American International University – Bangladesh

CSC 2211: Algorithms [F] Lab Exam

1. Write a warm up code to select your problem set.

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int middleId = 1997;
    cout<<(middleId%3+1)<<endl;
return 0;
}
```

2. According to your output, select the same number of problems to solve in the lab exam.
    1 Linear Search and Binary Search
    2 Selection Sort and Merge Sort
    3 Insertion sort and Quick Sort
3. Generate 100000 and 1000000 random data for the test cases.

Linear Search

Algorithm 5 Linear Search

```
1: procedure LINEAR(A, n, item)
2:     for i ← 0, n − 1 do
3:         if A[i] == item then
4:             return i
5:         end if
6:     end for
7:     return −1
8: end procedure
```

Binary Search

Algorithm 9 Binary Search Recursive algorithm

```
1: procedure BinarySearch(A, low, high, x)
2:     if low > high then
3:         return −1
4:     end if
5:     mid = (low + high)/2
6:     if x == A[mid] then
7:         return mid
8:     else if x < A[mid] then
9:         return BinarySearch(A, low, mid − 1, x)
10:    else
11:        return BinarySearch(A, mid + 1, high, x)
12:    end if
13: end procedure
```

Selection Sort

Algorithm 5 Selection Sort

```
1: procedure SelectionSort(A, n)
2:     for i ← 0, n − 1 do
3:         iMin ← i
4:         for j ← i + 1, n − 1 do
5:             if A[j] < A[iMin] then
6:                 iMin = j
7:             end if
8:             swap(A[iMin], A[i])
9:         end for
10:    end for
11: end procedure
```

Merge Sort

Algorithm 8 Merge

```
1:  procedure MERGE(A, left, mid, right)
2:      n1 = mid − left + 1
3:      n2 = right − mid
          L[1....n1]andR[1....n2]
4:      for i ← 0, n1 − 1 do
          L[i] ← A[left + i]
5:      end for
6:      for j ← 0, n2 − 1 do
          R[j] ← A[mid + 1 + j]
7:      end for
8:      i ← 0, j ← 0, k ← left
9:      while i ≤ n1 − 1&j ≤ n2 − 1 do
10:         if L[i] < R[j] then
11:             A[k + +] ← L[i + +]
12:         else
13:             A[k + +] ← R[i + +]
14:         end if
15:     end while
16:     while i <= n1 − 1 do
17:         A[k + +] ← L[i + +]
18:     end while
19:     while j <= n2 − 1 do
20:         A[k + +] ← R[j + +]
21:     end while
22: end procedure
```

Algorithm 9 Merge Sort

```
1:  procedure MERGESORT(A, left, right)
2:      if left < right then
3:          mid = (left + right)/2
4:          MERGESORT(A, left, mid)
5:          MERGESORT(A, mid + 1, right)
6:          MERGE(A, left, mid, right)
7:      end if
8:  end procedure
```

Insertion sort

Algorithm 8 Insertion Sort

```
1: procedure INSERTIONSORT(A, n)
2:     for j ← 1, n − 1 do
3:         value ← A[j]
4:         i ← j − 1
5:         while i > 0&A[i] > value do
6:             swap(A[i], A[i + 1])
7:             i ← i − 1
8:         end while
9:         A[i + 1] = value
10:    end for
11: end procedure
```

Quick Sort

Algorithm 10 partition

```
1: procedure PARTITION(A, start, end)
2:     pivot = A[end]
3:     pIndex = start
4:     for i ← start, end − 1 do
5:         if A[i] < pivot then
6:             swap(A[i], A[pIndex]) pIndex + +
7:         end if
8:     end for
9:     swap(A[pIndex], A[end]) return pIndex
10: end procedure
```

Algorithm 11 Quick Sort

```
1: procedure QUICKSORT(A, start, end)
2:     if start >= end then
3:         pIndex =PARTITION(A, start, end)
4:         QUICKSORT(A, start, pIndex − 1)
5:         QUICKSORT(A, pIndex + 1, end)
6:     end if
7: end procedure
```