Tutorial & Laboratory

Programming & Data Structure: CS11001/19001

Section - 4/D

DO NOT POWER ON THE MACHINE

Department of Computer Science and Engineering I.I.T. Kharagpur

Spring Semester: 2013 - 2014 (**27.02.2014**)

Download

Download the file date270214.pdf from Programming & Data Structures ... of

http://cse.iitkgp.ac.in/~goutam

View the file using the command acroread & or xpdf &

Sorting Problem

Unsorted Data									
7.5	8.0	8.5	8.25	9.25	9.0	6.5	8.0	7.0	7.5

Sorted Data (non-ascending)									
9.25	9.0	8.5	8.25	8.0	8.0	7.5	7.5	7.0	6.5

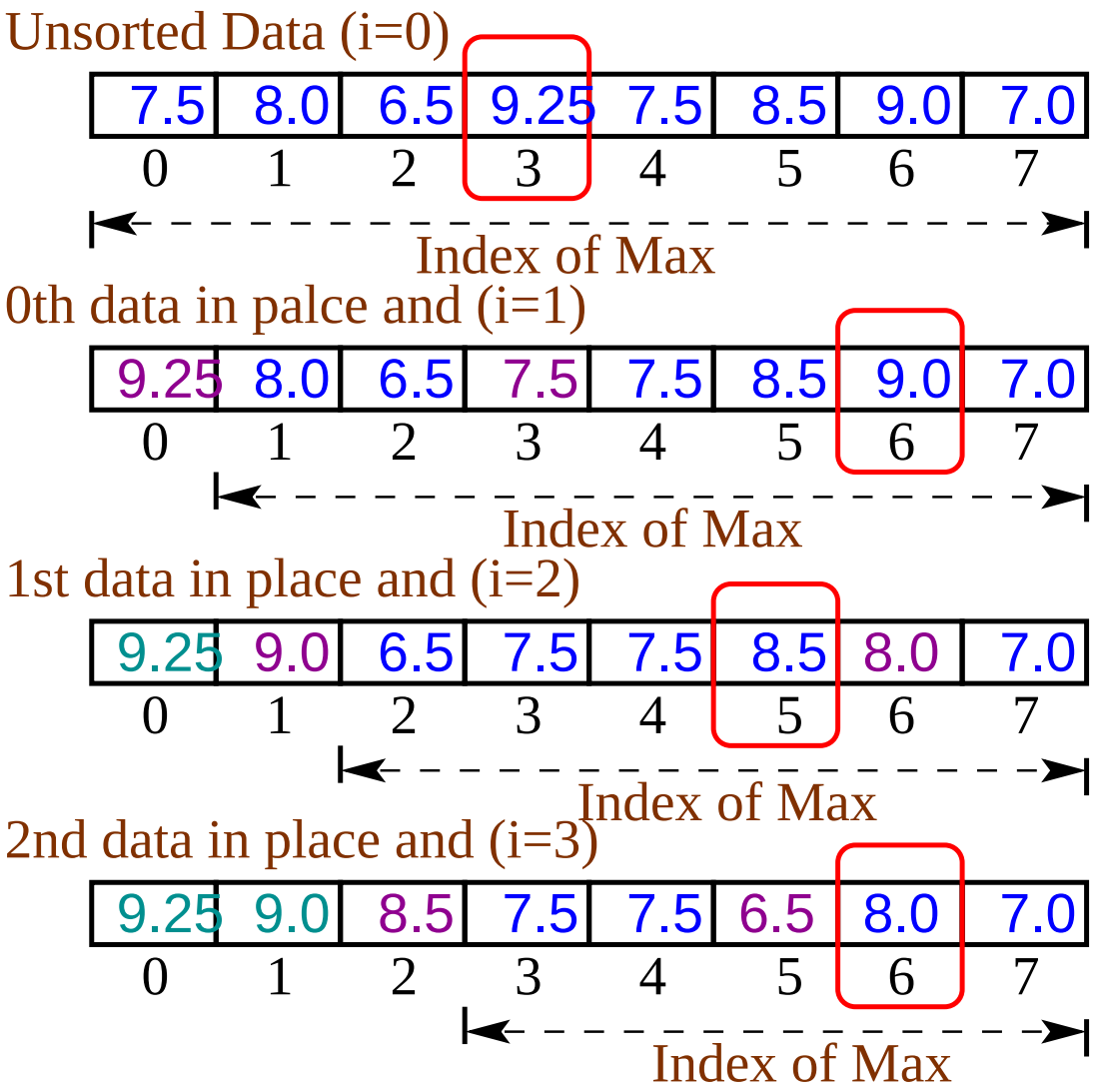
Sorted Data (non-descending)									
6.5	7.0	7.5	7.5	8.0	8.0	8.25	8.5	9.0	9.25

Selection Sort

The data is stored in an 1-D array and we sort them in non-ascending order. Let the number of data be n

Selection Sort Algorithm

```
for i ← 0 to n − 2 do
    max ← a[i], maxIndex ← i
    for j ← i + 1 to n − 1 do
        if max < a[j] then
            max ← a[j], maxIndex ← j
        endIf
    endFor
    a[i] ↔ a[maxIndex]      #Exchange
endFor
```

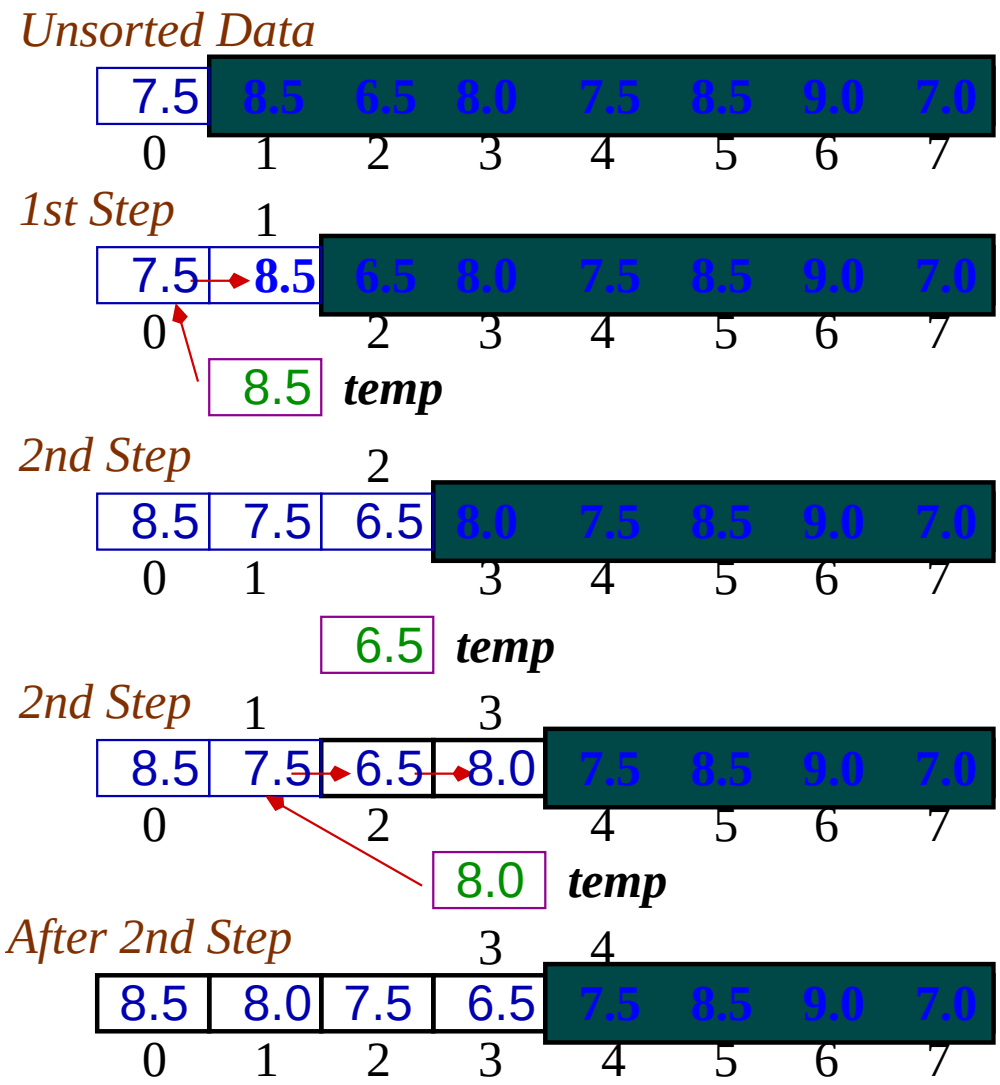

Unsorted Data (i=0)
7.5 8.0 6.5 9.25 7.5 8.5 9.0 7.0
0 1 2 3 4 5 6 7
Index of Max
0th data in palce and (i=1)
9.25 8.0 6.5 7.5 7.5 8.5 9.0 7.0
0 1 2 3 4 5 6 7
Index of Max
1st data in place and (i=2)
9.25 9.0 6.5 7.5 7.5 8.5 8.0 7.0
0 1 2 3 4 5 6 7
Index of Max
2nd data in place and (i=3)
9.25 9.0 8.5 7.5 7.5 6.5 8.0 7.0
0 1 2 3 4 5 6 7
Index of Max

Insertion Sort

The data is stored in an 1-D array and we sort them in non-ascending order. Let the number of data be n

Insertion Sort Algorithm

```
for i ← 1 to n −1 do
    temp ← data[i]
    for j ← i-1 downto 0 do
        if data[j] < temp
            data[j+1] ← data[j]
        else go out of loop
    endFor
    data[j+1] ← temp
endFor
```


Unsorted Data
7.5 8.5 6.5 8.0 7.5 8.5 9.0 7.0
0 1 2 3 4 5 6 7
1st Step
1
7.5 8.5 6.5 8.0 7.5 8.5 9.0 7.0
0 2 3 4 5 6 7
8.5 temp
2nd Step
2
8.5 7.5 6.5 8.0 7.5 8.5 9.0 7.0
0 1 3 4 5 6 7
6.5 temp
2nd Step
1 3
8.5 7.5 6.5 8.0 7.5 8.5 9.0 7.0
0 2 4 5 6 7
8.0 temp
After 2nd Step
3 4
8.5 8.0 7.5 6.5 7.5 8.5 9.0 7.0
0 1 2 3 4 5 6 7

Assignment VIII

Write a C program that reads two sets of data, sorts the first set in non-ascending (larger to smaller) and the second set in non-descending (smaller to larger) order. Then it merges the sorted data in non-descending order. [Marks: 25]

Task 1

In the function `main()` read two positive integers m and n, then read two sets of floating-point data in two 1-D array of type `double`. The first set has m and the second set has n elements. Print the input data after reading. [Marks: 5]

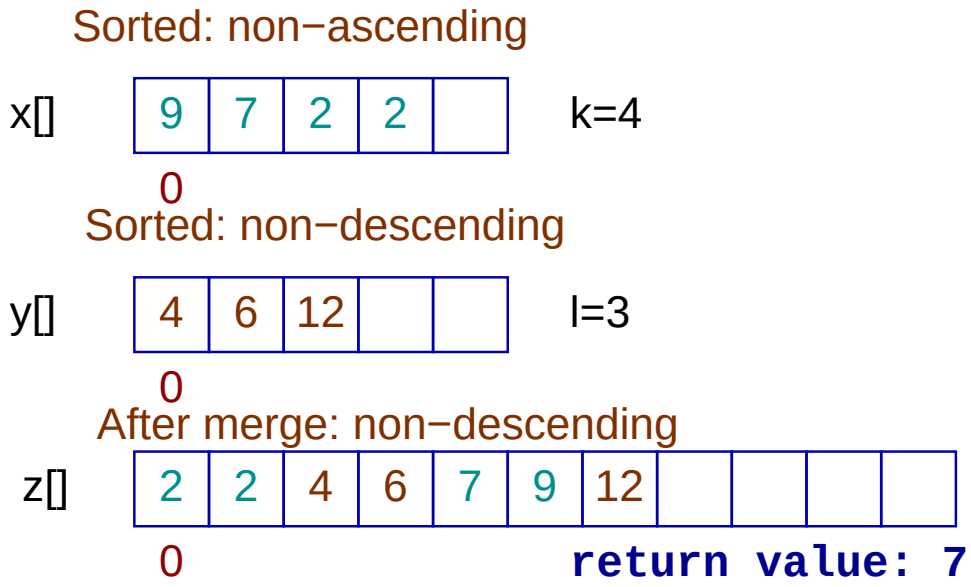
Task 2

From `main()`, call the function
void selSort(double x[], int k) to sort the first set of data in *non-ascending* order (larger to smaller).
Then call the function
void insSort(double y[], int l) to sort the second set of data in *non-descending* order.
Print two sets of sorted data.

Task 2 (cont.)

Now call the function
`int merge(double x[], int m, double y[], int n, double z[])`,
to merge the sorted data to a new array `z[]` in non-descending order.
The function `merge()` returns the total number of data merged in `z[]`.
Finally print the merged data. [Marks: 5]

Merge


Sorted: non−ascending
x[]
9
7
2
2
k=4
0
Sorted: non−descending
y[]
4
6
12
l=3
0
After merge: non−descending
z[]
2
2
4
6
7
9
12
0
return value: 7

Task 3

Use the given *selection sort* and *insertion sort* algorithms to write the functions `selSort()` and `insSort()`. [Marks: 5+5]

Task 4

The function
`int merge(double x[], int k, double y[], int l, double z[])`,
takes five parameters and returns a value. The first parameter `x[]` points to a non-ascending sorted array of `k` data. The third parameter `y[]` points to a non-descending sorted array of `l` data.

Task 4 (cont.)

The fifth parameter points to a non-descending sorted array of $k + l$ data after merging.
The function returns the total number of data it has merged. [Marks: 5]

Input/Output

Input: 3 5

5 1 7

8 2 5 1 9

Output: Input: 5.00 1.00 7.00

and : 8.00 2.00 5.00 1.00 9.00

Sorted: 7.00 5.00 1.00

and : 1.00 2.00 5.00 8.00 9.00

Merged: 1.00 1.00 2.00 5.00 5.00 7.00 8.00 9.00

Submission by ftp

```
$ ftp 10.5.17.186
Connected to 10.5.17.186.
220---------- Welcome to Pure-FTPd ----
220-You are user number 1 of 50 allowed.
220-Local time is now 07:54. ....... 21.
220-IPv6 connections ........
220 ... disconnected .. inactivity.
Name (10.5.17.186:...): pds
```

Submission by ftp

```
331 User pds OK. Password required
Password: pds04
230-User pds has group access to:  pds
230 OK. Current restricted directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd assignment8
250 OK. Current directory is /assignment8
```

Submission by ftp

```
ftp> put D0608.c
local: D0608.c remote: D0608.c
200 PORT command successful
150 Connecting to port 47093
226-File successfully transferred
226 0.001 seconds .. 39.00 Kbytes ..
27 bytes sent in 0.00 secs (1098.6 kB/s)
ftp> bye
21-Goodbye. .....
221 Logout.
$
```

Assignment IX

Write a C program that reads `n` number of positive integers in a 1-D array. It uses two recursive functions `gcd()` and `countRelPrimes()`. The first one computes *gcd* of two positive integers. The second function takes an 1-D array and other parameters. It returns the number of pairs of data in the 1-D array that are relatively prime.

Assignment IX

You are not allowed to use any global variable, static variable or any loop construct in these two functions. [Marks: 05]

Input-Output

Input: 5
Input: 24 12 13 25 51
Output: 7

Relatively prime pairs are
$(24, 13)(24, 25)(12, 13)(12, 25)(13, 25)(13, 51)(25, 51)$