*Heaven's Light is Our Guide*

**Computer Science & Engineering**
**Rajshahi University of Engineering & Technology**

# Lab Manual

## Module- 09

**Course Title:** Sessional based on CSE 1201
**Course No.** : CSE 1202

**Experiment No.** 9

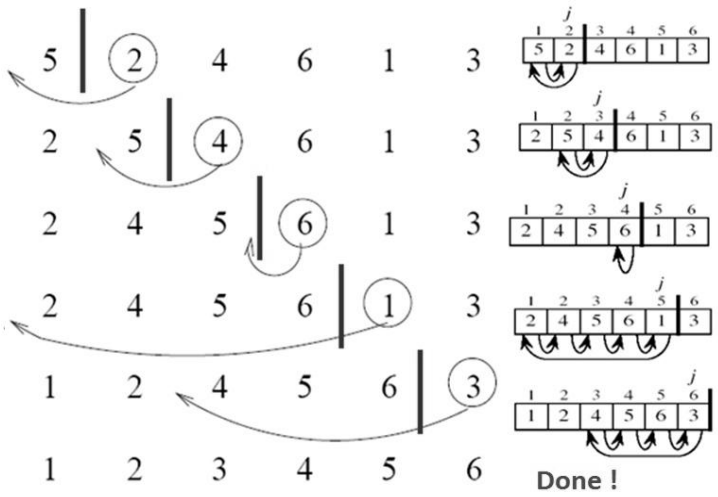
**Name of the Experiment:** Sorting and Searching

**Duration:** 1 cycle

**Background Study:** Chapter 9 (Theory and Problems of Data Structures Written by Seymour Lipschutz)

[Contain:
Insertion Sort & its Complexity
Selection Sort & its Complexity
Merge Sort & its Complexity
Radix Sort & its Complexity
]

**9.1 Insertion Sort**



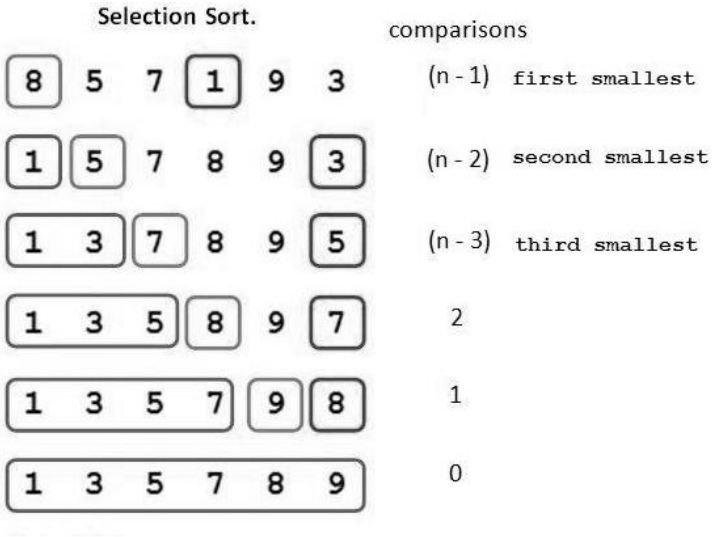

1. Select TEMP ← A[j]
2. Shift Data, and copy TEMP to proper place

**Problem I:** Insertion Sort
**Algorithm 9.1: INSERTION(A, N)**
This algorithm sorts the array A with N elements

1. Set A[0]:=-∝
2. Repeat Steps 3 to 5 for K = 2,3, ..... , N:
3. Set TEMP:=A[K] and PTR:=K-1
4. Repeat while TEMP<A[PTR]:
5. (a) Set A[PTR+1]:=A[PTR].
6. (b) Set PTR:=PTR-1.
7. [End of loop]
8. Set A[PTR+1]:=TEMP.
9. [End of Step 2]
10. Return.

**Flow Chart:** Draw a flow chart.
**Complexity of Insertion Sort:**
Worst Case: $O(n^2)$
Average Case: $O(n^2)$

## 9.2 Selection Sort




Total comparisons = n (n - 1)/2

~ O ($n^2$)

1. Find Min from A[2:N], and Interchange with First Element.
2. Find Min from A[3:N], and Interchange with second Element.
3. So on.

**Problem II:** Selection Sort
**Algorithm 9.2(a): GETMIN(A,K,N,LOC)**
An array A is in memory. The procedure finds the location LOC of the smallest element among A[K], a[K+1],..., A[N].

1. Set MIN:=A[K] and LOC:=K.
2. Repeat for J=K+1 to N:
   If MIN>A[J], then:
   Set MIN:=A[J] and LOC:=J.
   [End of If statement]
   [End of loop]
3. Return.

**Algorithm 9.2(a): SELECTION(A,N)**
This algorithm sorts the array A with N elements

1. Repeat Steps 2 and 3 for K=1,2, ....., N-1
2. **CALL GETMIN(A,K,N,LOC).**
3. Set TEMP:=A[K], A[K]:=A[LOC] and A[LOC]:=TEMP.
4. [End of Step 1 loop]
5. Exit

**Flow Chart:** Draw a flow chart.
**Complexity of Selection Sort:**
Worst Case: $O(n^2)$

Average Case: $O(n^2)$

### 9.3 Merge Sort

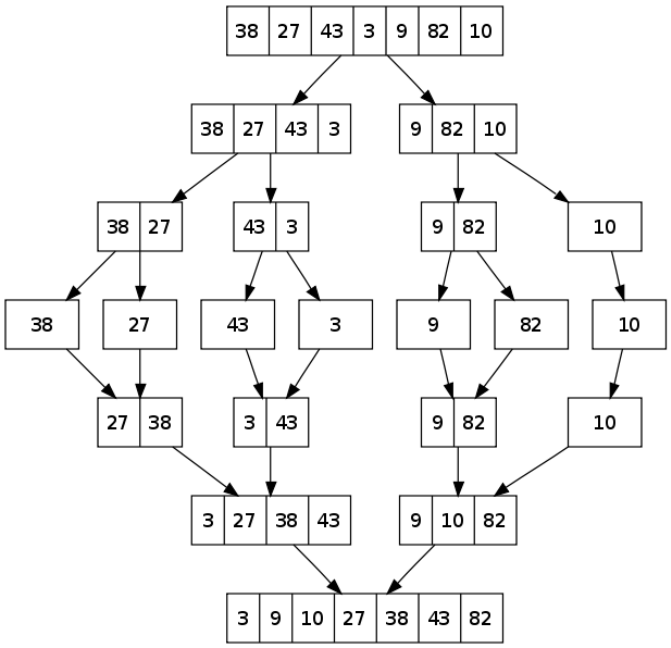



Fig. Merge Sort

**Problem III:** Merge Sort

**Algorithm 9.3(a): MERGE(A,R,LBA,B,S,LBB,C,LBC)**

Let A be a sorted array with R elements and lower bound LBA, B be a sorted array with S elements and lower bound LBA and C has lower bound LBC. The procedure merges A and B into an array C with R+S elements.

1. Set NA:=LBA, NB:=LBB, PTR:=LBC, UBA:=LBA+R-1, UBB:=LBB+S-1.
2. Repeat while NA≤UBA and NB≤UBB
   If A[NA]<B[NB], then:
   (a) Set C[PTR]:=A[NA].
   (b) Set PTR:=PTR+1 and NA:=NA+1.
   Else:
   (a) Set C[PTR]:=B[NB].
   (b) Set PTR:=PTR+1 and NB:=NB+1.
   [End of If Statement]
   [End of loop]
3. If NA>UBA, then:
   Repeat for K=0,1,2,....., UBB-NB
   Set C[PTR+K]:=B[NB+K].
   [End of loop]
   Else:
   Repeat for K=0,1,2,....., UBA-NA
   Set C[PTR+K]:=A[NA+K].
   [End of loop]
   [End of If statement]
4. Exit.

**Algorithm 9.3(b): MERGEPASS(A,N,L,B)**

The N elements array A is composed of sorted subarrays where each subarray has L elements except possibly the last subarray, which may have fewer than L elements. The procedure merges the pairs of subarrays of A and assigns them to the array B.

1. Q:=INT(N/(2*L)), S:=2*L*Q and R:=N-S.
2. Repeat for J=1,2,....., Q:
   (a) Set LB:= 1+)2*J-2)*L.
   (b) **Call MERGE(A,L,LB,A,L,LB+L,B,LB)**

[End of Loop]
3. If R≤L, then:
   Repeat for J=1,2, ....., R:
   Set B(S+J):=A(S+J).
   [End of loop]
   Else:
   **Call MERGE(A,L,S+1,A,R,L+S+1,B,S+1).**
   [End of If statement]
4. Return.

**Algorithm 9.3(c): MERGESORT(A,N)**
The algorithm sorts the array A with N elements using an auxiliary array B
1. Set L:=1
2. Repeat Steps 3 to 5 while L<N:
3. **Call MERGEPASS(A,N,L,B).**
4. **Call MERGEPASS(B,N,2*L,A).**
5. Set L:=4*L.
   [End of Step 2 loop]
6. Exit.

**Flow Chart:** Draw a flow chart.
**Complexity:**
Worst Case: O(nlogn)
Average Case: O(nlogn)
Extra Memory: O(n).

**9.4 Radix Sort**

LSD Radix Sorting:
Sort by the last digit, then by the middle and the first one

| | | | |
|---|---|---|---|
| 362 | 291 | 207 | 207 |
| 436 | 362 | 436 | 253 |
| 291 | 253 | 253 | 291 |
| 487 | 436 | 362 | 362 |
| 207 | 487 | 487 | 397 |
| 253 | 207 | 291 | 436 |
| 397 | 397 | 397 | 487 |

MSD Radix Sorting:
Sort by the first digit, then sort each of the groups by the next digit

| | | | |
|---|---|---|---|
| 237 | 237 | 216 | 211 |
| 318 | 216 | 211 | 216 |
| 216 | 211 | 237 | 237 |
| 462 | 268 | 268 | 268 |
| 211 | 318 | 318 | 318 |
| 268 | 462 | 462 | 460 |
| 460 | 460 | 460 | 462 |

**A:**

| 62 | 36 | 91 | 87 | 53 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**d = 10,**

**PASS-I:**

**STEP-1: Initialize**
**COUNT:**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 2: Consider LSB of each Number. (2,6,1,7,3,7)**
**COUNT:**

| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 2 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 3: Now decide how to store the number**
**COUNT:**

| 0 | 1 | 2 | 3 | 3 | 3 | 4 | 6 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 4:**
**Now consider 97, LSB = 7 , Let Auxiliary array B**
**COUNT**:

| 0 | 1 | 2 | 3 | 3 | 3 | 4 | **5**(6-1) | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**B:**

| | | | | | **97** |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 53, LSB = 3**
COUNT:

| 0 | 1 | 2 | **2(3-1)** | 3 | 3 | 4 | 5 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| | | **53** | | | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 87, LSB = 7**
COUNT:

| 0 | 1 | 2 | 2 | 3 | 3 | 4 | **4(5-1)** | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| | | 53 | | **87** | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 91, LSB = 1**
COUNT:

| 0 | **0** | 2 | 2 | 3 | 3 | 4 | 5 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| **91** | | 53 | | 87 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 36, LSB = 6**
COUNT:

| 0 | 0 | 2 | 2 | 3 | 3 | **3** | 5 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| 91 | | 53 | **36** | 87 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 62, LSB = 2**
COUNT:

| 0 | 0 | 2 | **1** | 3 | 3 | 3 | 5 | 6 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| 91 | **62** | 53 | 36 | 87 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**STEP 5: Copy from B to A:**
A:

| 91 | 62 | 53 | 36 | 87 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**PASS-II:**

**STEP1: Again, Initialize**
**COUNT:**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 2: Consider Next DIGIT of each Number. (6,3,9,8,5,9)**
**COUNT:**

| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 2 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 3: Now decide how to store the number**
**COUNT:**

| 0 | 0 | 0 | 1 | 1 | 2 | 3 | 3 | 4 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**STEP 4:**
**Now consider 97, DIGIT = 9 , Let Auxiliary array B**
**COUNT**:

| 0 | 0 | 0 | 1 | 1 | 2 | 3 | 3 | 4 | **5** |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

**B:**

| | | | | | **97** |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 87, LSB = 8**
COUNT:

| 0 | 0 | 0 | 1 | 1 | 2 | 3 | 3 | **3** | 5 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| | | | **87** | | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 36, DIGIT = 3**
COUNT:

| 0 | 0 | 0 | **0** | 1 | 2 | 3 | 3 | 3 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| **36** | | | 87 | | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 53, DIGIT = 5**
COUNT:

| 0 | 0 | 0 | 0 | 1 | **1** | 3 | 3 | 3 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| 36 | **53** | | 87 | | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 62, DIGIT = 6**
COUNT:

| 0 | 0 | 0 | 0 | 1 | 1 | **2** | 3 | 3 | 5 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| 36 | 53 | **62** | 87 | | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Now consider 91, DIGIT = 9**

COUNT:

| 0 | 0 | 0 | 0 | 1 | 1 | 2 | 3 | 3 | **4** |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

B:

| 36 | 53 | 62 | 87 | **91** | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**STEP 5: Copy from B to A:**

A:

| 36 | 53 | 62 | 87 | 91 | 97 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

**Problem IV:** Radix Sort to sort a set of integer number

**Algorithm 9.4(a): GETMAX(A,N,MAX)**

An array A is in memory. The procedure returns largest element among A[1], a[2],..., A[N].

1. Set MAX:=A[1].
2. Repeat for J=2 to N:
   If MAX<A[J], then:
   Set MAX:=A[J].
   [End of If statement]
   [End of loop]
3. Return.

**Algorithm 9.4(b): COUNTSORT(A,N.EXP)**

EXP is used to find the digit of each integer number. This procedure sorts the integer array A according to the digit represented by EXP using an auxiliary array B.

1. Repeat for J=0,1, ....., 9:
   Set COUNT(J):=0.
   [End of loop]
2. Repeat for J=1,2, ....., N:
   Set COUNT((A[j]/EXP)%10):= COUNT((A[j]/EXP)%10)+1.
   [End of loop]
3. Repeat for J=1,2, ....., 9:
   Set COUNT(J):= COUNT(J) + COUNT(J-1).
   [End of loop]
4. Repeat for J=N,N-1, ....., 1:
   Set COUNT((A[j]/EXP)%10):= COUNT((A[j]/EXP)%10)-1.
   Set B(COUNT((A[j]/EXP)%10)):=A[J].
   [End of loop]
5. Repeat for J=1,2, ....., N:
   Set A(J):=B(J).
   [End of loop]
6. Exit.

**Algorithm 9.4(c): RADIXSORT(A,N)**

The algorithm sorts the array A with N elements.

1. **Call GETMAX(A,N,MAX)**.
2. Set EXP:=1.
3. Repeat Steps 4 and 5 while (MAX/EXP)>0
4. **Call COUNTSORT(A,N.EXP).**
5. EXP:=EXP*10.
   [End of Step 3 loop]
6. Exit.

**Flow Chart:** Draw a flow chart.

**Complexity:**
Let d be the radix, s be the total pass (i.e. max length of items) and n be the total number of items into a list.
Hence the number of comparisons for this algorithm is bounded as follows:

$$C(n) \leq d*s*n$$

d is constant for a specific case (e.g. d =10 for decimal digits) so d is independent of n.
In worst case, s = n , then $O(n^2)$
In best case, s = $\log_d n$, then O(nlogn).

**Exercise:**

**[1] Supplementary Problem: Searching, Hashing**

## MORE PROBLEMS

1. Programming (or supplementary) Problems of Chapter 9 of "Data Structures" by Seymour Lipschutz.

**LAB REPORT:** You have to submit all assigned problems in next lab.