GTU Department of Computer Engineering
CSE 222/505 - Spring 2022
HOMEWORK 2

Due Date: March 14, 2022 – 09:00 AM

1) For each of the following statements, specify whether it is true or not, and prove your claim. Use the definition of asymptotic notations.

a) $\log_2 n^2 + 1 = O(n)$ $2\log_2 n$ ✓

$T(n) = O(g(n))$ $T(n) \le c \cdot g(n)$ $N \ge n_0$ $c, n_0 \in Z^+$

$2\log_2 n + 1 \le 10 \cdot n$ $n=4 \wedge c=10$ ✓

b) $\sqrt{n(n+1)} = \Omega(n)$ ✗

$c \cdot f(n) \le T(n)$ $nc \le \sqrt{n^2+n}$ $c \le \sqrt{\frac{n^2+n}{n^2}}$ $n=1$ $c=\sqrt{2}$ $\sqrt{2}\, n \le \sqrt{n^2+n}$ $2n^2 \le n^2+n$ $n>1$ $n\ge 2$ false

c) $n^{n-1} = \theta(n^n)$ ✗ $c_1 \le \frac{1}{n} \le c_2$ $n>1$

we have upper bound

we don't have lower

2) Order the following functions by growth rate and explain your reasoning for each of them. Use the limit method.

$\lim_{n\to\infty} \frac{f(n)}{g(n)}$ $\log n > \sqrt{n}$ $n^3 > \log n$

$10^n > 2^n > n^3 > n^2 > n^2\log n > \sqrt{n} > \log n$

$\lim_{n\to\infty} \frac{n^2\log n}{n^3 \wedge} = \lim_{n\to\infty} \frac{1}{\ln(a) n^2} = 0$

$n^2,\ n^3,\ n^2 \log n,\ \sqrt{n},\ \log n,\ 10^n,\ 2^n,\ 8^{\log_2 n}$ n^3

3) What is the time complexity of the following programs? Use most appropriate asymptotic notation. Explain by giving details.

a)

```
int p_1 ( int my_array[]){
        for(int i=2; i<=n; i++){
                if(i%2==0){
                        count++;
                } else{
                        i=(i-1)i;
                }
        }
}
```

$T_{worst}(n) = n-2$

$O(n)$

$i^2 - i + 1 > n$

$\Omega(\sqrt{n})$

b)

```
int p_2 (int my_array[]){
        first_element = my_array[0];
        second_element = my_array[0];
        for(int i=0; i<sizeofArray; i++){
                if(my_array[i]<first_element){
                        second_element=first_element;
                        first_element=my_array[i];
                }else if(my_array[i]<second_element){
                        if(my_array[i]!= first_element){
                                second_element= my_array[i];
                        }
                }
        }
}
```

$O(1)$

$O(n)$

$\Theta(n)$

c)

```
int p_3 (int array[]) {
        return array[0] * array[2];
}
```

$O(1)$

$\Theta(1)$

d)

```
int p_4(int array[], int n) {
        Int sum = 0
        for (int i = 0; i < n; i=i+5)
                sum += array[i] * array[i];
        return sum;
}
```

$\to 1$

$\frac{n}{5}$

$\to 1$

$i = 5k$

$5k \geqslant n$

$k \geqslant \frac{n}{5}$

$T(n) = \frac{n}{5} + 2 = O(n)$

e)

```
void p_5 (int array[], int n){
        for (int i = 0; i < n; i++)
                for (int j = 1; j < i; j=j*2)
                        printf("%d", array[i] * array[j]);
}
```

$\log i$ times

$j = 2^k$

$2^k \geqslant i$

$k \geqslant \log i$

$\sum_{i=0}^{n} \log i = \log(n!)$ $\quad O(\log(n!))$

f)

```
int p_6(int array[], int n) {
        If (p_4(array, n)) > 1000)
                p_5(array, n)
        else printf("%d", p_3(array) * p_4(array, n))
}
```

$\to n$, $\to \log n!$ $\Big\} O(n)$

$\underbrace{p\_3(array)}_{1} * \underbrace{p\_4(array, n)}_{n}$ $\Big\} O(n)$

$\lim_{n\to\infty} \frac{\log n!}{n} = \lim_{n\to\infty} \left(\overset{0}{\cancel{\frac{\log n}{n}}} + \overset{0}{\cancel{\frac{\log(n-1)}{n}}} + \dots \right) = 0$

$n > \log n!$

$O(n)$

g)

```
int  p_7( int n ){
        int i = n;
        while (i > 0) {
                for (int j = 0; j < n; j++)
                        System.out.println("*");
                i = i / 2;
        }
}
```

$T_{best}(n) = O(1)$ if n less than zero

$T_{worst}(n) = O(\log n)$

$\frac{n}{2^k} \leq 1$ $\quad \log(n)$

h)

```
int  p_8( int n ){
        while (n > 0) {
                for (int j = 0; j < n; j++)
                        System.out.println("*");
                n = n / 2;
        }
}
```

$\log n$

$\log n + \log \frac{n}{2} + \dots + \log 1$

$= \log \frac{n^n}{2^k} = O(\log n^n) = O(n \log n)$

constant

i)

$T_{best}(n) = \Theta(1)$

$T_{worst}(n) = \Theta(n\log n)$

```
int p_9(n){
        if (n = 0)
                return 1
        else
                return n * p_9(n-1)
}
```

$\rightarrow \log n^n$

j)

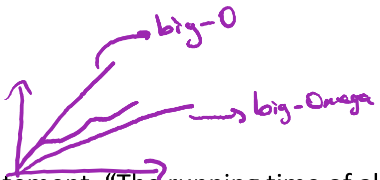
$T_{best}(n) = O(1)$

```
int p_10 (int A[ ], int n) {
      if (n == 1)
            return;
      p_10 (A, n − 1);
      j = n − 1;
      while (j > 0 and A[j] < A[j − 1]) {
            SWAP(A[j], A[j − 1]);
            j = j − 1;
      }
}
```

$\rightarrow n-1+n-2+\ldots+1 \rightarrow T_{worst}(n) = \frac{(n-1)(n)}{2} = O(n^2)$

$O(n-1)$

$\Omega(1)$



4)

a) Explain what is wrong with the following statement. "The running time of algorithm A is at least $O(n^2)$". big-O notation denotes the upper bound time complexity, not lower bound. But we can say the run time is at most $O(n^2)$

b) Prove that clause true or false? Use the definition of asymptotic notations.

I. $2^{n+1} = \Theta(2n)$ $\quad 2n \cdot c_1 \le 2^{n+1} \le 2n \cdot c_2$ false

II. $2^{2n} = \Theta(2n)$ $\quad 2n \cdot c_1 \le 2^{2n} \le 2n \cdot c_2$ false

III. Let $f(n)=O(n^2)$ and $g(n)=\Theta(n^2)$. Prove or disprove that: $f(n) * g(n) = \Theta(n^4)$

$O(n^2) * \Theta(n^2) = O(n^2)$ $\quad$ Θ denotes upper and lower band

maybe $\Theta(\log n)$??

5) Solve the following recurrence relations. Express the result in most appropriate asymptotic notation. Show details of your work.

a) $T(n) = 2T(n/2) + n$, $T(1) = 1$

$2(2^{k-1}(2^k \times T(n/2^k) + n/2^{k-1}) + \ldots)$

$2^k \times T(n/2^k) + k.n = n + 1 + n\log n = \Theta(n\log n)$

$\frac{n}{2^k} = 1 \quad k = \log_2 n$

b) $T(n) = 2T(n-1) + 1$, $T(0)=0$ $\quad T(n) = 2^k T(n-k) + 2^k - 1$

$2(2(2.T(n-3)+1)+1)+1$

$2^3.T(n-3)+1+2+4$

$2^k - 1$

$n=k$

$T(n) = 2^n - 1 = O(2^n)$

6) In an array of numbers (positive or negative), find pairs of numbers with the given sum. Design an iterative algorithm for the problem. Test the algorithm with different size arrays and record the running time. Calculate the resulting time complexity. Compare and interpret the test result with your theoretical result.

7) Write a recursive algorithm for the problem in 6 and calculate its time complexity. Write a recurrence relation and solve it.