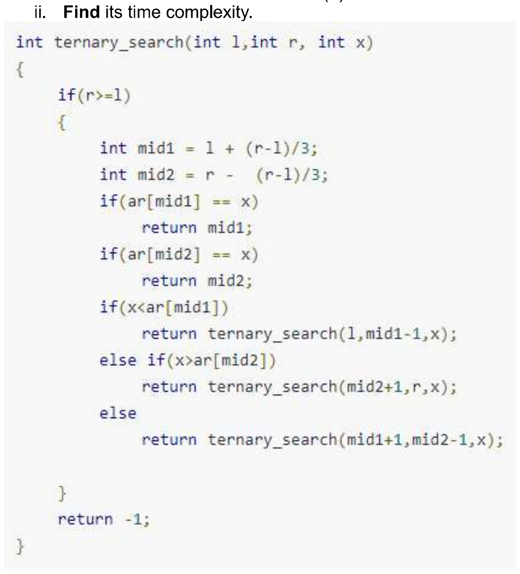
CSE221: Algorithms

Assignment 1

1. **Time Complexity (30 points)**
   a. **Sort** the following functions in ascending order of their growth. For example- $n^2 < n^3$, etc.
   Hint: you can plot the functions in any graphing tool like Desmos to visualize their growth.
   5
   2^n, $\log n$, $\log\log n$, n^2, n, $\sqrt{n}$, $n!$, n^3, $n^{3/2}$, $n \log n$, e^{n+1}, $n^2 \log n$
   b. **Prove** the following (you can use any method):
   2.5x4
      i. $log(n!) = O(n\,log\,n)$
      ii. $n^2 + 15n - 3 = \theta(n^2)$
      iii. $T(n) = 4T(n/2) + n = \theta(n^2)$
      iv. $T(n) = 2T(n/2) + n^3 = \theta(n^3)$
   c. **Show** that the following code is $\theta(n)$:
   5
```
count = 0;
for (i=1, i<=n; i*=2)
   For (j=1, j<=i; j++)
       count++;
```
   d. For the following code-
   5+5
      i. **Derive** the recurrence function: T(n)
      ii. **Find** its time complexity.



```
int ternary_search(int l,int r, int x)
{
    if(r>=l)
    {
        int mid1 = l + (r-l)/3;
        int mid2 = r -  (r-l)/3;
        if(ar[mid1] == x)
            return mid1;
        if(ar[mid2] == x)
            return mid2;
        if(x<ar[mid1])
            return ternary_search(l,mid1-1,x);
        else if(x>ar[mid2])
            return ternary_search(mid2+1,r,x);
        else
            return ternary_search(mid1+1,mid2-1,x);

    }
    return -1;
}
```

2. **Searching (10 + 5 = 15 points)**
   a. You are given two arrays: Arr1 and Arr2.
   Arr1 will be given sorted. For each element v in Arr2, you need to **write a pseudo code** that will print the number of elements in Arr1 that is **less than or equal** to v. For example: if I give you two arrays of size 5 and 4
   5 4 [size of two arrays]
   Arr1 = 1 3 5 7 9
   Arr2 = 6 4 8
   The output should be: 3 2 4

   Firstly, you should search how many numbers are there in Arr1 which are less than 6. There are 1,3,5 which are less than 6 (total 3 numbers). So the answer for 6 will be 3.
   After that, you will do the same thing for 4 and 8 and output the corresponding answers which are 2 and 4. **Your searching method should not take more than *O(log n)* time.**

Sample input	**Sample output**

5 5
1 1 2 2 5
3 1 4 1 5
	4 2 4 2 5

   b. **Show** the calculation of the time complexity for your written code.
3. **Sorting**
   You recently got a job as a library assistant. You are assigned to sort an archive of files. Files are marked by how old they are by years and you need to put the most recent files on the left.
   Here's the stack of files-

File 5
File 20
File 9
File 15
File 13
File 25
File 50
File 45
File 40
File 35

   a. **Sort** the files using a suitable algorithm. (Show step by step simulation)
   5
   b. **What** made you choose this sorting algorithm over others?

4. Simulation
10+5
Look at the two functions [`Merge-and-Count & Sort-and-Count`]. You are also given an array `Awesome_Array= 3 7 10 14 18 19 2 11`

```
Merge-and-Count (A, B)
{
        curA = 0; curB = 0;
        count = 0;
        mergedList = empty list
        while (not at end of A && not at end of B)
        {
                a = A[curA]; b = B[curB];
                if (a < b)
                        append a to mergedList;
                        curA++;
                else
                        append b to mergedList;
                        curB++;
                        count = count + number of elements left in A
        }
        if (at end of A)
                append rest of B to mergedList;
        else
                append rest of A to mergedList;

        return (count, mergedList);
}

Sort-and-Count(L)
{
        if list L has one element
                return (0, L)

        Divide the list into two halves A and B
        (rA, A) ← Sort-and-Count(A)
        (rB, B) ← Sort-and-Count(B)
        (rC, L) ← Merge-and-Count(A, B)
        total_count = rA + rB + rC
        return (total_count, L)
}
```

Now, answer the questions:

a. **Write** down the output values: **`(total_count & L)`** for `Sort-and-Count(Awesome_Array)`.
b. **What** do you think the returned value of **`total_count`** represents? Why?