**CSCI 2110**
**Data Structures and Algorithms**
**Practice Set for Test No. 1- SOLUTIONS**

1. Derive the big O complexities for each of the following running times:
   a. $25n^4 + 45n - 256$

**→ $25n^4 + 45n - 1$ → $25n^4$ → $n^4$ → $O(n^4)$**

   b. $n^2 + 3000n + 500\ n^2\log n + 1000$

**→ $n^2 + 3000n + 500\ n^2\log n + 1$ → $500\ n^2\log n$ → $n^2\log n$ → $O(n^2\log n)$**

   c. $(3\log n + 5)(n^3 + 2)$

**= $3n^3\log n + 5n^3 + 6\log n + 10$**
**→ $3n^3\log n + 5n^3 + 6\log n + 1$ → $3n^3\log n$ → $n^3\log n$ → $O(n^3\log n)$**

   d. $(n^3 + 2n^2 + 3)/n^2$

**= $n^3/n^2 + 2n^2/n^2 + 3/n^2$**
**= $n + 2 + 3/n^2$ → $n + 1 + 3/n^2$ → $n$ → $O(n)$**

   e. $(1+2+3+\ldots+n)(1+2+3+\ldots+n)$

**= $n(n+1)/2$ * $n(n+1)/2$ = $n^4/4$ + $n^3/2$ + $n^2/4$ → $n^4/4$ → $n^4$ → $O(n^4)$**

   f. $(1^2+2^2+3^2+\ldots.+n^2) / (1 + 3 + 5 + \ldots + 2n-1)$

**= $n(n+1)(2n+1)/6$ * $1/n^2$ = $(n+1)(2n+1)/6n$ = $(2n^2+3n+1)/6n$ = $n/3 + ½ + 1/6n$ → $O(n)$**

   g. $\log n^3 + n \log n^4 + n \log n^2$

**= $3\log n + 4n\log n + 2n\log n$ = $3\log n + 6n\log n$ → $6n\log n$ → $n\log n$ → $O(n\log n)$**

2. Arrange the following growth functions in *increasing order* of growth rates. (Reduce the growth functions wherever necessary).

   a. The growth functions that you derived in Question 1 above.

   **$n < n \log n < n^2 \log n < n^3 \log n < n^4$**

   b. $10000n, 100n^2, \sqrt{n}, 10n^3, 2^n, \sqrt{n} * \log n, n\log n, n\log n^n$

   $10000n \rightarrow O(n)$; $100n^2 \rightarrow O(n^2)$;
   $10n^3 \rightarrow O(n^3)$; $n\log n^n = n*n\log n = n^2\log n \rightarrow O(n^2\log n)$

   **$\sqrt{n} < \sqrt{n} * \log n < n < n\log n < n^2 < n^2\log n < n^3 < 2^n$**

3. a) What does it mean for a function to be O(1)?

   b) Which has the larger time complexity – O(1) or O(1/n)? Why?

**O(1) complexity means that the number of basic operations is a constant, irrespective of the size of the input.**

**O(1) is larger than O(1/n) since 1/n asymptotically decreases as n increases.**

4. An algorithm with complexity $O(n^2)$ takes 5 ms to process 50 data items.

   a) Estimate how long it will take to process 5000 data items.

   **50 X 50 is proportional to 5ms**
   **5000 X 5000 is proportional to *x* ms**

   ***x* = (5 X 5000 X 5000)/ (50 X 50) = 50,000 ms**

   **It will take 50,000 ms to process 5000 data items.**

   b) Estimate how much data can be processed in 500 ms.

   **5 ms is proportional to 50 X 50**
   **500 ms is proportional to *x* X *x***

   ***x* X *x* = (50 X 50 X 500) / 5 = 250000**

   **Therefore, *x* = 500**
   **500 data items can be processed in 500 ms.**

5. An algorithm with complexity $O(n\log_2 n)$ takes 1 ms to process 1024 data items.

a) Estimate how long it will take to process 512 data items.

**1024 log 1024 → 1 ms**
**512 log 512 → *x* ms**

***x* = (1 X 512 log 512)/(1024 log 1024) = (512 X 9) / (1024 X 10) = 0.5 X 0.9 = 0.45**

**512 data items can be processed in 0.45 ms.**

b) Estimate how long it will take to process 1024*1024 data items.

**1024 log 1024 → 1 ms**
**(1024 X 1024) log (1024 X 1024) → *x* ms**

***x* = (1 X (1024 X 1024) log(1024 X 1024)) / (1024 log 1024)**
**= (1024 X 1024 X 20) / (1024 X 10)**
**= (1024 X 2)**
**= 2048**

**1024 * 1024 data items can be processed in 2048 ms.**

6. Three software packages A, B and C are being used to determine DNA patterns. Software package A takes exactly $T_A = K_A * n + 100$ seconds, B takes exactly $T_B = K_B * n + \log_2 n$ seconds and C takes $T_C = K_C * n^2$ seconds to finish the task for a problem of size n.
During an experiment, it was observed that A takes 1700 seconds, B takes 100 seconds, and C takes 256 seconds to complete to process n = 16 data items.

**$T_A = K_A * n + 100$**
**$1700 = K_A * 16 + 100$**
**$K_A = 1600/16 = 100$**

**$T_B = K_B * n + \log_2 n$**
**$100 = K_B * 16 + \log_2 16$**
**$100 = K_B * 16 + 4$**
**$KB = 96/16 = 6$**

**$T_C = K_C * n^2$**
**$256 = K_C * 16 * 16$**
**$K_C = 256/ 256 = 1$**

- Which package would be the best for processing n = 1024 data items?
  **For n=1024,**
  **$T_A = K_A * n + 100 = 100 X 1024 +100 = 102500$ seconds**
  **$T_B = K_B * n + \log_2 n = 6 X 1024 + \log_2 1024 = 6 X 1024 + 10 = 6154$ seconds**
  **$T_C = K_C * n^2 = 1 X 1024 X 1024 = 1048576$ seconds**

  **Package B is the best.**
- Which package would be the worst for the same?
  **Package C would be the worst.**

- Which two packages would have the same Big O complexity?
  **Packages A and B have the same Big O complexity of O(n)**

7. Give one example each of an algorithm or operation that has the following time complexity

| Complexity | Algorithm / Operation |
|---|---|
| O(1) | **Adding a node to the front of a linked list.** |
| O(n) | **Traversing a linked list of n nodes.** |
| $O(n^2)$ | **Bubble Sort/Polynomial Evaluation** |
| O(log n) | **Binary search of a sorted list.** |
| $O(n^3)$ | **Multiplication of 2 n X n matrices.** |
| $O(2^n)$ | **Brute force hacking of an n-bit binary number.** |

8. Derive the number of iterations for each of the following code segments, the big O complexity of each and the overall complexity of the full code.

```
for (int i = 1; i <= n; i++)
      sum++;
```

→ n iterations → O(n)

```
for (int i = 1; i <= n; i++)
      for (int j = 1; j <=n; j++)
            sum++;
```

→ $n^2$ iterations → $O(n^2)$

```
if (x==10)                          → 1 operation

      for (int i = 1; i <= n; i++)    → n iterations
            sum++;
else
{                                                  → n^3 iterations
      for(int i=1;i<=n;i++)
            for(int j=1; j<=n;j++)
                  for(int k=1;k<=n;k++)
                        sum++;
}
```

Complexity = 1 + max(n, $n^3$)  = 1 + $n^3$ = $O(n^3)$

```
for (int j = 1; j < n; j=2*j)
            sum++;
```

| n | number of iterations |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 4 | 2 |
| 8 | 3 |
| 16 | 4 |
| … | … |
| n | $\log_2 n$ |

Complexity is $O(\log_2 n)$

```
for (int i = 1; i <= n; i++)
      for (int j = 1; j <= i; j++)
                  sum++;
```

| n | i | j | total number of iterations (count the number of j's) |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 2 | 1 | 1 | |
| | 2 | 1,2 | 3 |
| 3 | 1 | 1 | |
| | 2 | 1,2 | |
| | 3 | 1,2,3 | 6 |
| 4 | 1 | 1 | |
| | 2 | 1,2 | |
| | 3 | 1,2,3 | |
| | 4 | 1,2,3,4 | 10 |

See the pattern

In general, for n , the number of iterations is $1 + 2 + 3 + 4 + \ldots + n = n(n+1)/2 \rightarrow O(n^2)$

Overall complexity is $O(n^3)$

**9.** There is a highway with n exits numbered 1 to n. You are given a list of the distances between them. For instance:

| Exits: | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| Distances: | 0 | 5 | 3 | 4 | 2 | 8 | 7 |

The distance from exit0 to exit1 is 5, from exit1 to exit2 is 3, from exit2 to exit3 is 4, and so on.

Design an algorithm (in pseudocode) that would calculate the distance between a given pair of exits:

For example, Distance(exit4, exit6) is 8+7 = 15
As a second example, Distance(exit0, exit5) is 5+3+4+2+8= 22

Remember, n can be any number. You may use any appropriate storage structure.
Note: A pseudocode is just an outline of the program written in a form that can be easily converted to a program. Syntax is not important in a pseudocode.

What is the worst case complexity of your algorithm?

What is the best case complexity of your algorithm?

What is the average case complexity of your algorithm?

**Solution:**

**Assume that the distances are stored in an array of integers, dist, of size n.**

**Algorithm findDistance (i, j )**

**Input: i, j, dist**

```
Sum ← 0
For x: from i+1 to j
  Sum ← Sum + dist [x]
```

**Worst case complexity: i is 0, j is n → O(n)**

**Best case complexity: i is 0, j is 1 → O(1)**

**Average case complexity: i is 0, j is n/2 → O(n)**

10. Two people compare their favourite playlists for matching songs. Both playlists have n songs each. Each playlist is stored in an array, in no particular order.

Design an algorithm in pseudocode to find the common songs in these lists. The algorithm should search the two playlists and put all the common songs into a third array.

What is the worst-case complexity of the algorithm?

**Solution:**

**Input: playlist1, playlist2**

**Initialize an empty array playlist3 of size n**
**For x: from 1 to n**
**{**
**song ← playlist1[x]**
**For y: from 1 to n or until found**
**{**
**check ←playlist2 [y]**
**if check == song then found**
**}**
**if found then put check in playlist3.**

**}**

**Order of complexity is $O(n^2)$**

11. The Node<T> class has been defined and is given on the Reference sheet.
The LinkedList<T> class has the following structure.

```
public class LinkedList<T>
{
      private Node<T> front;
      private int count;

      public LinkedList()
      {
            front = null;
            count=0;
      }
```

Add the following methods to the above code

a. Add a given item to the end of the list.
b. Print the data contained in every second item in the linked list.
c. Remove the last node from the linked list (if it exists)
d. Remove the second last node from the linked list (if it exists).

```
public void addToEnd(T item)
{
	Node<T> curr = front;
	while (curr.getNext()!=null)
		curr = curr.getNext();
	Node<T> newN = new Node<T>(item, curr.getNext());
	curr.setNext(newN);
	count++;
}


public void printEven()
{
	Node<T> curr = front;
	int index=0;
	while(curr!=null)
	{
		if (index%2!=0)
			System.out.println(curr.getData());
		index++;
		curr=curr.getNext();
	}
}

public void removeLast()
{
	if (count==0) return;
	if (count==1) { front = null; count=0; return;}
	Node<T> curr = front;
	while(curr.getNext()!=null)
		curr=curr.getNext();
	curr.setNext(null);
	count--;
}

public void removeSecondLast()
{
	if (count==0||count==1) return;
	if (count==2){ front=front.getNext(); count--; return;}
	Node<T> curr = front;
	while (curr.getNext().getNext()!=null)
		curr = curr.getNext();
	curr.setNext(curr.getNext().getNext());
	count--;
}
```

12. Short Snappers

1. The class in java from which all classes directly or indirectly inherit from is the _____**Object**_____ class.

2. The operator that can be used to determine whether a reference variable references an object of a particular class is the _____**instance of**_____ operator.

3. If ClassB extends ClassA, and ClassC is a superclass of ClassA, and the following declaration has been made:

```
ClassA ref;
```

which of the following are valid?

```
ref = new ClassA();  → valid
ref = new ClassB();  → valid
ref = new ClassC();  → invalid
```

4. ClassB extends ClassA, and ClassC extends ClassB, and ClassD extends ClassC. The following is a part of the code in a program:

ClassB ref;
ref = new ClassC();

What will each of the following statements return?

if (ref instanceof ClassA) → True

if (ref instanceof ClassB) → True

if (ref instanceof ClassC) → True

if (ref instanceof ClassD) → False

13. Write a class called Point that has two attributes (xpos and ypos), a constructor that sets xpos and ypos, set methods for xpos and ypos, get methods for xpos and ypos, and a toString method that returns the values.

The class must be written as a generic class, which means that it should work for any type of object, say Integer, or Double, or String.

```
public class Point<T>
{
        private T xpos;
        private T ypos;

        public Point(T x, T y)
        {
```

```
            xpos = x;
            ypos = y;
      }
      public void setX(T x)
      {
            xpos = x;
      }
      public void setY(T y)
      {
            ypos = y;
      }
      public T getX()
      {
            return xpos;
      }
      public T getY()
      {
            return ypos;
      }
      public String toString()
      {
            return "XPOS: " + xpos + "\tYPOS: " + ypos + "\n";
      }

}
```

14. You are given the methods in the Unordered List class on the reference sheet. ***Using these methods***, implement the following method. You may declare any necessary variables inside the methods.

```
//create a new list that has list1 concatenated with
// list2.Order doesn't matter
public List concatenate(List<T> list1, List<T> list2)
{
      List<T> list3 = new List<T>();
      T item = list1.first();
      while (item!=null)
      {
            list3.add(item);
            item = list1.next();
      }
      item = list2.first();
      while (item!=null)
      {
            list3.add(item);
            item=list2.next();
      }
      return list3;
}
```

15. An application program class PokemonList uses the Unordered List class. Another class Pokemon defines the Pokemon object items that will be used by the PokemonList class. The first few lines of the program are given. Some lines are missing or some parameters in each line may be missing. Write the missing pieces.

public class Pokemon {….} //defines the Pokemon object. This class is complete

```
public class PokemonList
{
        //missing line – define an unordered list of Pokemon objects called pokelist
        private List<Pokemon> pokelist;

        public PokemonList()
        {
                //missing line – construct an empty pokelist
                pokelist = new List<Pokemon>();
        }
        public void add (Pokemon p//missing parameter) //adds one Pokemon object to
                                                                //the list
        {
                //missing line
                pokelist.add(p);
        }
}
```

**REFERENCE SHEET**

***Useful properties of logarithms:***

$\log_b(xy) = \log_b x + \log_b y$

$\log_b (x/y) = \log_b x - \log_b y$

$\log_b x^a = a\log_b x$

$\log_x a/\log_x b = \log_b a$

***Useful properties of exponentials:***

$a^{(b+c)} = a^b a^c$

$a^{bc} = (a^b)^c$

$a^b / a^c = a^{(b-c)}$

***Useful series:***

Sum of first n integers: $1+2+3 + \ldots + n = n(n+1)/2$

Sum of first n odd positive integers: $1 + 3 + 5 + \ldots (2n-1) = n^2$

Sum of squares of first n integers: $1^2 + 2^2 + 3^2 + \ldots. + n^2 = n*(n+1)*(2n+1)/6$