Sorts algorithm

November 16, 2020

0.1 Alireza Soltani Neshan 98111033302016

0.2 Data Structure

0.2.1 Course code 23032

1 Selection Sorting

[15]:
```
def selectionSorting(arr):
    for i in range(len(arr)-1, 0, -1):
        max_i = 0
        max_v = 0
        for j in range(i):
            if arr[j] > max_v:
                max_v = arr[j]
                max_i = j
        if arr[i] < max_v:
            arr[i], arr[max_i] = arr[max_i], arr[i]

    return arr
```

[16]:
```
data = [11, 33, 55, 22, 92, 44]
```

[17]:
```
selectionSorting(data)
```

[17]:
```
[11, 22, 33, 44, 55, 92]
```

2 Bubble Sorting

[5]:
```
def bubbleSort (ls):
    for i in range (len(ls)):
        for j in range (len(ls)-1):
            if ls[j] > ls[j+1]:
                ls[j], ls[j+1] = ls[j+1], ls[j]
    return ls
```

[6]:
```
data_t_1 = [2, 0, 1, 5, 1, 2, 7]
data_t_2 = [40, 25, 58, 1, 25, 877, 51, 968, 11, 0, 1 ,-1, 5, -10]
data_test_b_3 = [3, 1, 7, 20, 2, 6]
```

[7]:
```
bubbleSort(data_t_2)
```

[7]:
```
[-10, -1, 0, 1, 1, 5, 11, 25, 25, 40, 51, 58, 877, 968]
```

[8]:
```
bubbleSort(data_test_b_3)
```

[8]:
```
[1, 2, 3, 6, 7, 20]
```

3 Insertion Sorting

[19]:
```
def insertionSort(ls):
    for i in range (len(ls)):
        for j in range(i, 0, -1):
            if ls[j] < ls[j-1]:
                ls[j], ls[j-1] = ls[j-1] , ls[j]
    return ls
```

[20]:
```
data_test_1 = [4, 3, 2, 10, 12, 7, 5, 6]
data_test_2 = [35, 51, 27, 85, 66, 23]
data_test_geeks_for_geeks_1 = [7, 8, 5, 2, 4, 6, 3]
data_test_geeks_for_geeks_2 = [4, 3, 2, 10, 12, 1, 5, 6]
```

[21]:
```
insertionSort(data_test_1)
```

[21]:
```
[2, 3, 4, 5, 6, 7, 10, 12]
```

[22]:
```
insertionSort(data_test_2)
```

[22]:
```
[23, 27, 35, 51, 66, 85]
```

[23]:
```
insertionSort(data_test_geeks_for_geeks_1)
```

[23]:
```
[2, 3, 4, 5, 6, 7, 8]
```

[24]:
```
insertionSort(data_test_geeks_for_geeks_2)
```

[24]:
```
[1, 2, 3, 4, 5, 6, 10, 12]
```

4 Quick Sort

[1]:
```
def quickSort(ls, low, high):
    if len(ls) == 1:
        return ls

    if low < high:
        pivot = partition(ls, low, high)
        quickSort(ls, low, pivot - 1)
        quickSort(ls, pivot + 1, high)
```

[2]:
```
def partition(ls, low, high):
    i = low - 1
    pivot = ls[high]

    for j in range(low, high):
        if ls[j] <= pivot:
            i += 1
            ls[i], ls[j] = ls[j], ls[i]
    ls[i + 1], ls[high] = ls[high], ls[i + 1]
    return i + 1
```

[3]:
```
data_for_quick_sort= [35, 51, 27, 85, 66, 23]
quickSort(data_for_quick_sort, 0, len(data_for_quick_sort)-1)

data_for_quick_sort
```

[3]:
```
[23, 27, 35, 51, 66, 85]
```