Algorithm Lab 4

Supta Richard Philip

February 14, 2023

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

```
procedure FIBONACCI($n$)
    if $n \leq 1$ then
        return n
    else
        return FIBONACCI($n-1$)+FIBONACCI($n-2$)
    end if
end procedure
```

3 Divide and Conquer Algorithm

Algorithm 2 Factorial

```
procedure FACTORIAL($n$)
    if $n \leq 0$ then
        return 1
    else
        return n*FACTORIAL($n - 1$)
    end if
end procedure
```

Algorithm 3 Euclid's algorithm[Recursive]

```
procedure GCD($n, m$)
    if $n = m$ then
        return $m$
    else if $n \geq m$ then
        return GCD($n - m, m$)
    else
        return GCD($n, m - n$)
    end if
end procedure
```

Algorithm 4 Euclid's algorithm for GCD[Iterative]

```
procedure GCD($a, b$)                          ▷ The g.c.d. of a and b
    $r \leftarrow a \bmod b$
    while $r \neq 0$ do                        ▷ We have the answer if r is 0
        $a \leftarrow b$
        $b \leftarrow r$
        $r \leftarrow a \bmod b$
    end while
    return $b$                                 ▷ The gcd is b
end procedure
```

Algorithm 5 Binary Search Recursive algorithm

```
procedure BinarySearch($A, low, high, x$)
    if $low > high$ then
        return $-1$
    end if
    $mid = (low + high)/2$
    if $x == A[mid]$ then
        return $mid$
    else if $x < A[mid]$ then
        return BinarySearch($A, low, mid - 1, x$)
    else
        return BinarySearch($A, mid + 1, high, x$)
    end if
end procedure
```

Algorithm 6 Binary Search Iterative algorithm

```
procedure BinarySearch($A, n, x$)
    $low = 0, high = n - 1$
    while $low < high$ do
        $mid = (low + high)/2$
        if $x == A[mid]$ then
            return $mid$
        end if
        if $x < A[mid]$ then
            $high = mid - 1$
        else
            $low = mid + 1$
        end if
    end while
    return $-1$
end procedure
```

Algorithm 7 Merge Two Arrays

```
procedure MERGE($A, B, n, m$)
    $i \leftarrow 0, j \leftarrow 0, k \leftarrow 0$
    while $i \leq n - 1 \& j \leq m - 1$ do
        if $A[i] > B[j]$ then
            $C[k++] \leftarrow B[j++]$
        else
            $C[k++] \leftarrow A[i++]$
        end if
    end while
    while $i <= n - 1$ do
        $C[k++] \leftarrow A[i++]$
    end while
    while $j <= m - 1$ do
        $C[k++] \leftarrow B[j++]$
    end while
end procedure
```

Algorithm 8 Merge

```
procedure MERGE($A, left, mid, right$)
    $n1 = mid - left + 1$
    $n2 = right - mid$
      $L[1....n1] and R[1....n2]$
    for $i \leftarrow 0, n1 - 1$ do
        $L[i] \leftarrow A[left + i]$
    end for
    for $j \leftarrow 0, n2 - 1$ do
        $R[j] \leftarrow A[mid + 1 + j]$
    end for
    $i \leftarrow 0, j \leftarrow 0, k \leftarrow left$
    while $i \leq n1 - 1 \& j \leq n2 - 1$ do
        if $L[i] < R[j]$ then
            $A[k++] \leftarrow L[i++]$
        else
            $A[k++] \leftarrow R[i++]$
        end if
    end while
    while $i <= n1 - 1$ do
        $A[k++] \leftarrow L[i++]$
    end while
    while $j <= n2 - 1$ do
        $A[k++] \leftarrow R[j++]$
    end while
end procedure
```

Algorithm 9 Merge Sort

```
procedure MergeSort($A, left, right$)
    if $left < right$ then
        $mid = (left + right)/2$
        MergeSort($A, left, mid$)
        MergeSort($A, mid + 1, right$)
        Merge($A, left, mid, right$)
    end if
end procedure
```

Algorithm 10 partition

```
procedure partition($A, start, end$)
    $pivot = A[end]$
    $pIndex = start$
    for $i \leftarrow start, end - 1$ do
        if $A[i] < pivot$ then
            $swap(A[i], A[pIndex])$
            $pIndex + +$
        end if
    end for
    $swap(A[pIndex], A[end])$
    return $pIndex$
end procedure
```

Algorithm 11 Quick Sort

```
procedure QuickSort($A, start, end$)
    if $start >= end$ then
        $pIndex =$partition($A, start, end$)
        QuickSort($A, start, pIndex - 1$)
        QuickSort($A, pIndex + 1, end$)
    end if
end procedure
```