National University of Computer and Emerging Sciences

Assignment No. 1

Design and Analysis of Algorithms

Time Complexity

CS2009

Fall 2024

Deadline: September 18, 2024

Submission Instructions:

- All problems must be solved and submitted on Google Classroom.
- Use A4 size papers.
- Submit the scanned Assignment in PDF form on Google Classroom.
- This is an individual assignment.
- **Plagiarism is strictly prohibited.**
- **Please do not use any AI tool and do not copy from others.**
- **Just analyze the problem and then brainstorm the solution.**

Question 01 (Asymptotic Notations): (100 Marks)

What is the **best-case** and **worst-case** time complexity of all the **below mentioned** code snippets in terms of **asymptotic notation** (Big O, Ω, and Θ)?

1.

```
int sum = 0;
for (int i = 0; i < n; i++) {
    for (int j = i; j > 0; j /= 2)
    {
        sum++;
    }
}
```

2.

```
int sum = 0;
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        for (int k = j; k < n; k += j) {
        sum++;
        }
    }
}
```

3.

```
k = m = 0;
for( i=1; i<n; i*=2) {
    k++;
}
for (j=0; j<k*k; j*=2) {
    m++;
}
```

4.

```
i = n, tot=0;
while (i>1) {
    tot += i;
    i=i/2;
}
```

5.

```
for(int i=0;i<n;++i) {
    for(int j=0;j<n;j=j++) {
        for( int k=0;k<10;k++) {
            cout<<"bawa"<<endl;
        }
    }
}
```

6.

```
void function( int n) {
    if(n<s){
        cout << n << endl;
    }
    else {
        for(int i=0; i<n;i=i/k) {
            cout<<i<<endl;
        }
    }
}
```

7.

```
int binarySearch(int arr[], int l, int r, int x) {
    while (l <= r) {
        int m = l + (r - l) / 2;
        if (arr[m] == x)
            return m;
        if (arr[m] < x)
            l = m + 1;
        else
            r = m - 1;
    }
    return -1;
}
```

8.

```
void multiplyMatrices(int a[3][3], int b[3][3], int result[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            result[i][j] = 0;
            for (int k = 0; k < 3; k++) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}
```

9.

```
void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod) {
    if (n == 1) {
        cout << "Move disk 1 from rod " << from_rod << " to rod " << to_rod << endl;
        return;
    }
    towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
    cout << "Move disk " << n << " from rod " << from_rod << " to rod " << to_rod << endl;
    towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
```

10.

```
int fibonacci(int n) {
    if (n <= 1)
        return n;
    return fibonacci(n - 1) + fibonacci(n - 2);
}
```

Question 02 (Iteration Method): (50 Marks)

For each code snippet that involve **recurrence relations**, you need to provide an analysis of its **time complexity** using **Big O notation**, considering the **iterative steps.**

1.

```
int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

2.

```c
int factorial(int n) {
    int result = 1;
    for (int i = 2; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

3.

```c
int power(int x, int n) {
    int result = 1;
    while (n > 0) {
        if (n % 2 == 1)
            result *= x;
        x *= x;
        n /= 2;
    }
    return result;
}
```

4.

```c
void computePrefixSum(int arr[], int n, int prefixSum[]) {
    prefixSum[0] = arr[0];
    for (int i = 1; i < n; i++) {
        prefixSum[i] = prefixSum[i - 1] + arr[i];
    }
}
```

5.

```c
int findMin(int matrix[][N], int rows, int cols) {
    int min = matrix[0][0];
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] < min) {
                min = matrix[i][j];
            }
        }
    }
    return min;
}
```