**Problem Set**

[](http://www.google.com.pk/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiJiqa_ieDQAhWH7BQKHXE7DUkQjRwIBw&url=http://uet.edu.pk/brainiac/2015/&psig=AFQjCNFMepkpIAh21A-LgUZttYfzys-ikQ&ust=1481131092118684)

***Harmain Iftikhar (2022-CS-07)***

Sir. Nazeef-ul-haq

Data Structures & Algorithms

Section: A

25-09-2023

**Department of Computer Science**

**University of Engineering and Technology, Lahore**

* **Bubble Sort**

**Description:** First of all why we call this algorithm bubble sort. Let’s assume when you throw a rock into water as you know rock is a heavy so rock will move down when rock reach the bottom water also start moving upward. So, in the same bubble sort in each pass the greatest or heaviest element will be sorted at the end. So after (n-1) passes smallest element will be at the start and greater element will afterwards. In each pass specific elements are being compared and then they swap suppose 12, 9 are being compared so the smaller greater will be at the end and our final result will be 9,12. In each pass specific no of comparison and swapping will be done. After (n-1) total array will be sorted out.

**Pseudo Code:**

**A = {9,8,7,6,4,3,2,1}**

**def Bubble\_sort(array)**

for i in range(start,end-1,+1):

for j in range(start,end):

if array[j] > array[j+1]:

temp = array[j]

array[j] = array[j+1]

array[j+1] = temp

Bubble\_sort(A)

**Working:**

After 1st pass Arary will be :{8,7,6,4,3,2,1,9}

After 2ndpass : {7,6,4,3,2,1,8,9}

After 3rd pass : {6,4,3,2,1,7,8,9}

After 4th pass : {4,3,2,1,6,7,8,9}

After 5th pass : {3,2,1,4,6,7,8,9}

After 6th pass : {2,1,3,4,6,7,8,9}

After 7th pass : {1,2,3,4,6,7,8,9}

So, after (n-1) passes means after 7 passes our list is sorted out.

**1st pass** :First 9 and 8 are being compared if 1st one is greater

9

8

7

6

4

3

2

1

Than 2nd one then swapping so after completing one pass we

Will get ans as

8

7

6

4

3

2

1

9

**4th pass**

**2nd passs : 3rd pass:**

3

2

1

4

6

7

8

9

4

3

2

1

6

7

8

9

6

4

3

2

1

7

8

9

7

6

4

3

2

1

8

9

**5th Pass 6th Pass 7th pass**

1

2

3

4

6

7

8

9

2

1

3

4

6

7

8

9

So final array is **A = {1,2,3,4,6,7,8,9}**

* **Insertion Sort:**

**Code :**

def InsertionSort(array,start, end):

for i in range(start+1,end,+1):

j = i - 1

x = array[i]

while array[j] > x and j >= start:

array[j+1] = array[j]

j -= 1

array[j+1] = x

Dry Running the Problem**:**

If there is only one element in that case that element is already sorted so in this sorting we will assume our first element is sorted that. In this we will perform (n-1) passes.

**A = {5,43,76,2,98,23,12,32}**

**1st pass:**

we willtake three variables i for outer loop that control passes and j that will compare the elements and x that will hold specific index element.

“i” will point towards 1 index and j = i-1= 0 and x = A[i] = 43

If A[j] > x means A[0] = 5 > 43 no,,, so no swapping will done but now A[j+1] = x = 43

Now assume list become = [5,43] that is sorted but complete list {**5,43,76,2,98,23,12,32**} ok

**2nd pass :**

**Now** i = 2, j = 1, x = A[2] = 76

A[1] > x ; 43 > 76 no,,, so no swapping but A[2]= x = 76

Sorted list = [5,43,76]

**3rd pass:**

Now i = 3, j = 2, x = 2

* 76>2 so now : A[3] = A[2] ;;; A[3] = 76
* But now j - - bcuz we are in inner loop so ,,, A[1] > x;;; 43 > 2 yes so A[2] = 43 but now
* J = 0 A[0] > 76 yes so A[1] = 5 but now j = -1

So inner loop break out.

After A[0] = x = 2

So list become {2,5,43,76,**98,23,12,36**}

**4th pass:**

**Now** i = 4, j= 3, x = 98

**A[3] >** 98 no,,,so list will remain {2,5,43,76,98**,23,12,36**}

**5th pass:**

i = 5, j = 4, x = 23

* A[4]> 23 yes so A[5] = 98 j = 3
* A[3]> 23 yes so A[4] = 76 j = 2
* A[2]> 23 yes so A[3] = 43 j = 1
* But now A[1] > 23 not true so loop ended

A[2] = 23,,, Final array = {2,5,23,43,76,98**,12,36**}

**6th pass:**

i = 6, j = 5,x = 12

* A[5]> 12 yes so A[6] = 98 j = 4
* A[4]> 12 yes so A[5] = 76 j = 3
* A[3]> 12 yes so A[4] = 43 j = 2
* A[2]> 12 yes so A[3] = 23 j = 1

So now A[2] = 12,,, Final array = {2,5,12,23,43,76,98**,36**}

**7th pass:**

i = 7,j = 6, x= 36

* A[6]> 36 yes so A[7] = 98 j = 5
* A[5]> 36 yes so A[6] = 76 j = 4
* A[4]> 36 yes so A[5] = 43 j = 3

Now terminate inner loop

So now A[4] = 36,,, Final array = {2,5,12,23,36,43,76,98}

**Final Array = {2,5,12,23,36,43,76,98}**

**-----------------------------------------------------------------------------------------**

* **Selection Sort:**In each pass or iteration one element will be sorted out. (n-1) passes required to sort specific list or array. In this we find out an element for that specific position. Suppose for first index we need a smallest element that is present in list.

In this sorting we use three variables suppose i,j and k. At fisrt iteration i, j and k will at zero index.Then j will move towards next index and if j index is less than k index k will be point where j is pointing. So after one pass j will be out of the list and wherever k is pointing i index value will be swap to k index value and then i,j and k will be point towards next index. That’s how it should work.

Analysis:

Step : 1

Minimum value at step : 1 is 5

array at Step : 1 is [5, 43, 76, 2, 98, 23, 12, 32]

array at Step : 1 is [5, 43, 76, 2, 98, 23, 12, 32]

array at Step : 1 is [2, 43, 76, 5, 98, 23, 12, 32]

array at Step : 1 is [2, 43, 76, 5, 98, 23, 12, 32]

array at Step : 1 is [2, 43, 76, 5, 98, 23, 12, 32]

array at Step : 1 is [2, 43, 76, 5, 98, 23, 12, 32]

array at Step : 1 is [2, 43, 76, 5, 98, 23, 12, 32]

Step : 2

Minimum value at step : 2 is 43

array at Step : 2 is [2, 43, 76, 5, 98, 23, 12, 32]

array at Step : 2 is [2, 5, 76, 43, 98, 23, 12, 32]

array at Step : 2 is [2, 5, 76, 43, 98, 23, 12, 32]

array at Step : 2 is [2, 5, 76, 43, 98, 23, 12, 32]

array at Step : 2 is [2, 5, 76, 43, 98, 23, 12, 32]

array at Step : 2 is [2, 5, 76, 43, 98, 23, 12, 32]

Step : 3

Minimum value at step : 3 is 76

array at Step : 3 is [2, 5, 43, 76, 98, 23, 12, 32]

array at Step : 3 is [2, 5, 43, 76, 98, 23, 12, 32]

array at Step : 3 is [2, 5, 23, 76, 98, 43, 12, 32]

array at Step : 3 is [2, 5, 12, 76, 98, 43, 23, 32]

array at Step : 3 is [2, 5, 12, 76, 98, 43, 23, 32]

Step : 4

Minimum value at step : 4 is 76

array at Step : 4 is [2, 5, 12, 76, 98, 43, 23, 32]

array at Step : 4 is [2, 5, 12, 43, 98, 76, 23, 32]

array at Step : 4 is [2, 5, 12, 23, 98, 76, 43, 32]

array at Step : 4 is [2, 5, 12, 23, 98, 76, 43, 32]

Step : 5

Minimum value at step : 5 is 98

array at Step : 5 is [2, 5, 12, 23, 76, 98, 43, 32]

array at Step : 5 is [2, 5, 12, 23, 43, 98, 76, 32]

array at Step : 5 is [2, 5, 12, 23, 32, 98, 76, 43]

Step : 6

Minimum value at step : 6 is 98

array at Step : 6 is [2, 5, 12, 23, 32, 76, 98, 43]

array at Step : 6 is [2, 5, 12, 23, 32, 43, 98, 76]

Step : 7

Minimum value at step : 7 is 98

array at Step : 7 is [2, 5, 12, 23, 32, 43, 76, 98]

**Sorted Array by Selection Sort is** : [2, 5, 12, 23, 32, 43, 76, 98]

Time Complexity

**Loop Invariant of Selection Sort:**

* Initialization (Before the loop)

The variable i initialize to 1, it will execute only one time after that only condition and increment value is considered. At beginning the unsorted part contains elements and sorted part is empty. The loop invariant here is that the left part (sorted) is empty and right portion (unsorted) contains all elements.

* Maintenance

When the loop starts, the first element is selected and compared with all the rest elements, the element which is too smaller than it will be swapped by index. The sorted values lie in the left side and unsorted lies in the right side. This process continues until the last element is sort.

* Termination

When the outer loop condition is encountered the loop terminates, it means the array is sorted. The condition depends either we sort the elements in ascending order or descending order.

**--------------**

* **Merge Sort:**

**Pesudo Code :**

**Def** Merge\_sort(A,L,H)

**If** L < H

Mid = (L+H)/2

MergeSort(A,L,Mid)

MergeSort(A,Mid+1,H)

Merge (A,L,Mid,H)

It is a two way merge sort. It follow the top down approach.

Low Mid High

0 4 9

A= {1,3,54,2,72,23,12,32,76,12}

Merge will divide list in two then merge them together in sorted position.

In recursion, function will call itself until low < high

0,9

**Recusrion(**Low🡪Mid)

8,9

5,9

0,4

9,9

8,8

6,6

5,5

4,4

3,4

3,3

0,1

0,1

7,7

5,6

5,7

2,2

0,1

0,2

So by doing this our final result will be :

{1,**2,3,12,12,23,32,54,72,76**}

**5th part**:

**Pseudo Code:**

**def merge(A, low, mid, high):**

**print(A)**

**B = []**

**i = low**

**j = mid + 1**

**while i <= mid and j <= high:**

**if A[i] < A[j]:**

**B.append(A[j])**

**i += 1**

**else:**

**B.append(A[i])**

**j += 1**

**while i <= mid:**

**B.append(A[i])**

**i += 1**

**while j <= high:**

**B.append(A[j])**

**j += 1**

**for k in range(low, high + 1):**

**A[k] = B[k-low]**

**def merge\_sort(A,l,h):**

**if l < h:**

**mid = int((l+h)/2)**

**merge\_sort(A, l, mid)**

**merge\_sort(A, mid+1, h)**

**merge(A, l, mid, h)**

**array = RandomArray(10000)**