Ahmed Abd-Elsalam Muhammed Afify

AI_Algorithm_Assignment

Factorial Iterative

In [10]:
```
def fact_iter(n):
    if n<0:
        factorial = 0
    elif n in [0,1]:
        factorial = 1
    else:
        factorial = 1
        while n>1:
            factorial *= n
            n -= 1

    return factorial

#---------------- Test ----------------------
start_time = datetime.datetime.now()
fact = fact_iter(5)
print(fact)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
120
0:00:00
```

Recursive Factorial

In [11]:
```python
import datetime

def fact_recur(n):
    if n<0:
        factorial = 0
    elif n in [0,1]:
        factorial = 1
    else:
        factorial = n*fact_recur(n-1)
    return factorial

# Complexity O(n)

#---------------- Test ----------------------
start_time = datetime.datetime.now()
fact = fact_recur(5)
print(fact)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
120
0:00:00.001000
```

Fibonacci Iterative

In [24]:
```
def fibonacci_loop(n):
    if n<0 :
        raise Exception
        "Input Error"
    elif n == 0:
        return 0
    elif n in [1, 2]:
        return 1
    else:
        fabonnaci_a = 0
        fabonnaci = 1
        for i in range(1,n):
            c = fabonnaci_a + fabonnaci
            fabonnaci_a = fabonnaci
            fabonnaci = c

    return fabonnaci


# Complexity O(n)
#---------------- Test ----------------------
start_time = datetime.datetime.now()
v = fibonacci_loop(20)
print(v)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
6765
0:00:00.001000
```

Fibonacci Recursion

In [23]:
```
def fibonacci_recur(n):
    if n<0 :
        raise Exception
        "Input Error"
    elif n == 0:
        return 0
    elif n in [1, 2]:
        return 1
    else:
        return fabonnaci_recur(n-1)+fabonnaci_recur(n-2)


# Complexity O(2^n)

#---------------- Test ----------------------
start_time = datetime.datetime.now()
v = fibonacci_recur(20)
print(v)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
6765
0:00:00.003998
```

Iterative Linear Search

In [27]:
```
def search_loop(array, x):

    for i in range(len(array)):
        if arr[i] == x:
            return i

    return "Not Found"


# Complexity O(n)

# ------------------ Test --------------
start_time = datetime.datetime.now()
arr = [1,2,3,4,5,6,7,8,9]
x = 8
index = search_loop(arr,x)
print(index)
end_time = datetime.datetime.now()
print(end_time - start_time)

x = 10
index = search_loop(arr,x)
print(index)
```

```
7
0:00:00.003999
Not Found
```

Recursion linear search

In [29]:
```
def search_rec( array, begin, end, x):
    if end < begin:
        return -1
    if arr[begin] == x:
        return begin
    if arr[end] == x:
        return end
    return recSearch(arr, begin+1, end-1, x)


# Complexity O(n)

# ------------------- Test ---------------
start_time = datetime.datetime.now()
arr = [1,2,3,4,5,6,7,8,9]
x = 8
begin = 0
end = len(arr)-1
index = search_rec(arr,begin,end,x)
print(index)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
7
0:00:00
```

Insertion sort

In [31]:
```
def insertionSort(array):
    for i in range(1, len(array)):
        key = array[i]
        j = i-1
        while j >=0 and key < array[j] :
                array[j+1] = array[j]
                j -= 1
        array[j+1] = key
    return array


# Complexity O(n*n)

# ------------------- Test ---------------
start_time = datetime.datetime.now()
arr = [8,6,4,3,8,1,3,5,8,7,6]
sorted_array = insertionSort(arr)
print(sorted_array)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
[1, 3, 3, 4, 5, 6, 6, 7, 8, 8, 8]
0:00:00
```

Bubble sort

In [33]:
```
def bubbleSort(array):
    n = len(array)
    for i in range(n-1):
        for j in range(0, n-i-1):
            if array[j] > array[j+1] :
                array[j], array[j+1] = array[j+1], array[j]
    return array


# Complexity O(n*n)

# ------------------ Test ---------------
start_time = datetime.datetime.now()
arr = [8,6,4,3,8,1,3,5,8,7,6]
sorted_array = bubbleSort(arr)
print(sorted_array)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
[1, 3, 3, 4, 5, 6, 6, 7, 8, 8, 8]
0:00:00
```

Binary Search Iterative

In [37]:
```
def binarySearch_iter(array, begin, end, x):
    while begin <= end:
        mid = begin + (end - begin) // 2;
        if array[mid] == x:
            return mid
        elif array[mid] < x:
            begin = mid + 1
        else:
            end = mid - 1
    return -1


# Complexity O(log(n))

# ------------------ Test ---------------
start_time = datetime.datetime.now()
arr = [1,2,3,4,5,6,7,8,9]
x = 8
begin = 0
end = len(arr)-1
index = binarySearch_iter(arr,begin,end,x)
print(index)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
7
0:00:00.001000
```

Binary Search Recursive

In [36]:
```
def binarySearch_recer(array, begin, end, x):
    if end >= begin:
        mid = begin + (end-begin)//2
        if arr[mid]==x:
            return mid
        elif arr[mid] > x:
            return binarySearch_recer(array, begin, mid-1, x)
        else:
            return binarySearch_recer(array, mid + 1, end, x)
    else:
        return -1


# Complexity O(log(n))

# ------------------- Test ---------------
start_time = datetime.datetime.now()
arr = [1,2,3,4,5,6,7,8,9]
x = 8
begin = 0
end = len(arr)-1
index = binarySearch_recer(arr,begin,end,x)
print(index)
end_time = datetime.datetime.now()
print(end_time - start_time)
```

```
7
0:00:00.000998
```

In [ ]:
```
```