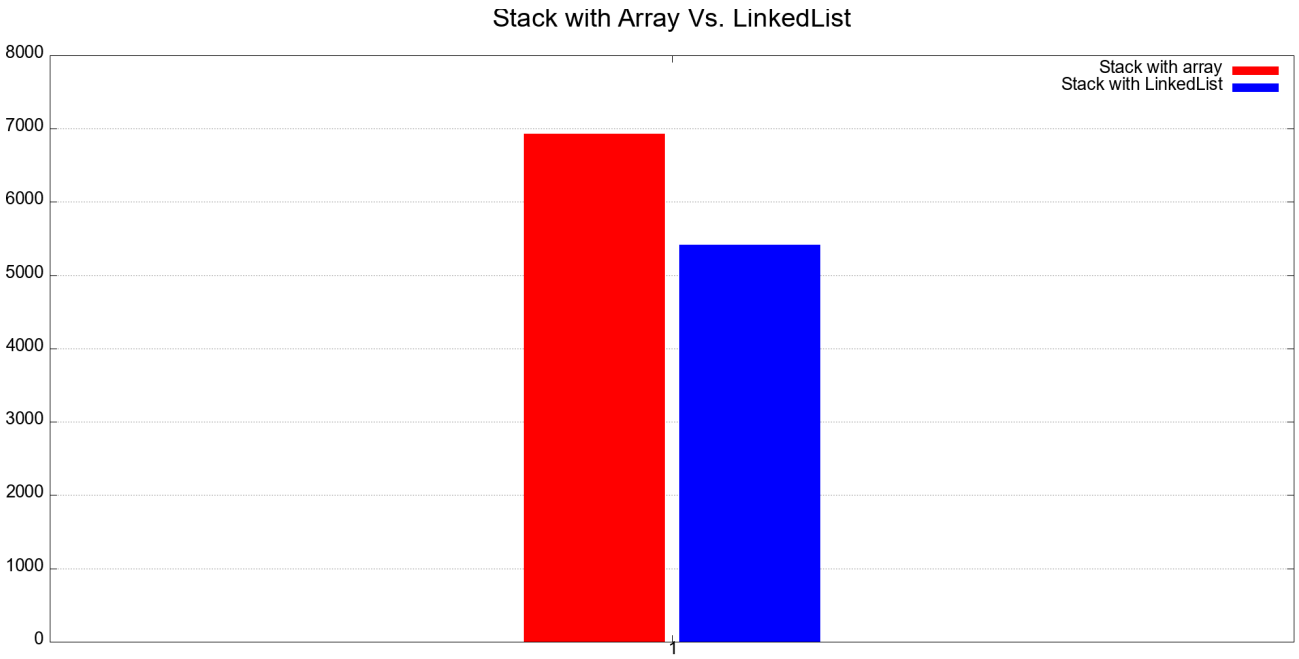
Lab4 (Stacks)

Graph



Y axis -> Time (nano seconds)

X axis -> Test case

Code in VSCode for plotting the graph

```
#include <iostream>

using namespace std;

int main() {

  FILE *gnuplotPipe = popen("gnuplot -persistent", "w");

  fprintf(gnuplotPipe, "set terminal png size 2000,1000\n");

  fprintf(gnuplotPipe, "set output 'time taken plot.png'\n");

  fprintf(gnuplotPipe, "set yrange[0:8000:1000]");
```

```
  fprintf(gnuplotPipe, "set mytics 2\n");

  fprintf(gnuplotPipe, "set style data histogram\n");

  fprintf(gnuplotPipe, "set style histrogram cluster gap 1\n");

  fprintf(gnuplotPipe, "set style fill solid\n");

  fprintf(gnuplotPipe, "set boxwidth 0.9\n");

  fprintf(gnuplotPipe, "set xtrics format \"\"\n");

  fprintf(gnuplotPipe, "set grid ytics\n");

  fprintf(gnuplotPipe, "set title 'Stack with Array Vs. LinkedList'\n");

  fprintf(gnuplotPipe, "set title font \",30\"\n");

  fprintf(gnuplotPipe, "set ytics font \",20\"\n");

  fprintf(gnuplotPipe, "set xtics font \",20\"\n");

  fprintf(gnuplotPipe, "set key font \",20\"\n");

  fprintf(gnuplotPipe,"plot 'data.txt' using 2:xtic(1) title 'Stack with array'
linecolor \"red\",'data.txt' using 3 title 'Stack with LinkedList' linecolor
\"blue\"");

  fflush(gnuplotPipe);

  return 0;

}
```

Data

17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8

1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8

Stack with Array
#########
6934.80000000000018189894

---

17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8
17 20 18 6 23 15 11 5 10 8
15 11 5 10 8
1 3 30 4 15 11 5 10 8

Stack with LinkedList
#########
5412.19999999999981810106

Code for getting data from [https://www.tutorialspoint.com/compile_cpp_online.php]

```cpp
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
//Add your programs and other functions here.

class StackA {
    // <--- ADD YOUR CODE HERE --->
    public:
    int *arr;
    int n;
    int top;
    StackA(int max_size){
        arr=new int[max_size];
        n=max_size;
        top=-1;
    }
    void push(int val){
        top++;
        if(top>n-1){
            std::cout<<"Stack Overflow"<<std::endl;
            top--;
            return;
        }
        arr[top]=val;
    }
    int pop(){
        if(isEmpty()){
            std::cout<<"Stack Underflow"<<std::endl;
            return -1;
        }
        int data=arr[top];
        top--;
        return data;
    }
    bool isEmpty(){
        if(top==-1){
            return true;
        }
        return false;
    }
    bool isFull(){
        if(top==n-1){
            return true;
        }
        return false;
```

```
    }
    int stackTop(){
        return arr[top];
    }
    void display(){
        int temp_top=top;//keep top safely in another variable
        for(int i=top;i>=0;i--){
            std::cout<<arr[i]<<" ";
        }
        std::cout<<std::endl;
        top=temp_top;//restore top
    }
};

class Node {
    // <--- ADD YOUR CODE HERE --->
    public:
        Node *next;
        int data;
        Node(int val){
            data=val;
            next=nullptr;
        }

};

class StackLL {
    // <--- ADD YOUR CODE HERE --->
    public:
        Node *top;
        Node *head;
        StackLL(){
            top=nullptr;
            head=nullptr;
        }
        void push(int val){
                Node *temp=new Node(val);
            if(isEmpty()){//new Stack;(head=top);(first node)
                top=temp;
                head=temp;
                return;
            }
            //add a new node for existing stack
            top->next=temp;
            top=temp;
        }
```

```
        int pop(){
            if(isEmpty()){//no elements
                std::cout<<"Stack Underflow";
                return -1;
            }
            int d=top->data;
            if(head==top){//only one node
                top=nullptr;
                head=nullptr;
                return d;
            }
            //more than one node
            Node *current=head;
            while(current->next!=top){
                current=current->next;
            }
            current->next=nullptr;
            top=current;
            return d;
        }
        bool isEmpty(){
            if(top==nullptr){
                return true;
            }
            return false;
        }
        int stackTop(){
            return top->data;
        }
        void display(){
            StackLL *temp=new StackLL();//to store the stack in the same order
            Node *current=head;
            while(current->next!=nullptr){
                temp->push(current->data);
                current=current->next;
            }
            temp->push(current->data);//top element
            while(temp->head!=temp->top){
                std::cout<<temp->pop()<<" ";
            }
            std::cout<<temp->pop()<<std::endl;
        }

};
void runtheProgramStackArray(){
    StackA *stk=new StackA(11);
```

```
    stk->push(8);
    stk->push(10);
    stk->push(5);
    stk->push(11);
    stk->push(15);
    stk->push(23);
    stk->push(6);
    stk->push(18);
    stk->push(20);
    stk->push(17);
    stk->display();
    for(int i=0;i<5;i++){
        stk->pop();
    }
    stk->display();
    stk->push(4);
    stk->push(30);
    stk->push(3);
    stk->push(1);
    stk->display();
}
void runtheProgramStackLinkedList(){
    StackLL *stk=new StackLL();
    stk->push(8);
    stk->push(10);
    stk->push(5);
    stk->push(11);
    stk->push(15);
    stk->push(23);
    stk->push(6);
    stk->push(18);
    stk->push(20);
    stk->push(17);
    stk->display();
    for(int i=0;i<5;i++){
        stk->pop();
    }
    stk->display();
    stk->push(4);
    stk->push(30);
    stk->push(3);
    stk->push(1);
    stk->display();
}

int main() {
```

```
    //Get the values

    double sum_duration;
    vector<double> avg_duration;
    string topic;
    for(int programs=0;programs<2;programs++){  // change the number of algorithms
(programs < (number of algorithms))
        avg_duration.clear();
            sum_duration=0.0f;

            for(int i=0;i<5;i++){//5 times

                auto start = chrono::high_resolution_clock::now();


                switch(programs){
                    case 0:
                    runtheProgramStackArray();
                    topic="\n\n\nStack with Array\n#########\n";
                    break;
                    case 1:
                    runtheProgramStackLinkedList();
                    topic="\n\n\nStack with LinkedList\n#########\n";
                    break;
                    default:
                    break;
                }
                auto end = chrono::high_resolution_clock::now();

                // Calculating total time taken by the program.
                double time_taken =
                chrono::duration_cast<chrono::nanoseconds>(end - start).count();


                sum_duration=sum_duration+time_taken;




        }
        avg_duration.push_back(sum_duration/5.0f);
        cout<<topic;
        for(int i=0;i<avg_duration.size();i++){
            printf("%.20f\n\n\n\n\n----------------------------
\n",avg_duration[i]);
        }
```

```
    }


    return 0;
}
```

Reasons

- Linked list uses pointers for their mechanism while array uses an index
- Since that, the Linked list doesn't need to check for contiguous memory locations for insertions and it stores value anywhere and links it by a pointer, while the pointers help to increase the access speed to a memory location rather than copying a whole huge object. (Insertion and deletion are faster than arrays (just have to add/change the pointer))
- But in arrays, even if they can use pointers to their indices, they have to maintain their size fixed and check for contiguous memory locations during the construction. And they have to keep track on their size.