| Data Structures and Algorithms Laboratory | | |
|---|---|---|
| **Laboratory 4:** Circular Linked Lists and Algorithm Analysis | **School of Information Technology** | |
| **Name:Sai Hae Naing Lay** | **ID:6531501208** | **Section:4** |
| **Date: 17 Feb 2023** | **Due date: on the LMS** | |

**Objective**

- To analyze algorithms based on experimental methods
- To implement circular linked lists

**Exercise 1:** (In-class) Use experimental analysis to compare the following two algorithms. Fill in the times from your experiments and plot the graphs of data size (n) versus time (ms).

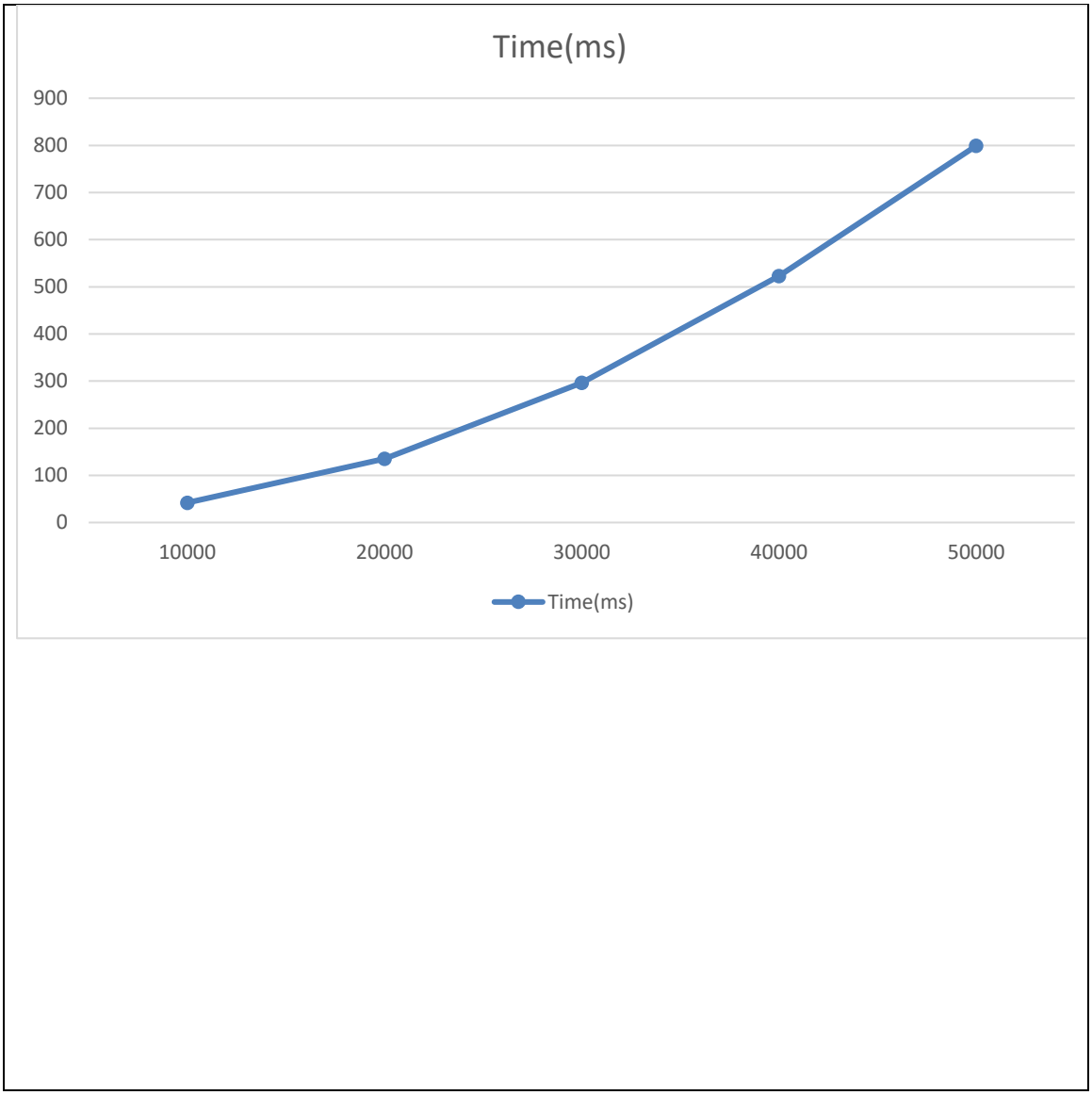
**Algorithm 1**

Assume that the data size (n) is from 10000 to 50000

```
for(int i=1;i<=n;i++) {
      for(int j=1;j<=n;j++) {
       result = i+j;
      }
}
```

| n | 10000 | 20000 | 30000 | 40000 | 50000 |
|---|---|---|---|---|---|
| Time (ms) | 42 | 135 | 296 | 523 | 799 |

Capture a graph image and paste here.



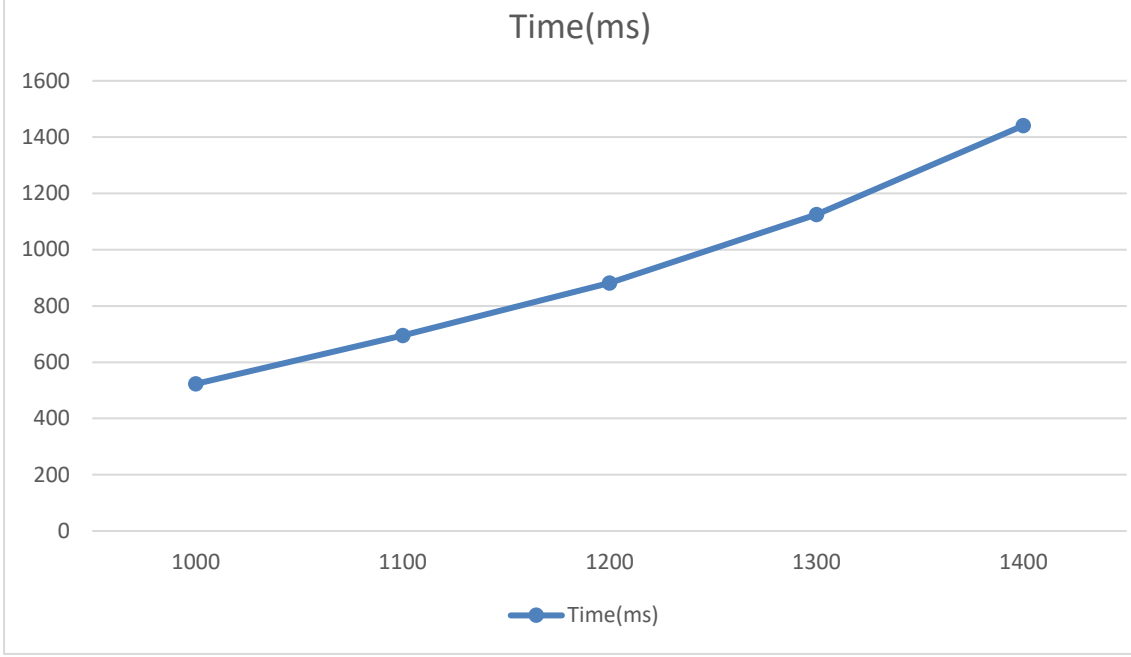

**Algorithm 2**

Assume that the data size (n) is from 1000 to 1400

```
for(int i=1;i<=n;i++) {
      for(int j=1;j<=n;j++) {
            for(int k=1;k<=n;k++) {
              result = i+j+k;
            }
      }
}
```

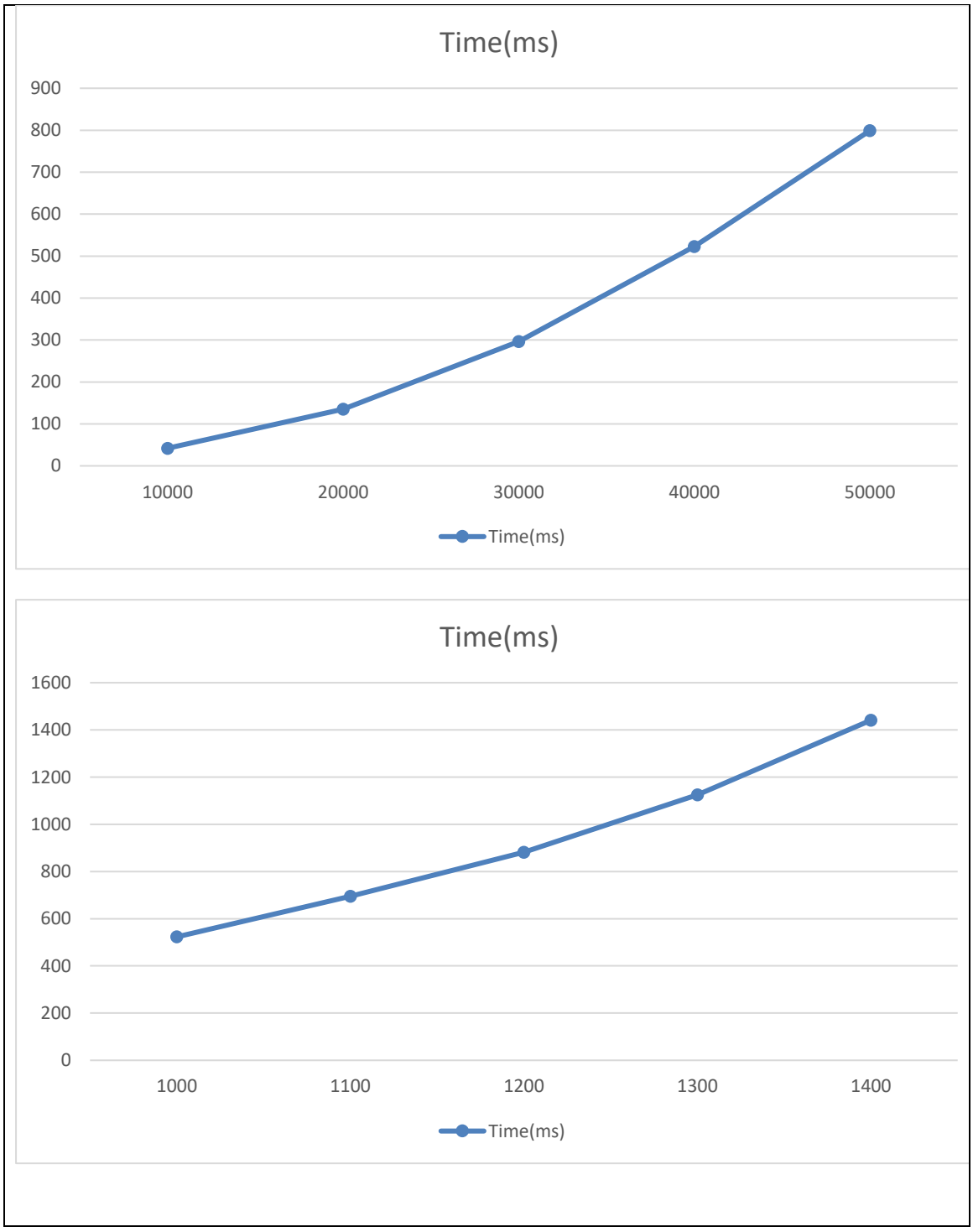
| n | 1000 | 1100 | 1200 | 1300 | 1400 |
|---|---|---|---|---|---|
| Time (ms) | 523 | 695 | 882 | 1125 | 1441 |

Capture a graph image and paste here.




**Which algorithm is faster? Explain.**

The algorithm 1 is faster than algorithm 2.Because in algorithm 1,the program need to process the 2 for loops and algorithm 2 needs to process the 3 for loops.


Time(ms)
900
800
700
600
500
400
300
200
100
0
10000
20000
30000
40000
50000
Time(ms)
Time(ms)
1600
1400
1200
1000
800
600
400
200
0
1000
1100
1200
1300
1400
Time(ms)

**Exercise 2:** (In-class) Write a program to count equal numbers in array A, B and C.

For example,
A = {1, 2, 3}
*B* = {2, 3, 4}
*C* = {3, 4, 5}

Here there is only one number that is the same in these three sets which are a number 3. So count is 1.

In general,
A = {1,2,3,…,n}
B = {2,3,4,…,n+1}
C = {3,4,5,…,n+2}

Fill in the missing codes.

```
public class CountSameNumber {

       public static void main(String[] args) {
           //change value n and run the program
           int n=500;

              //three arrays
           int[] a = new int[n];
           int[] b = new int[n];
           int[] c = new int[n];
           for(int i=0;i<n;i++){
             a[i]= i+1;
             b[i]= i+2;
             c[i]= i+3;
           }

           //algorithm
           //----------------------------------------
           long startTime = System.currentTimeMillis();

           //add your algorithm, here

          for(int i = 0;i < n; i++) {

for(int j = 0; j < n; j++) {

for(int k = 0; k < n; k++) {

if((a[i] == b[j]) && (b[j] == c[k])) {

count++;

}
```

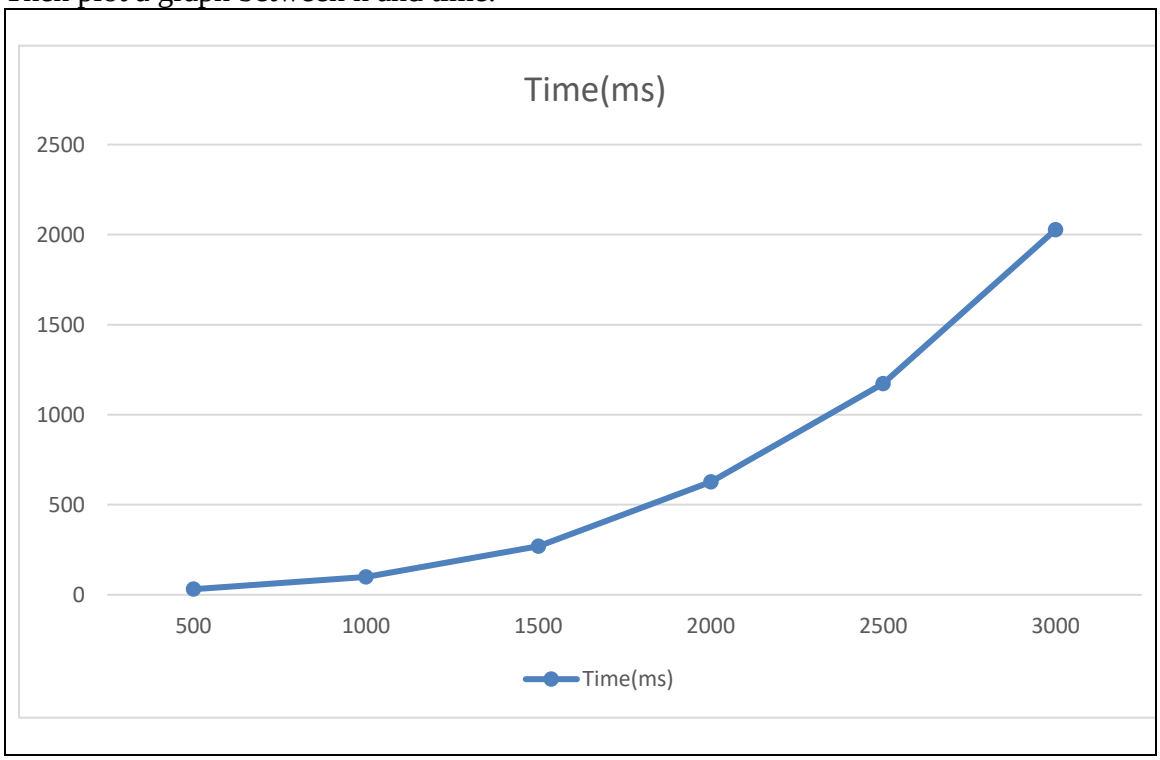
```
}

}

}

            long endTime = System.currentTimeMillis();

            //----------------------------------------

            System.out.println("Count = "+count);
            System.out.println("Elapsed time = "+(endTime-startTime));
        }
}
```

If n =500, the output will be
Count = 498
Elapsed time = 203*

**The elapsed time will be different for each computer!!!*
Try to change n from 500, 1000, 1500… 3000 and fill in the table below. Note that if your computer is too fast, start n with a larger number i.e. 10000, 20000… 60000 instead.

| n | 500 | 1000 | 1500 | 2000 | 2500 | 3000 |
|---|---|---|---|---|---|---|
| time (ms) | 31 | 100 | 269 | 627 | 1173 | 2027 |

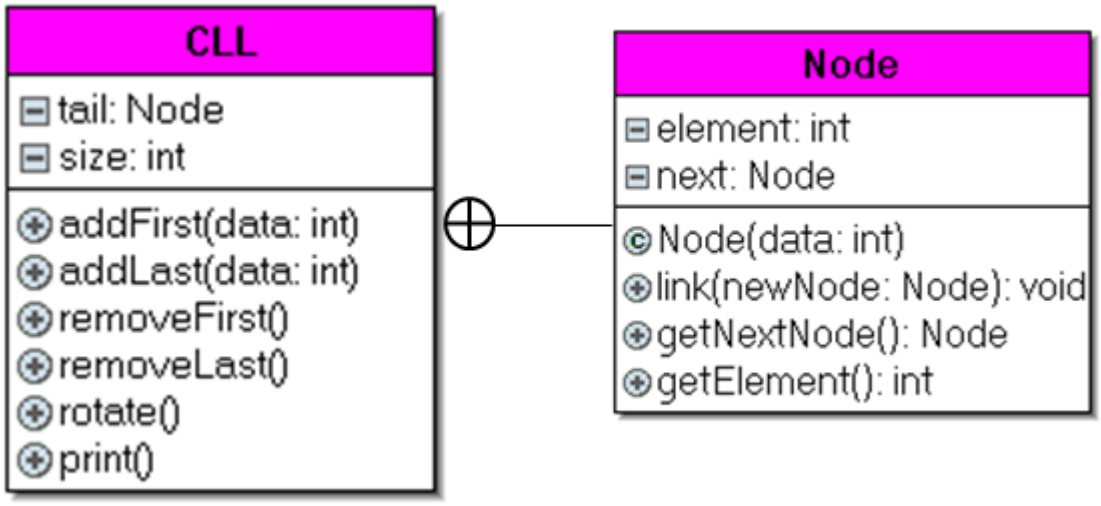
Then plot a graph between n and time.



What is Big-O of your algorithm?

Big-O of algorithm is n^3.

**Exercise 3:** (Homework) From the given class diagram, create a circular linked list and complete the program to get the results as shown.




Expected result

```
Empty linked list
->1->
->1->2->
->3->1->2->
->1->2->3->
->2->3->
->2->
```

```
//================== CLL class ========================

package DSALab04;

//================== CLL class ========================

class CLL {

// ---------------------- Node --------------------

private class Node {

private int element;

private Node next;

// constructor

public Node(int data) {
```

```
element = data;

next = null;

}

// link a new node to this node

public void link(Node newNode) {

next = newNode;

}

// return next node

public Node getNextNode() {

return next;

}

// return element of this node

public int getElement() {

return element;

}

}

// ---------------------- End Node --------------------

// CLL properties and methods

private Node tail = null;

private int size = 0; // SLL's size

public void addFirst(int data) {

Node n = new Node(data);

if(size == 0) {

tail = n;

tail.link(n);
```

```
size++;

}

else{

n.link(tail.getNextNode());

tail.link(n);

size++;

}

}

public void addLast(int data) {

Node n = new Node (data);

if (size == 0) {

tail = n;

tail.link(n);

size++;

}

else{

n.link(tail.getNextNode());

tail.link(n);

tail = n;

size++;

}

}

public void removeFirst() {

if (size == 1) {

tail = null;
```

```
size--;

}

else {

Node p = tail.getNextNode();

tail.link(p.getNextNode());

p = null;

size--;

}

}

public void removeLast() {

if(size == 1) {

tail = null;

size--;

}

else {

Node p = tail.getNextNode();

while(p.getNextNode() != tail) {

p = p.getNextNode();

}

p.link(tail.getNextNode());

tail = null;

tail = p;

size--;

}

}
```

```
public void rotate() {

if(size > 1) {

tail = tail.getNextNode();

}

}

public void print() {

if(size == 0) {

System.out.println("Empty link list");

}

else{

Node p = tail.getNextNode();

do {

System.out.print("->" + p.getElement());

p = p.getNextNode();

}while(p != tail.getNextNode());

System.out.println("->");

}

}

}

public class MainCLL {

public static void main(String[] args) {

// TODO Auto-generated method stub

CLL cll = new CLL();

cll.print();

cll.addFirst(1);
```

```
cll.print();

cll.addLast(2);

cll.print();

cll.addFirst(3);

cll.print();

cll.rotate();

cll.print();

cll.removeFirst();

cll.print();

cll.removeLast();

cll.print();

}

}
```