# Assignment 3

Pseudocode – Selection Sort

```
Algorithm Selection_Sort(A)
    Input array or string A of size n
    Output array or string A of size n

    for i ← 0 to n-2 do
            index ← i
            for j ← i to n-1 do
                if A[index] > A[j]
                    index ← j
            if index != i
                swap(A[index], A[i])
    return A
```

Pseudocode – Insertion Sort

```
Algorithm Insertion_Sort(A)
    Input array or string A of size n
    Output array or string A of size n

    for i ← 0 to n-1 do
            index ← i
            while index >0 and A[index-1] > A[index]
                    swap(A[index-1], A[index])
                    index ← index – 1
    return A
```

# Priority Queue

โครงสร้างพื้นฐานของ Priority Queue with linked list

```
class PriorityQueue:
    class Node:
        def __init__(self,k,v):

            self.key=k
            self.value=v
            self.next=None


    def __init__(self):

        self.front = None
        self.size=0
```

1) Method IsEmpty():

```
def isEmpty(self):
    if self.front == None:
        return (True)
    else:
        return (False)
```



pseudocode:

If front-node == None

return True

else:

Return False

```
def insert(self,key,value):

    if self.isEmpty() == True:

        self.front=Node(key,value)

        self.size+=1


    else:

        if self.front.key > key: #

            new=Node(key,value)

            new.next=self.front

            self.front=new

            self.size+=1
```

```
else:

    A=Node(key,value)

    currentnode=self.front

    while currentnode.next:

        if key < currentnode.next.key:
            break

        currentnode=currentnode.next


    A.next=currentnode.next
    currentnode.next=A

    self.size+=1
```

Else condition ต่อจากภาพด้านซ้าย

2)Method insert(key,value)

Pseudocode:

```
If linked list isEmpty()
    Front-node = Node(key,value)
    Size of linked list + 1

else:
    if  front-node.key > key
        new.next ← front-node
        front-node ← new
        size ← size+1

    else
        A←Node(key,value)
        currentnode←front-node
        While currentnode.next
                If key < currentnode.next.key
                        break
                currentnode←currentnode.next
        A.next←currentnode.next
        Currentnode.next←A
```

```
def remove_min(self):
    if self.isEmpty():
        return
    else:
        A=self.front.value
        self.front=self.front.next
        self.size-=1
        return A
```

3)Method remove_min()

pseudocode:

If Queue is Empty

return

else

front-node←front.next

size←size-1

return front-node

```
def Pri_size(self):
    return self.size

def min_key(self):
    if self.isEmpty():
        return
    else:
        return(self.front.key)
```

4)Method Pri_size()

Pseudocode:

return size

5)Method min_key()

Pseudocode:

If Queue is empty

return

else

return front.key

```
def min_element(self):
    if self.isEmpty():
        return
    else:
        return(self.front.value)
```

6)Method min_element()

If Queue is empty

return

else

return front-value

## Implementation Priority Queue

```
A=PriorityQueue()
A.insert(1,100)
A.insert(2,-50)
A.insert(3,89)
A.insert(4,-2)
A.insert(5,6)
A.insert(6,-9)
A.insert(7,10)
A.insert(8,66)
print("\nSorting element")
print(A.Sort_element())
print("\nremoving minimum element with minimum key")
print(A.remove_min())
print("\ntraverse nodes")
print(A.traverse_node())
```

เรียก print method remove_min() เช่นดังตัวอย่าง remove element with minimum key ออกจาก priority queue

```
Sorting element
[-50, -9, -2, 6, 10, 66, 89, 100]

removing minimum element with minimum key
100
```

เมื่อเสร็จขั้นตอน remove element with minimum key แล้ว traverse node ดู จะเห็นได้ว่า Node(1,100) ได้ถูกลบออกไปแล้วดังภาพต่อไปนี้

```
traverse nodes
-50
89
-2
6
-9
10
66
None
```

เมื่อเรียก print function min_key กับ min_element

ตัวอย่าง:

```
print(A.min_key())
print("\nเรียกใช้ min_element()")
print(A.min_element())
```

ผลที่ได้:

```
เรียกใช้ min_key()
2

เรียกใช้ min_element()
-50
```

จะเห็นได้ว่าเป็น minimum key กับ min element ที่น้อยที่สุดหลังจากได้ทำการลบ Node(1,100)

## Implementation Selection Sort and Insertion Sort

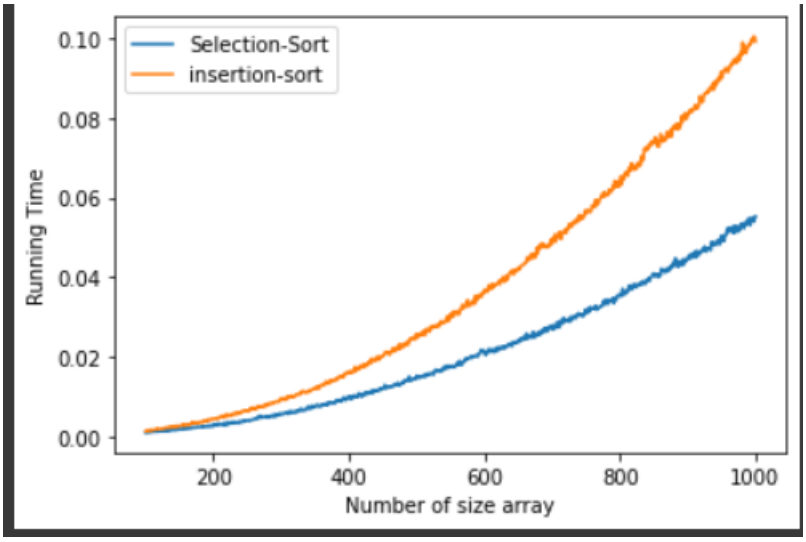
1. Selection Sort

```
def Selection_Sort(A):

    for i in range(len(A)-1):#---> Do (N-1) iterations#
                                                      #
        index=i                                       #
                                                      #
        for j in range(i,len(A)): #-------------------#
                                                      #
            if A[index]>A[j]:                         #
                index=j                               #-------> O(N^2)
                                                      #
        if index != i:                                #
                                                      #
            A[index],A[i]=A[i],A[index]#--------------#


    return A
```

## 2. Insertion Sort

```python
def insertion_sort(A):

    for i in range(len(A)):

        index=i

        while index>0 and A[index-1]>A[index]:

            A[index-1],A[index]=A[index],A[index-1]

            index-=1


    return(A)
```

เปรียบเทียบ running time complexity ของสอง algorithm จะได้คร่าวๆดังนี้



# Implementation of PQ-Sort

```python
#Algorithm PQ-Sort

from priorityqueue import *


def PQSort(S):
    P=PriorityQueue()
    n=len(S)

    for i in range (len(S)):
      e=S[0]
      P.insert(i,e)
      S.pop(0)


    A=P.Sort_element()

    for i in range(n):

        S.append(A[i])

    return S

print(PQSort([1,3,2,-100,-50,23,45,48,77]))
```

ผลที่ได้:

```
[-100, -50, 1, 2, 3, 23, 45, 48, 77]
PS C:\Users\yoshi\Desktop\Code>
```

**การจัดเรียงคำไทยแบบพจนานุกรมไทย**

โดยในภาษา python สามารถเรียกใช้ function ที่มีชื่อว่า sorted(<List>) จะสามารถจัดเรียงได้ตามปกติในภาษาอังกฤษ แต่ในภาษาไทยนั้นจะไม่ถูกจัดเรียงตามพจนานุกรมไทย ตัวอย่างเช่น

```
Country=["พม่า","ไทย","อังกฤษ","อเมริกา","ไอร์แลนด์","กรีซ","เกาหลี","เอกวาดอ"]
print("\n ยังไม่ทำการจัดเรียงแบบถูกวิธี")
print(sorted(Country))
```

ผลที่ได้:

```
 ยังไม่ทำการจัดเรียงแบบถูกวิธี
['กรีซ', 'พม่า', 'อังกฤษ', 'อเมริกา', 'เกาหลี', 'เอกวาดอ', 'ไทย', 'ไอร์แลนด์']
```

แต่ถ้าใช้ Module ที่ชื่อว่า pyuca แล้วใช้ function ที่ชื่อว่า Collator จะสามารถจัดเรียงได้ตรงตามพจนานุกรมไทยได้ตัวอย่างเช่น

```
Correct=sorted(Country,key=Collator().sort_key)
print(Correct)
```

ผลที่ได้:

```
 ได้ทำการจัดเรียงจาก Module pyuca
['กรีซ', 'เกาหลี', 'ไทย', 'พม่า', 'อเมริกา', 'อังกฤษ', 'เอกวาดอ', 'ไอร์แลนด์']
```