CMPT 435 - Spring 2019 - Dr. Labouseur

Assignment Duo

Jae Kyoung Lee (LJ)

March 15, 2019

1 Selection Sort

```python
# Jae Kyoung Lee (LJ)

# This will store the comparisons
comparisons = 0

def selectionSort(inputList):
  global comparisons
  for i in range(0, len(inputList)):
    # Settting minimum value in the list as temp
    minVal = i
    for j in range(i+1, len(inputList)):
      # Checks whether the value in minVal is bigger than
      # the list of array besides minVal
      if inputList[minVal] > inputList[j]:
        # Increment how many times we've compared so far
        comparisons+=1
        # If it's true, then change minVal to current value
        minVal = j
      # End of if statement
    # Do the swapping
    temp = inputList[i]
    inputList[i] = inputList[minVal]
    inputList[minVal] = temp
    # end of for loop
  # end of for loop
  return inputList
```

2 Insertion Sort

```python
# This will store the comparisons
comparisons = 0

def insertionSort(inputList):
 global comparisons
 for i in range(1, len(inputList)):
   j = i-1
   while j >= 0 and inputList[j] > inputList[j+1]:
     # Increment how many times we've compared so far
     comparisons+=1
     # Temporary variable that stores the current value
     tempVal = inputList[j]
     # Current value becomes the next value or next value becomes the current value
     inputList[j] = inputList[j+1]
     # Next element overwrites the temporary value
     inputList[j+1] = tempVal
     j-=1
 return inputList
```

3 Quick Sort

```python
# This will store the comparisons
comparisons = 0
def quickSort(inputList):
 global comparisons
 if len(inputList) <= 1:
   return inputList
 else:
   leftArray=[]
   rightArray=[]
   equalArray = []
   # Pivot will always start from the start
   pivot = inputList[0]
   # Traverse through the inputList
   for i in inputList:
     # Comparing each value to pivot then add to the corresponding array
     if i < pivot:
       comparisons+=1
       leftArray.append(i)
     elif i > pivot:
       comparisons+1
       rightArray.append(i)
     elif i == pivot: # Check for equality since there might be another element that has the same value
       comparisons+=1
       equalArray.append(i)
   # Repeat this until the array is sorted
   sortedList = quickSort(leftArray)+equalArray+quickSort(rightArray)
   return sortedList
```

4 Merge Sort

```python
# This will store the comparisons
comparisons = 0

def merge(leftArr, rightArr):
 global comparisons
 # Empty array to store the sorted list of both inputs
 sortedList = []
 while len(leftArr) > 0 and len(rightArr) > 0:
   # Comparing to see the first element in both arrays
   if leftArr[0] < rightArr[0]:
     # Increment comparisons because we just compared
     comparisons+=1
     # Storing the first element because we just compared
     # and found out that the element is less than the value in right array
     sortedList.append(leftArr[0])
     # Since we sorted the first element, we no longer have to check for that element again, so get rid of it
     leftArr.remove(leftArr[0])
   else:
     comparisons+=1
     sortedList.append(rightArr[0])
     rightArr.remove(rightArr[0])
 # If the initial array was an odd array, one of the array
 # has one more element than the other
 if len(leftArr) == 0:
   sortedList += rightArr
 else:
   sortedList += leftArr
 return sortedList

def mergeSort(inputList):
 if len(inputList) <= 1:
   return inputList
 else:
   # Finding the mid point of the inputList
   midPoint = math.ceil(len(inputList) / 2)

   # This iterates from index 0 to midPoint because it's
   # the first half
   leftArray = mergeSort(inputList[0:midPoint])
   # This iterates from midpoint to the end of inputList
   # since it's the second half
   rightArray = mergeSort(inputList[midPoint:len(inputList)])
   return merge(leftArray, rightArray)
```

5 Linear Search

```python
def linearSearch(inputList):
 global LScomparisons
 noVal = False
 # Traverse through the array
 for search in inputList:
   if search == x:
     noVal = True
     LScomparisons+=1
     return noVal
   else:
     LScomparisons+=1
     noVal = False
 return noVal
```

6 Binary Search

```python
def binarySearch(inputList):
 global BScomparisons
 leftArray =[]
 rightArray=[]
 if len(inputList) == 1 and x != inputList[0]:
   return "ERROR"
 else:
   midPoint = math.ceil(len(inputList) / 2)
   if x == inputList[midPoint]:
     BScomparisons+=1
     return inputList[midPoint]
   elif x < inputList[midPoint]:
     BScomparisons+=1
     for search in range(0, midPoint):
       leftArray.append(inputList[search])
     return binarySearch(leftArray)
   else:
     BScomparisons+=1
     for search in range(midPoint, len(inputList)):
       rightArray.append(inputList[search])
     return binarySearch(rightArray)
```