Programming and Data Structures with Python

Lecture 10, 21 January 2021

In [1]:

```
def SelectionSort(l):
    # Scan slices l[0:len(l)], l[1:len(l)], ...
    for start in range(len(l)):
        # Find minimum value in slice . . .
        minpos = start
        for i in range(start,len(l)):
            if l[i] < l[minpos]:
                minpos = i
        # . . . and move it to start of slice
        (l[start],l[minpos]) = (l[minpos],l[start])
```

In [2]:

```
l = list(range(10,0,-1))
```

In [3]:

```
l
```

Out[3]:

```
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [4]:

```
SelectionSort(l)
```

In [5]:

```
l
```

Out[5]:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [6]:

```
l = list(range(100,90,-1))
```

In [7]:

```
l
```

Out[7]:

```
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91]
```

In [8]:

```
def InsertionSort(seq):
    isort(seq,len(seq))

def isort(seq,k): # Sort slice seq[0:k]
    if k > 1:
        isort(seq,k-1)
        insert(seq,k-1)

def insert(seq,k): # Insert seq[k] into sorted seq[0:k-1]
    pos = k
    while pos > 0 and seq[pos] < seq[pos-1]:
        (seq[pos],seq[pos-1]) = (seq[pos-1],seq[pos])
        pos = pos-1
```

In [9]:

```
InsertionSort(l)
```

In [10]:

```
l
```

Out[10]:

```
[91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

In [11]:

```
def merge(A,B): # Merge A[0:m],B[0:n]
    (C,m,n) = ([],len(A),len(B))
    (i,j) = (0,0) # Current positions in A,B
    while i+j < m+n: # i+j is number of elements merged so far
        if i == m: # Case 1: A is empty
            C.append(B[j])
            j = j+1
        elif j == n: # Case 2: B is empty
            C.append(A[i])
            i = i+1
        elif A[i] <= B[j]: # Case 3: Head of A is smaller
            C.append(A[i])
            i = i+1
        elif A[i] > B[j]: # Case 4: Head of B is smaller
            C.append(B[j])
            j = j+1
    return(C)
```

In [12]:

```
merge([1,3,5],[2,4,6])
```

Out[12]:

```
[1, 2, 3, 4, 5, 6]
```

In [13]:

```
def mergesort(A,left,right):
    # Sort the slice A[left:right]
    if right - left <= 1: # Base case
        return(A[left:right])
    if right - left > 1:  # Recursive call
        mid = (left+right)//2
        L = mergesort(A,left,mid)
        R = mergesort(A,mid,right)
        return(merge(L,R))
```

In [14]:

```
mergesort([1,3,5,2,4,6],0,6)
```

Out[14]:

```
[1, 2, 3, 4, 5, 6]
```