CS/CPE590: Algorithms
Spring 2023
Assignment2

Asymptotic Notations & Recurrence Relations

1. (10 Points) Consider the following pseudo-code and analyze the best and the worst-case running time of this by providing the form of the function the running time takes in each of the cases (Analyze statement-by-statement and use the length of the array A as n):

Algorithm 1 AdjacentDuplicates(A)

```
for i = 0 to A.length − 2 do
  if A[i] == A[i + 1] then
    return 1
  end if
end for
return 0
```

2. (10 Points) Consider the following pseudo-code and analyze the best and the worst-case running time of this by providing the form of the function the running time takes in each of the cases (Analyze statement-by-statement and use the length of the array A as n):

Algorithm 2 Linear-Search(A,v)

```
1: i = NIL
2: for j = 1 to A.length do
3:     if A[j] = v then
4:         i = j
5:         return i
6:     end if
7: end for
8: return i
```

3. (5 points) Find the upper bound for $f(n) = n^4 + 10n^2 + 5$. Prove your answer by giving values for the constants c and n_0. Choose the smallest integral value possible for c.

4. (10 points) Find an asymptotically tight bound for $f(n) = 3n^3 - 2n$. Prove your answer by giving values for the constants c_1, c_2, and n_0. Choose the tightest integral values possible for c_1 and c_2.

5. (5 points) Is $3n - 4 \in \Omega\left(n^2\right)$? Prove your response.

6. (20 points) Express the complexity of the following functions with the most appropriate notation

```
int function1(int n) {
    int count = 0;
    for (int i = n / 2; i <= n; i++) {
        for (int j = 1; j <= n; j *= 2) {
            count++;
        }
    }
    return count;
}
```

```
int function2(int n) {
    int count = 0;
    for (int i = 1; i * i * i <= n; i++) {
        count++;
    }
    return count;
}
```

```
int function3(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            for (int k = 1; k <= n; k++) {
                count++;
            }
        }
    }
    return count;
}
```

```
int function4(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            count++;
            break;
        }
    }
    return count;
}
```

```
int function5(int n) {
    int count = 0;
    for (int i = 1; i <= n; i++) {
        count++;
    }
    for (int j = 1; j <= n; j++) {
        count++;
    }
    return count;
}
```

7. (10 points) Find an asymptotically tight bound for $f(n) = \frac{n^2}{2} - 7n$. Prove your answer by giving values for the constants c_1, c_2, and n_0. Choose the tightest integral values possible for c_1 and c_2 [Hints: Solve it for the smallest possible integer n_0].

8. (5*4 = 20 Points) Solve the following recurrence relations (using the backward substitution method with a demonstration of all the steps):
a) $x(n) = 3x(n-1)$ for $n > 1, \quad x(1) = 4$
b) $x(n) = x(n-1) + n$ for $n > 0, \quad x(0) = 0$
c) $x(n) = x(n/2) + n$ for $n > 1, \quad x(1) = 1$
d) $x(n) = x(n/3) + 1$ for $n > 1, \quad x(1) = 1$

9. (10 Points) For each function below, compute the recurrence relation for its running time and then use the Master Theorem to find its complexity by specifying the different terms of the term explicitly:

```
int f(int arr[], int n) {
    if (n == 0) {
        return 0;
    }
    int sum = 0;
    for (int j = 0; j < n; ++j) {
        sum += arr[j];
    }
    return f(arr, n / 2) + sum + f(arr, n / 2);
}
```

```
void g(int n, int arrA[], int arrB[]) {
    if (n == 0) {
        return;
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            arrB[j] += arrA[i];
        }
    }
    g(n / 2, arrA, arrB);
}
```

10. (2*5 = 10 points) Use the Master Theorem to find the complexity of each of the following recurrence relations (show all the steps and the values of different terms to apply the theorem):

$$\textbf{a)}\ T(n) = 4T\left(\frac{n}{2}\right) + n^2$$

$$\textbf{b)}\ T(n) = 3T\left(\frac{n}{3}\right) + \sqrt{n}$$

11. (2*10 = 20 points) Use the Recursion tree method to find appropriate notation for the complexity of each of the following recurrence relations (draw the recursion tree and show all the steps to your solution):

$$\textbf{a)}\ T(n) = 7T\left(\frac{n}{5}\right) + cn^3$$

$$\textbf{b)}\ T(n) = 5T(n-6) + c$$

12. (2*10 = 20 points) Implement (in C++) a recursive version of the Bubble sort and Insertion sort algorithms on an array of integers (array size: 100) (Implement a function for each of the sorting algorithms, follow the provided code template). Provide sufficient comments to document your code.

Remarks:

- The assignment has to be completed individually. No collaboration is allowed between students. No code from online resources is allowed to be used. Any sign of collaboration or use of online materials will result in a 0 and be reported to the Graduate Academic Integrity Board. You have to strictly follow the provided template. The late submission policy is applicable to the assignment as specified in the course syllabus.
- You have to submit a typed report containing your solutions to problems 1 to 11 as a pdf file.
- The assignment is quite comprehensive and will take some time. Start early.
- You have to make sure your program works as expected in the following online compiler: https://www.onlinegdb.com/online_c++_compiler.
- **Submit a zip file containing your report(.pdf) file and code(.cpp) file**.