

# Algorithm Lab 4

Supta Philip Philip

October 10, 2022

## 1 Pre-Requisite in C++

1. Function
2. Pointer
3. Array / Dynamic Array
4. Dynamic Array - 2D/ Matrix
5. Random number
6. File Operation
7. Standard Template Library(STL)

## 2 Recursive Function

1. Fibonacci
2. Factorial
3. Euclid's algorithm for GCD

---

**Algorithm 1** Fibonacci

---

```
1: procedure FIBONACCI( $n$ )
2:   if  $n \leq 1$  then
3:     return  $n$ 
4:   else
5:     return FIBONACCI( $n - 1$ )+FIBONACCI( $n - 2$ )
6:   end if
7: end procedure
```

---

---

**Algorithm 2** Factorial

---

```
1: procedure FACTORIAL( $n$ )
2:   if  $n \leq 0$  then
3:     return 1
4:   else
5:     return  $n * \text{FACTORIAL}(n - 1)$ 
6:   end if
7: end procedure
```

---

---

**Algorithm 3** Euclid's algorithm[Recursive]

---

```
1: procedure GCD( $n, m$ )
2:   if  $n = m$  then
3:     return  $m$ 
4:   else if  $n \geq m$  then
5:     return GCD( $n - m, m$ )
6:   else
7:     return GCD( $n, m - n$ )
8:   end if
9: end procedure
```

---

---

**Algorithm 4** Euclid's algorithm for GCD[Iterative]

---

```
1: procedure GCD( $a, b$ ) ▷ The g.c.d. of  $a$  and  $b$ 
2:    $r \leftarrow a \bmod b$ 
3:   while  $r \neq 0$  do ▷ We have the answer if  $r$  is 0
4:      $a \leftarrow b$ 
5:      $b \leftarrow r$ 
6:      $r \leftarrow a \bmod b$ 
7:   end while
8:   return  $b$  ▷ The gcd is  $b$ 
9: end procedure
```

---

### 3 Iterative Algorithm

1. Linear Search
2. Bubble sort
3. Selection sort
4. Insertion Sort

---

**Algorithm 5** Linear Search

---

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

---

---

**Algorithm 6** Bubble Sort

---

```
1: procedure BUBBLESORT( $A, n$ )
2:   for  $k \leftarrow 0, n - 1$  do
3:     for  $i \leftarrow 0, n - 1$  do
4:       if  $A[i] > A[i + 1]$  then
5:          $swap(A[i], A[i + 1])$ 
6:       end if
7:     end for
8:   end for
9: end procedure
```

---

---

**Algorithm 7** Selection Sort

---

```
1: procedure SELECTIONSORT( $A, n$ )
2:   for  $i \leftarrow 0, n - 1$  do
3:      $iMin \leftarrow i$ 
4:     for  $j \leftarrow i + 1, n - 1$  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
```

---

---

**Algorithm 8** Insertion Sort

---

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

---

## 4 Divide and Conquer Algorithm

---

**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
```

---

---

**Algorithm 10** Binary Search Iterative algorithm

---

```
1: procedure BINARYSEARCH( $A, n, x$ )
2:    $low = 0, high = n - 1$ 
3:   while  $low < high$  do
4:      $mid = (low + high)/2$ 
5:     if  $x == A[mid]$  then
6:       return  $mid$ 
7:     end if
8:     if  $x < A[mid]$  then
9:        $high = mid - 1$ 
10:    else
11:       $low = mid + 1$ 
12:    end if
13:  end while
14:  return  $-1$ 
15: end procedure
```

---

---

**Algorithm 11** Merge Two Arrays

---

```
1: procedure MERGE( $A, B, n, m$ )
2:    $i \leftarrow 0, j \leftarrow 0, k \leftarrow 0$ 
3:   while  $i \leq n - 1 \& j \leq m - 1$  do
4:     if  $A[i] > B[j]$  then
5:        $C[k++] \leftarrow A[i++]$ 
6:     else
7:        $C[k++] \leftarrow B[j++]$ 
8:     end if
9:   end while
10:  while  $i \leq n - 1$  do
11:     $C[k++] \leftarrow A[i++]$ 
12:  end while
13:  while  $j \leq m - 1$  do
14:     $C[k++] \leftarrow B[j++]$ 
15:  end while
16: end procedure
```

---

---

**Algorithm 12** Merge

---

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

---

---

**Algorithm 13** 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
```

---