CS 302 – Assignment #02

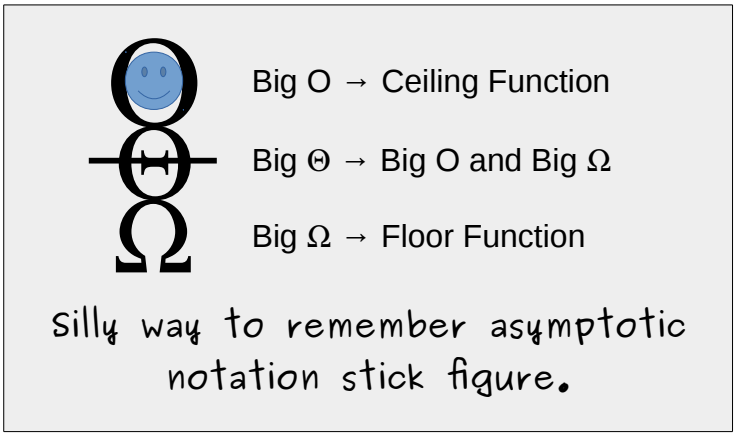
Purpose: Learn concepts regarding algorithm analysis.
Due: Tuesday (1/31) → Must be submitted on-line before class.
Points: 100

Reading/References:

Chapter 2, Mathematical Preliminaries
Chapter 3, Algorithm Analysis

Assignment:

Answer the following questions.



1) For each of the following;

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

   Provide both a
   a) Formal definition (as per the text). (7 pts)
   b) Short informal description explaining what the formal definition means. (7 pts)

2) Order the following functions by growth rate (slowest to fastest growth): (10 pts)

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

3) With regard to *Big Theta*, Θ; (9 pts)
   a) Informally, explain the difference between the ***Big O*** notation and ***Big Θ*** notations.
   b) When it is appropriate to use *Big Theta* notation and when it is not appropriate?
   c) Does every algorithm have a ***Big*** Θ running time? Explain why or why not.

4) Consider a standard, correctly implemented binary search algorithm; (8 pts)
   a) Assuming an array data structure, what is the expected running time (in terms of ***Big O***)?
   b) Assuming a linked list data structure, what is the impact?
   c) Assuming an array data structure, when would a sequential search be required?
   d) Assuming an array data structure, what is the expected running time (in terms of ***Big O***) for a sequential search?

5) Consider the following three algorithms for determining whether anyone in the room has the same birthday as you. (9 pts, 3 pts each)
   - Algorithm 1: You say your birthday, and ask whether anyone in the room has the same birthday. If anyone does have the same birthday, they answer yes.
   - Algorithm 2: You tell the first person your birthday, and ask if they have the same birthday; if they say no, you tell the second person your birthday and ask whether they have the same birthday; etc, for each person in the room.
   - Algorithm 3: You only ask questions of person 1, who only asks questions of person 2, who only asks questions of person 3, etc. You tell person 1 your birthday, and ask if they have the same birthday; if they say no, you ask them to find out about person 2. Person 1 asks person 2 and tells you the answer. If it is no, you ask person 1 to find out about person 3. Person 1 asks person 2 to find out about person 3, etc.

   a) For each algorithm, what is the factor that can affect the number of questions asked (i.e., what is the problem size)?
   b) In the worst case, how many questions will be asked for each of the three algorithms?
   c) For each algorithm, say whether it is constant, linear, or quadratic in the problem size in the worst case.

6) For each of the following program fragments, what is the execution time complexity in terms of ***Big O***. *Note*, assume that all arrays are appropriately declared and sized. (24 pts, 3 pts each)

```
int summer1(int n) {
        int sum1 = 0;
        for (int i=1; i<=n; i++)
                for (j=1; j<=n; j++)
                        sum1++;
        return   sum1;
}

int summer2(int n) {
        int sum2 = 0;
        for (int i=1; i<=n; i++)
                for (j=1; j<=i; j++)
                        sum2++;
        return   sum2;
}

int summer3(int n) {
        sum3 = 0;
        for (int i=0; i<n; i++)
                for (int j=0; j<n; j++)
                         if (i == n%2)
                                 sum3++;
        return  sum3;
}
```

```
int looping(int n) {
        int output=0;
        for (int i=0; i<n; i++) {
                output ++;
        }
        for (int i=0; i<10; i++) {
                output ++;
        }
        for (int i=0; i<n; i++) {
                output ++;
        }
        return output;
}

void calSquares(int n, int squares[]) {
        for (int x=0; x<n; x++) {
                for (int y=0; y<n; y++) {
                        squares[n] = y * x;
                }
        }
}

int summer4(int n) {
        sum4 = 0;
        for (int i=0; i<n*n; i++)
                if (i%2 == 1)
                        for (int j=0; j<n*n; j++)
                                if (j == n%2)
                                        sum4++;
        return sum4;
}

int summer5(int n) {
        sum5 = 0;
        for (int i=0; i<n; i++)
                for (int j=0; j<n; j++)
                        for (int k=0; k<n; k++)
                                if (k == n%2)
                                        sum5++;
        return   sum5;
}

int summer6(int n) {
        sum6 = 0;
        for (int i=0; i<n; i++)
                for (int j=0; j<n; j++)
                        sum6++;
        for (int i=0; i<n; i++)
                sum6++;
        return   sum6;
}
```

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

(6 pts, 3 pts each)

Algorithm #1

```
long long pow( 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 pow( long long x, int n ) {
    long long  y;
    if (n == 0)
        return 1;
    if (n == 1)
        return x;
    if (n%2 == 0)
        y = pow(x, n/2);
        return y*y;
    else
        y = pow(x, n/2);
        return y*y*x;
}
```

a) What is the time complexity in terms of ***Big O*** for each algorithm?
b) Would using the better algorithm for assignment #1 have improved the running time for the brute force algorithm? Explain specifically why or why not.

8) Given the following functions, (6 pts, 3 pts each).

Algorithm #1

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

Algorithm #2

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

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

9) According to the text; (8 pts, 4 pts each)
   a) What is a ***space/time trade-off*** principal?
   b) Provide an example of a space/time trade-off.

10) With regard to the algorithms from assignment #1; (6 pts, 3 pts each)
   a) Provide a Big-Oh analysis for algorithm 1 (brute force).
   b) Provide a Big-Oh analysis 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.