Tutorial 5

September 21, 2018

1 Sorting

1.1 Define Bubble Sort

```
In [ ]: def BubbleSort(arr):
            # Get the length of input list
            n = len(arr)

            # Traverse through all array elements
            for i in range(n):

                # Last i elements are already in place
                for j in range(0, n-i-1):

                    # traverse the array from 0 to n-i-1
                    # Swap if the element found > next element
                    if arr[j] > arr[j+1]:
                        arr[j], arr[j+1] = arr[j+1], arr[j]
```

1.2 Define Selection Sort (Smallest)

```
In [ ]: def SelectionSortSmallest(arr):
            # Get the length of input list
            n = len(arr)

            # Traverse through all array elements
            for i in range(n):

                # Find the minimum element in remaining unsorted list
                index = i
                for j in range(i+1, n):
                    if arr[index] > arr[j]:
                        index = j

                # Swap the found minimum element with the first element
                arr[i], arr[index] = arr[index], arr[i]
```

1.3 Define Selection Sort (Largest)

```
In [ ]: def SelectionSortLargest(arr):
            # Get the length of input list
            n = len(arr)

            # Traverse through all array elements
            for i in range(n-1, 0, -1):

                # Find the maximum element in remaining unsorted list
                index = 0
                for j in range(1, i+1):
                    if arr[j] > arr[index]:
                        index = j

                # Swap the found maximum element with the last element
                arr[i], arr[index] = arr[index], arr[i]
```

1.4 Define Insertion Sort

```
In [ ]: def InsertionSort(arr):
            # Get the length of input list
            n = len(arr)

            # Traverse through all array elements (except first element)
            for i in range(1, n):
                position = i
                currentvalue = arr[position]

                while position > 0 and arr[position-1] > currentvalue:
                    arr[position] = arr[position-1]
                    position = position - 1

                arr[position] = currentvalue
```

1.5 Define Merge Sort

```
In [ ]: def MergeSort(alist):
        #    print("Splitting ",alist)
            if len(alist) >1:           # Check the length of input list
                mid = len(alist)//2     # Find the middle
                lefthalf = alist[:mid]  # Define the left list
                righthalf = alist[mid:] # Define the right list

                MergeSort(lefthalf)     # Merge Sort the left side
                MergeSort(righthalf)    # Merge Sort the right side

                i=0
```

```
            j=0
            k=0
            while i < len(lefthalf) and j < len(righthalf):
                if lefthalf[i] < righthalf[j]:
                    alist[k]=lefthalf[i]
                    i=i+1
                else:
                    alist[k]=righthalf[j]
                    j=j+1
                k=k+1

            while i < len(lefthalf):
                alist[k]=lefthalf[i]
                i=i+1
                k=k+1

            while j < len(righthalf):
                alist[k]=righthalf[j]
                j=j+1
                k=k+1
    #    print("Merging ",alist)
```

1.6 Quick Sort

```
In [ ]: def QuickSort(arr):
            # Skip process if only 1 element
            if len(arr) <= 1:
                return arr

            # Take middle element as pivot
            pivot = arr[len(arr) // 2]

            # Left Sublist
            left = [x for x in arr if x < pivot]

            # Middle SubList
            middle = [x for x in arr if x == pivot]

            # Right SubList
            right = [x for x in arr if x > pivot]

            # Use recursive to sort the SubList
            return QuickSort(left) + middle + QuickSort(right)
```

1.7 Compare Sorting Time

```
In [ ]: import random
```

```
# 1) Bubble Sort
SampleList = random.sample(range(0,1000), 1000)
%timeit BubbleSort(SampleList)

# 2) Selection Sort (Use Smallest)
SampleList = random.sample(range(0,1000), 1000)
%timeit SelectionSortSmallest(SampleList)

# 3) Selection Sort (Use Larget)
SampleList = random.sample(range(0,1000), 1000)
%timeit SelectionSortLargest(SampleList)

# 4) Insertion Sort
SampleList = random.sample(range(0,1000), 1000)
%timeit InsertionSort(SampleList)

# 5) Merge Sort
SampleList = random.sample(range(0,1000), 1000)
%timeit MergeSort(SampleList)

# 6) Quick Sort
SampleList = random.sample(range(0,1000), 1000)
%timeit QuickSort(SampleList)

# 7) Python Sort
SampleList = random.sample(range(0,1000), 1000)
%timeit SampleList.sort()
```

2 Searching

2.1 Sequential Search

```
In [ ]: def SequentialSearch(Input_List, Search_Item):
            # Define the found flag
            found = False

            # Travel through the list
            for i in Input_List:
                # Stop if item found
                if i == Search_Item:
                    found = True
                    break

            # Return result
            return found
```

2.2 Binary Search

```
In [ ]: def BinarySearch(Input_List, Search_Item):
            first = 0
            last = len(Input_List)-1
            found = False

            while first<=last and not found:
                midpoint = (first + last)//2
                if Input_List[midpoint] == Search_Item:
                    found = True
                    break
                else:
                    if Search_Item < Input_List[midpoint]:
                        last = midpoint-1
                    else:
                        first = midpoint+1

            # Return result
            return found
```

2.3 Compare Searching Time

```
In [ ]: import random

        # Generate a sorted list
        SampleList = random.sample(range(0,1000), 1000)
        SampleList.sort()

        %timeit SequentialSearch(SampleList, 888)
        %timeit BinarySearch(SampleList, 888)
```