CS 302 – Assignment #02

Purpose: Learn concepts regarding algorithm analysis.
Due: Monday (9/10) → Must be submitted on-line before class.
Points: 100

Reading/References:

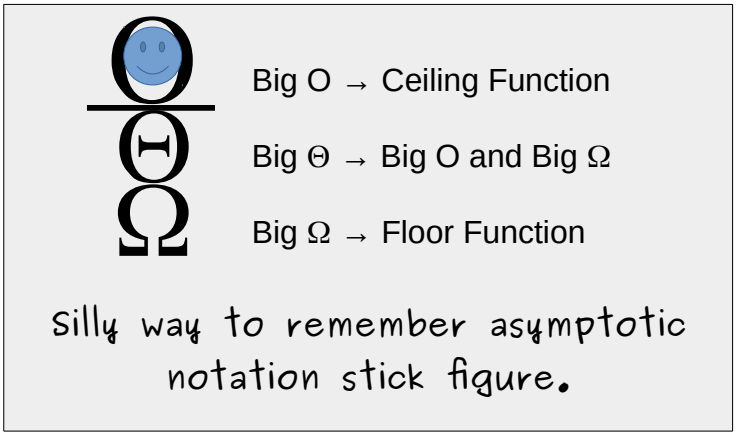
Chapter 2, Mathematical Preliminaries
Chapter 3, Algorithm Analysis

Assignment:

Answer the following questions.

1) Provide a formal definition (as per the text) for each of the following. (6 pts)

   Big Oh → $O(f(n))$
   Big Omega → $\Omega(g(n))$
   Big Theta → $\Theta(h(n))$



2) Provide a short informal description for each of the following explaining what the formal definition means. (6 pts)

   Big Oh → $O(f(n))$
   Big Omega → $\Omega(g(n))$
   Big Theta → $\Theta(h(n))$

3) Order the following functions by growth rate: (6 pts)

$$4n^2,\ 2^n,\ \log n,\ 730,\ 2n,\ 5n^3,\ n\log n,\ \sqrt{n},\ \frac{2}{n},\ 2^{\frac{n}{2}},\ n^2\log n,\ n^3$$

4) Given a standard, correct binary search algorithm on a sorted array. (6 pts, 3 pts each)
   a) What is the Big-O analysis of the running time for finding an element.

   b) Explain the answer for **a)**. including a comparison to a sequential search.

5) Given an unsorted linked list, what is the Big-O running time for inserting a new element in the unsorted linked list. (2 pts)

6) Assuming the data structure used is sorted linked list, what is the Big-O running time for inserting a new element in the sorted linked list. (2 pts)

7) What is the Big-Oh of an algorithm to determine if a number is prime (assuming an efficient algorithm). (3 pts)

8) Given the following code fragments; (6 pts)
Note, assume all variables are declared and initialized appropriately.

```
int count1=0;
for (int i=1; i<len; i++)
        for (int j=1; j<len; j++)
                count1++;

int count2 = 0;
for (int i=1; i<=len; i++)
        for (int j=1; j<=i; j++)
                count2++;
```

Provide a Big-Oh analysis for each code fragment.

9) Given the following code fragments; (6 pts)
Note, assume all variables are declared and initialized appropriately.

```
int count1=0;
for (int k=1; k<=len; k*=2)
        for (int j=1; j<=len; j++)
                count1++;

int count2 = 0;
for (int k=1; k<=len; k*=2)
        for (int j=1; j<=k; j++)
                count2++;
```

Provide a Big-Oh analysis for each code fragment.

10) For each of the following program fragments, what is the execution time complexity in terms of Big-Oh. *Note*, assume that all arrays are appropriately declared and sized. (27 pts, 3 pts each)

```
int func(int n) {
        sum = 0;
        for (int i=0; i<n; i++)
                for (int j=0; j<n; j++)
                        if (j != n%2)
                                sum++;
        return  sum;
```

```
}


int func(int n) {
        int sum = 0;
        for (int i=1; i<=len; i++)
                for (j=1; j<=i; j*=2)
                        sum++;
        return   sum;
}



int func(int arr[], int len) {
        int sum = 0;
        for (int i=1; i<=len; i++)
                sum += arr[i];
        return   sum;
}

int func(int n) {
        int sum = 0;
        for (int i=1; i<=n; i++)
                for (j=1; j<=n; j++)
                        sum++;
        return   sum;
}

int func(int arr[], int len) {
        int output=0;
        for (int i=0; i<len; i++)
                output += arr[i];
        for (int i=0; i<10; i++)
                output += arr[i];
        for (int i=0; i<len; i++)
                output += arr[i] / 2;
        for (int i=0; i<10; i++)
                output += arr[i];
        return   output;
}

void func(int **arr, int len) {
        int sum = 0;
        for (int i=0; i<len; i++)
                for (int j=0; j<len*len; j++)
                        sum += arr[i][j];
}

int func(int **arr, int len) {
        sum = 0;
        for (int i=0; i<len; i++)
                for (int j=0; j<i*i; j++)
                        for (int k=0; k<j; k++)
```

```
                                        sum += arr[i][j][k];
        return sum;
}

int func(int **arr, int len) {
        sum = 0;
        for (int i=0; i<len; i++)
                for (int j=0; j<len*len; j++)
                        if (j%i == 0)
                                for (int k=0; k<j; k++)
                                        sum += arr[i][j][k];
        return   sum;
}
int func(int *arr, int len) {
        sum = 0;
        for (int i=0; i<len; i++)
                for (int j=0; j<len; j++)
                        sum += arr[j];
        for (int k=0; k<len; k++)
                sum += arr[k];
        return   sum;
}
```

11) Given the follow two algorithms to computer the integer ***pow(x,y)*** functions; (6 pts)

Algorithm #1

```
long long power( long long x, int n ) {
     long long  ans = 1;
    for (int i=1; i<=n; i++)
        ans = x * ans;
    return ans;
}
```

Algorithm #2

```
long long power(long long x, int n ) {
    long long  y;

    if (n == 0 || n == 1)
        return 1;
    if (n%2 == 0)
        y = power(x, n/2);
        return y*y;
    else
        y = power(x, n/2);
        return y*y*x;
}
```

a) What is the Big Oh for each algorithm?

12) Given the following function, (6 pts, 3 pts each).

Algorithm #1

```
int rPerrin(int n) {
        if (n == 0)
                return 3;
        if (n == 1)
                return 0;
        if (n == 2)
                return 2;
        return (rPerrin(n-2) + rPerrin(n-3));
}
```

Algorithm #2

```
int iPerrin(int n) {
        if (n == 0)
                return 3;
        if (n == 1)
                return 0;
        if (n == 2)
                return 2;
        int ans = 1, a=3, b=0, c=2;
        for (int i=0; i<n-3; i++) {
                ans = b + c;
                a = b;
                b = c;
                c = ans;
        }
        return ans;
}
```

a) What is the time complexity in terms of ***Big Oh*** of each algorithm?

b) What is the space complexity in terms of ***Big Oh*** of each algorithm?

13) Given the following function, (6 pts, 3 pts each).

Algorithm #1

```
int rFact(int n) {
        if (n == 0)
                return 1;
        return (n * rFact(n-1));
}
```

Algorithm #2

```
int iFact(int n) {
        if (n == 0)
                return 1;
        int ans = 1;
        for (int i=0; i<n; i++)
                ans = ans * i;
        return ans;
}
```

a) What is the time complexity in terms of ***Big Oh*** of each algorithm?

b) What is the space complexity in terms of ***Big O*** of each algorithm?

14) According to the text; (6 pts, 3 pts each)

a) What is a ***space/time trade-off*** principal?

b) Provide an example of a space/time trade-off.

15) With regard to the algorithms from assignment #1; (6 pts, 3 pts each)

a) What is the time complexity in terms of Big-Oh for algorithm 1 (recursive).

b) What is the time complexity in terms of Big-Oh for algorithm 2 (dynamic).

Submission:

When complete, submit:

- A copy of the answers. Must use PDF format.

Assignments received after the due date/time will not be accepted.

You may re-submit as many times as desired. Each new submission will require you to remove (delete) the previous submission.