorstCase TimeComplexity = O (n³)
The BestCase  TimeComplexity = O (1)
The AverageCase TimeComplexity = O (n³)
```

BRUTEFORCE ANALYSIS

$$\boxed{\sum_{i=0}^{n-2} \boxed{\sum_{j=i+1}^{n-1} \sum_{k=j+2}^{n} 1}}$$

② ①

$$① \rightarrow \sum_{j=i+1}^{n-1} n-i-1 = \sum_{j=i+1}^{n-1} n-1 - \sum_{j=i+1}^{n-1} i$$

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

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

$$= \frac{2[n^2-n-(n-1)-(ni-i)] - n^2-n+in-i}{2}$$

$$= \frac{2n^2-2n-2n+2\cancel{-ni+i}-n^2-n\cancel{+in-i}}{2}$$

$$= \frac{n^2-5n+2}{2} = n^2-n \text{ (by ignoring constants)}$$

$$② \rightarrow \sum_{i=0}^{n-2} n^2 - \sum_{i=0}^{n-2} n = (n^2)(n-1) - (n)(n-1)$$

$$= n^3-n^2-n^2+n = n^3-2n^2+n$$

$$\boxed{= O(n^3)}$$

MERGESORT RECURRENCE METHOD

```
Algorithm isTriangular(A[], int N) {

    if (N < 3) then ---> n-1
        return 0;

    mergeSort(A, 0, N - 1); ---> n log n

    for i = 0 to N - 2 step 1 do ---> n-1
    {
        if ((long long)A[i] + A[i + 1] > A[i + 2])
            return 1;
    }

    return 0;
}

The T(n) Recurrence Raltion Will be = T(n) = T(n log n) + O(n)

The WorstCase TimeComplexity = O (n log n)
The BestCase  TimeComplexity = O (1)
The AverageCase TimeComplexity = O (n log n)
```

MERGESORT RECURRENCE METHOD ANALYSIS
ITERATIVE METHOD

$T(n) = 2T(n/2) + 2 \cdot O(n), \quad T(1) = 1$

$T(n) = 2T(n/2) + 2 \cdot c \cdot n$

$T(n/2) = 2T(n/4) + 2 \cdot c \cdot (n/2)$

$T(n) = 2(2T(n/4) + 2 \cdot c \cdot (n/2)) + 2 \cdot c \cdot n$

$T(n) = 4T(n/4) + 2 \cdot c \cdot n + 2 \cdot c \cdot n = 4T(n/4) + 4 \cdot c \cdot n$

$T(n/4) = 2T(n/8) + \cancel{2} \cdot c \cdot (n/\cancel{4})$

$T(n) = 4(2T(n/8) + c \cdot (n/2)) + 4 \cdot c \cdot n$

$= 8T(n/8) + 2 \cdot c \cdot n + 4 \cdot c \cdot n = 8T(n/8) + 6 \cdot c \cdot n$

$$\boxed{T(n) = 2^k T(n/2^k) + \sum_{i=0}^{k-1} 2 \cdot c \cdot n}$$

$\cancel{T(n/2^k) = T(1)} \quad T(n/2^k) = T(1)$

$n = 2^k \rightarrow k = \log_2 n$

$$T(n) = 2^{\log_2 n} + \sum_{i=0}^{\log_2 n - 1} 2 \cdot c \cdot n = n + 2 \cdot c \cdot n(\log_2 n \cancel{-1-0+1})$$

$$= n + 2 \cdot c \cdot n \cdot \log_2 n$$

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

MERGESORT RECURRENCE METHOD ANALYSIS
MASTER METHOD

$T(n) = 2T(n/2) + 2.O(n) = 2T(n/2) + 2.c \cdot n$

$\hookrightarrow a=2, b=2, k=1, p=0$

$\hookrightarrow \log_2 2 = 1 = k \longrightarrow \Theta(n^k \log^{p+1} n) \longrightarrow \Theta(n \log n)$

"Master method"

RECURSIVE ALGORTIHM PSEUDOCODE

```
Algorithm isTriangular(A[],  N,  i,  j, k) {
    if (i >= N - 2) then {
        return 0;
     }
    if (j >= N - 1) then
    {
        return isTriangular(A, N, i + 1, i + 2, i + 3);
    }
    if (k >= N) then
    {
        return isTriangular(A, N, i, j + 1, j + 2);
    }

    if (A[i] + A[j] > A[k] && A[j] + A[k] > A[i] && A[k]
+ A[i] > A[j]) then
    {
        return 1;
    }

    return isTriangular(A, N, i, j, k + 1);
}

The WorstCase TimeComplexity = O (3ⁿ)
The BestCase  TimeComplexity = O (1)
The AverageCase TimeComplexity = O (3ⁿ)
```

RECURSIVE ALGORITHM ANAYLSIS

ITERATIVE METHOD

$T(n) = T(n-1) + T(n-2) + T(n-3) + C$

$T(n) = 3T(n-1) + C$

$T(n-1) = 3T(n-2) + C$

$T(n) = 3(3T(n-2) + C) + C = 9T(n-2) + 4C$

$T(n-2) = 3T(n-3) + C$

$T(n) = 9(3T(n-3) + C) + 4C$

$T(n) = 27T(n-3) + 13C$

$T(n) = 3^k T(n-k) + \sum_{i=0}^{k-1} 3^i C$

$n - k = 0 \longrightarrow \boxed{n = k}$

$T(n) = 3^n + C \sum_{i=0}^{n-1} 3^i = 3^n + C.\left(\frac{3^n - 1}{3 - 1}\right)$

$= 3^n + C.\left(\frac{3^n - 1}{4}\right)$

$= O(3^n)$

RECURSIVE ALGORITHM ANAYLSIS

MASTER METHOD

$T(n) = T(n-1) + T(n-2) + T(n-3) + c$

$T(n) = 3T(n-1) + c$

$\rightarrow a = 3, b = 1, k = 0$

$\rightarrow a > 1 \longrightarrow O(a^n n^k) \longrightarrow O(3^n)$

"Master Method"

THE COMPARISON

TIMECOMPLEXITY	BRUTEFORCE	MERGESORT ALGORTIHM	RECURSIVE ALGORITHM
BESTCASE	$\Omega(1)$	$\Omega(1)$	$\Omega(1)$
WORSTCASE	$O(n^3)$	$O(n \log n)$	$O(3^n)$
AVERAGECASE	$\Theta(n^3)$	$\Theta(n \log n)$	$\Theta(3^n)$

The First Algorithm Take $O(n^3)$ in The Worst Case

The Second Algorithm Take $O(n \log n)$ in The Worst Case

The Third Algorithm Take $O(3^n)$ in The Worst Case

Regarding To The Increaseing of Growth Functions ,
$3^n > n^3 > n \log n$, n log n has the Slowest Growth and the Fastest Runtime

So The Best Algorithm To Choose Is The ‘ MergeSort Algorithm ’