1. **Queue**

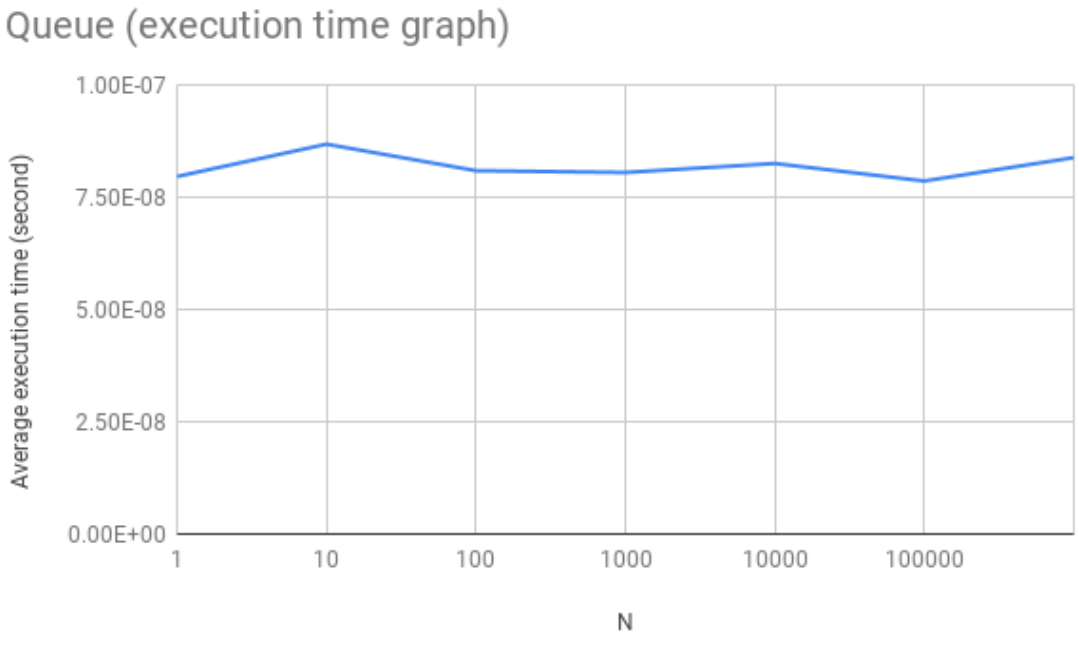
Queue is a linear data structure that follows the order of First In First Out (FIFO). When we input (enQueue) a data, the data will go at the end of the queue. When we delete (deQueue) a data, the data at the beginning of the queue will go first.

Inserting (enQueue) into a queue can happen at the back of the queue. Since we know the end of the queue already, the insertion run time will be O(1).

Deleting (deQueue) from a queue can happen at the front of the queue only. Since we know the front of the queue already, the deleting run time will be O(1) too.

So, the run time complexity of a sequence of one enqueue and one dequeue operation will be O(1). We can see clearly in the table that the average execution time is nearly constant for different size of queue.

N	Average execution time (second)
1	7.96e-08
10	8.68e-08
100	8.09e-08
1000	8.05e-08
10000	8.25e-08
100000	7.86e-08
1000000	8.38e-08



A sample code for enQueue.

```
void enQueue(struct Queue *q, float num)
{
	struct QueueNode *temp;
	temp = newNode(num);
	if (q->rear == NULL)
	{
		q->front = temp;
		q->rear = temp;
		return;
	}
	q->rear->next = temp;
	q->rear = temp;
}
```

A sample code for enQueue.

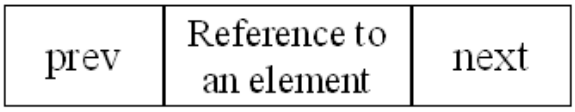
```
struct QueueNode *deQueue(struct Queue *q)
{
	if (q->front == NULL)
	{
		return NULL;
	}
	struct QueueNode *temp;
	temp = q->front;
	q->front = q->front->next;
	if (q->front == NULL)
	{
		q->rear = NULL;
	}
	return temp;
}
```

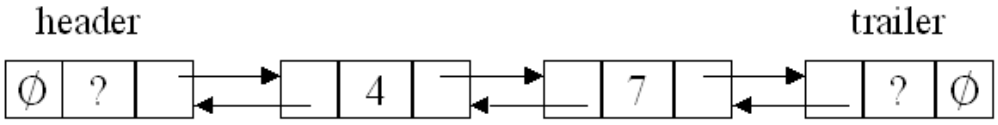
2. Doubly Linked List

A doubly linked list is a data structure that contains a set of sequence of nodes. All those nodes are connected by previous and next pointers. Since, we know the exact node that we want to add and remove, the running time for inserting and deleting will be O(1).

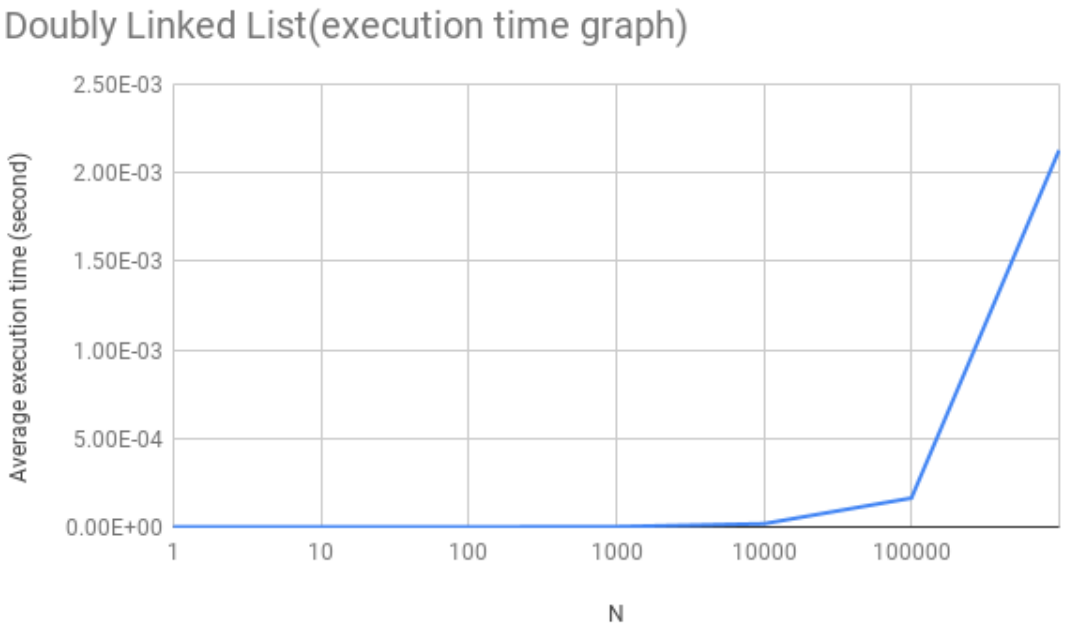
When we do the searching, we have to search and compare all the value starting from beginning until it reaches the end of the list or the value is found. The worst case will be that the key value is not found the list; so, the worst-case running time will be O(n). The average running time for searching and deleting case will be O(n).

We can see clearly that the execution time graph is going to longer time as the value of N increased.





N	Average execution time (second)
1	1e-07
10	1.2e-07
100	2.7e-07
1000	1.78e-06
10000	1.841e-05
100000	0.00016325
1000000	0.00212683


Doubly Linked List(execution time graph)
Average execution time (second)
2.50E-03
2.00E-03
1.50E-03
1.00E-03
5.00E-04
0.00E+00
1
10
100
1000
10000
100000
N

A sample code for searching the list.

```
struct Node *searchList(struct Node* start, int key)
{
	struct Node *temp = start;
	if(temp != NULL)
	{
		while(temp->next != NULL)
		{
			if(temp->data == key)
			{
				break;
			}
			temp = temp->next;
		}
	}
	else
	{
		return NULL;
	}
	return temp;
}
```

A sample code for deleting a node.

```
void deleteNode(struct Node** start, struct Node* delNode)
{
	if (*start == NULL || delNode == NULL)
	{
		return;
	}

	if (*start == delNode)
	{
		*start = delNode->next;
	}

	if (delNode->next != NULL)
	{
		delNode->next->prev = delNode->prev;
	}

	if (delNode->prev != NULL)
	{
		delNode->prev->next = delNode->next;
	}

	free(delNode);
	return;
}
```