Tasnimul Hasan Samit

18.02.04.142

Data Structure

02-02-2020

Text Book

* ① Schaum's outline

Data Structure

– Seymour Lipschutz

② Data Structure

– Edward M. Reingold

– Willfred J. Hansen

04-02-20

Sorting

Bubble Sort :

1. Repeat steps 2 to 3 for K = 1 to N-1.
2. set Ptr : = 1
3. Repeat while Ptr ≤ N-K

   ⓐ If Data [Ptr] > Data [Ptr+1] then

   Interchange Data [Ptr] and Data [Ptr+1]

   ⓑ set Ptr : = Ptr + 1

4. Exit

32, 51, 27, 85, 66, 23, 13, 57

27 51 66 85

23 85

13 85

57 85

32 27 51 66 23 13 57 (85)

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

(5) (4) 3 2 1

4 (5) (3) 2 1

4 3 (5) (2) 1

4 3 2 (5) (1)

4 3 2 1 5

Complexity Analysis :

① Comparision

② Interchange

③ Assignment

$$(n-1) + (n-2) + \_\_\_ + 2 + 1$$

$$= \frac{n(n-1)}{2}$$

$$= \frac{n^2}{2} - \frac{n}{2}$$

* Time

$$= O(n^2)$$

* Space

Selection sort :

Selection (A, N)

1. Repeat step 2 and 3 for $K = 1, 2 - - -, N-1$

2. Call Min (A, K, N, LOC)

3. Interchange A [K] and A [LOC]

    set Temp: = A [K]

    A [K] = A [LOC]

    A [LOC] = Temp

4. Exit

Min : (A, K, N, LOC)

1. set Min : = A [K] and LOC : = K

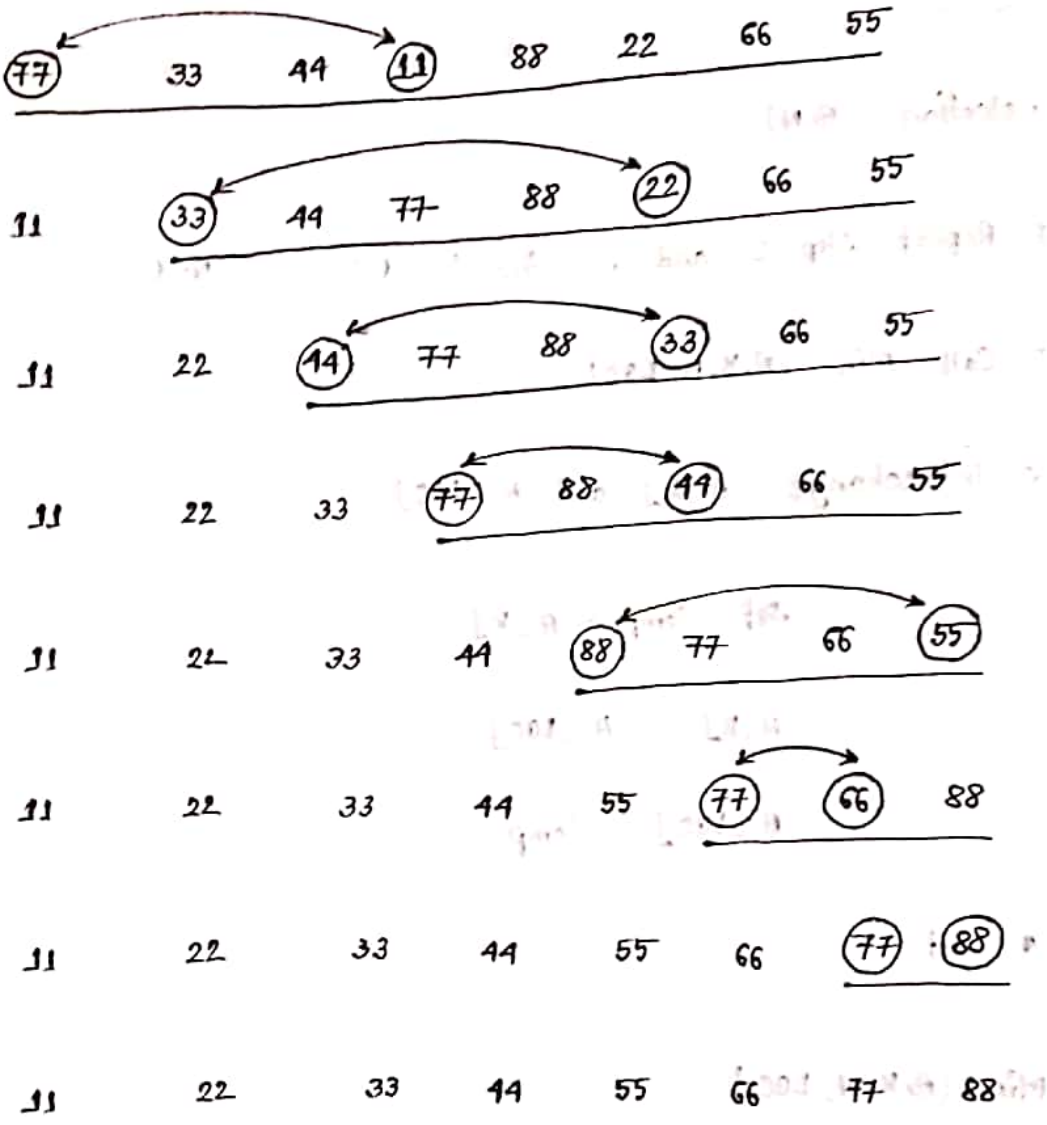
2. Repeat for $j = K+1, K+2 - - - + N$,

    If Min > A [j] then

    set Min : = A [j]

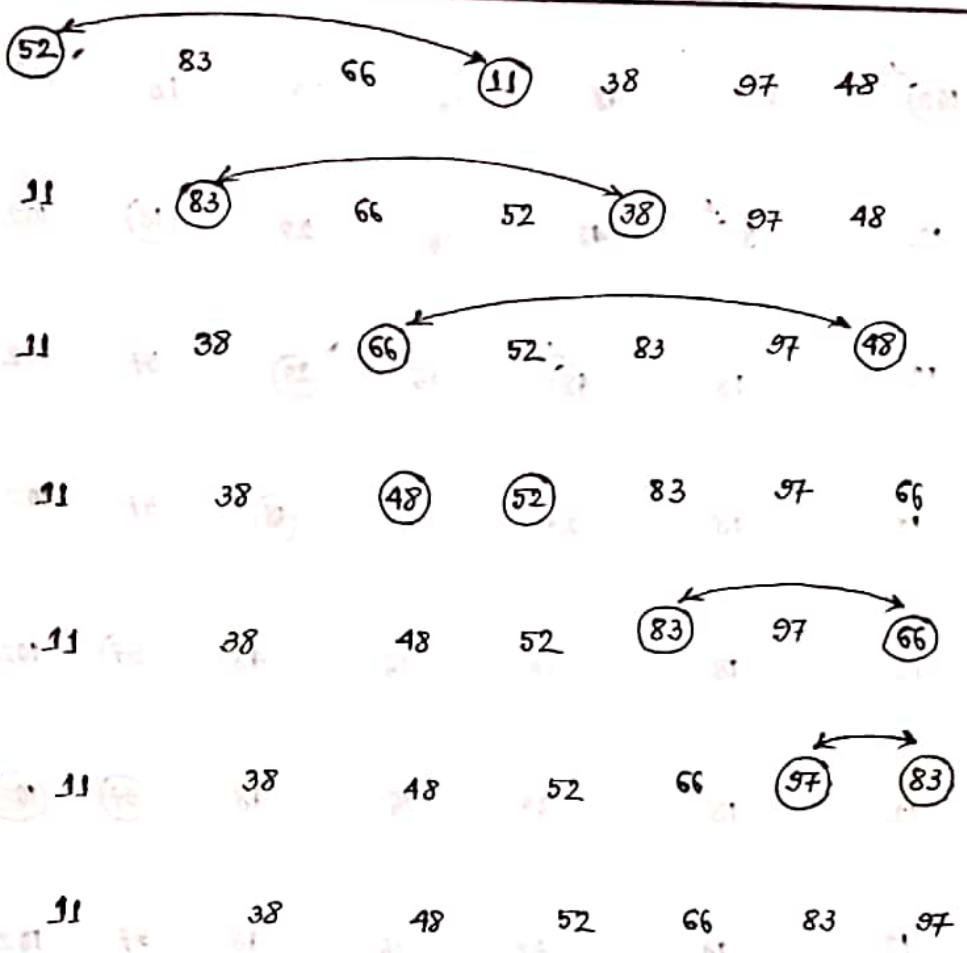
    LOC : = j

3. Return

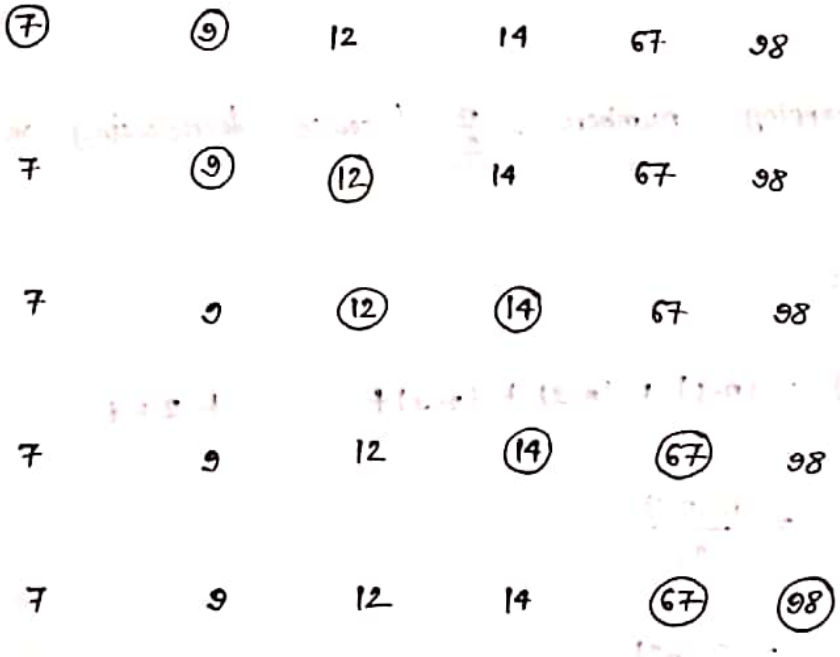


selection sort এটি এটি করে select করে তাকে right position এ নিয়ে আসে।

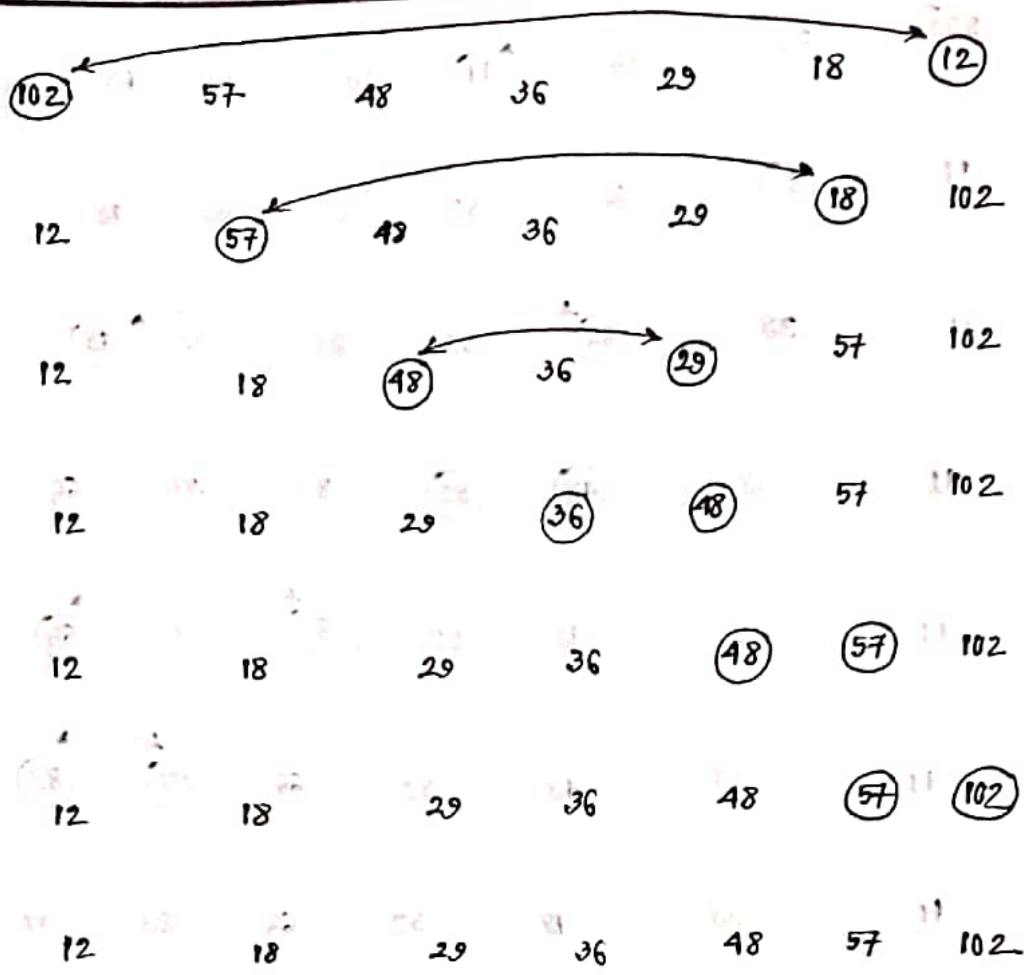
i)



ii)



iii)



swapping numbers $= \frac{n}{2}$ (cause decreasing order → worst)

Complexity:

$$f(n) = (n-1) + (n-2) + (n-3) + \cdots + 2 + 1$$

$$= \frac{n(n-1)}{2}$$

$$= O(n^2)$$

Insertion sort :

1. Set A[0] = $-\infty$ [Initialize sentinel element]

2. Repeat step 3 to 5 for K = 2, 3, ——— , N

3. Set Temp := A[K] and Ptr := K-1

4. Repeat while Temp < A[Ptr]

   (a) Set A[Ptr+1] = A[Ptr]

   [Move elements forward]

   (b) Set Ptr := Ptr-1

5. Set A[Ptr+1] := Temp

   [Inserts element in proper place]

6. Exit

$-\infty$ 77 33 44 11 88 22 66 55

$-\infty$ 77 (33) 44 11 88 22 66 55

$-\infty$ [33] 77 (44) 11 88 22 66 55

$-\infty$ 33 [44] 77 (11) 88 22 66 55

$-\infty$ [11] 33 44 77 (88) 22 66 55

$-\infty$ 11 33 44 77 88 (22) 66 55

$-\infty$ 11 [22] 33 44 77 88 66 55

$-\infty$ 11 22 33 44 77 88 (66) 55

$-\infty$ 11 22 33 44 [66] 77 88 (55)

$-\infty$ 11 22 33 44 [55] 66 77 88

Complexity :

$$1+2+3+ \_\_\_\_ + (n-1)$$

$$= \frac{n(n-1)}{2}$$

$$= O(n^2)$$

* Big data list থাকলে insertion sort use করা হয় না।

Merge sort (A, P, r)

1. If $p < r$
2. $q = (p+r)/2$
3. Merge-Sort (A, P, q)
4. Merge-Sort (A, q+1, r)
5. Merge (A, P, q, r)

Merge (A, P, q, r)

1. $n_1 = q - p + 1$
2. $n_2 = r - q$
3. L [1 ... n_1+1] and R [1 ... n_2+1] be new arrays
4. for $i = 1$ to n_1

   $L[i] = A[p+i-1]$

5. for $j = 1$ to n_2

   $R[j] = A[q+j]$

6. $L[n_1+1] = \infty$ and $R[n_2+1] = \infty$
7. $i = 1$ and $j = 1$

8. for K = P to R

if L[i] ≤ R[j]

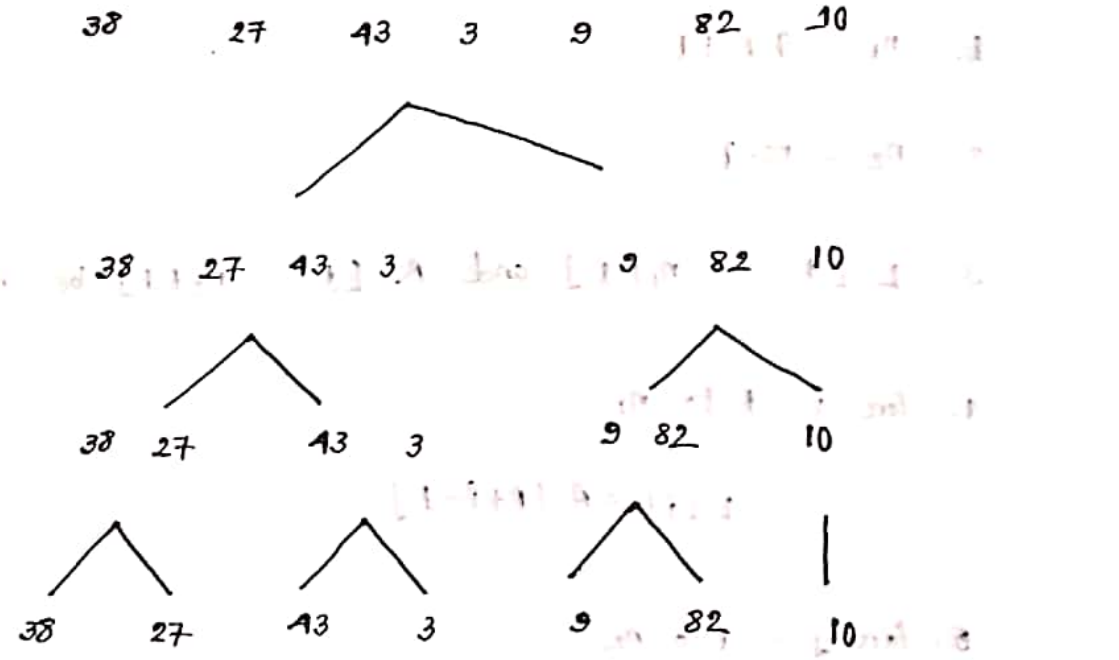
A[K] = L[i]

i = i+1

else A[K] = R[j]

j = j+1

9. Return

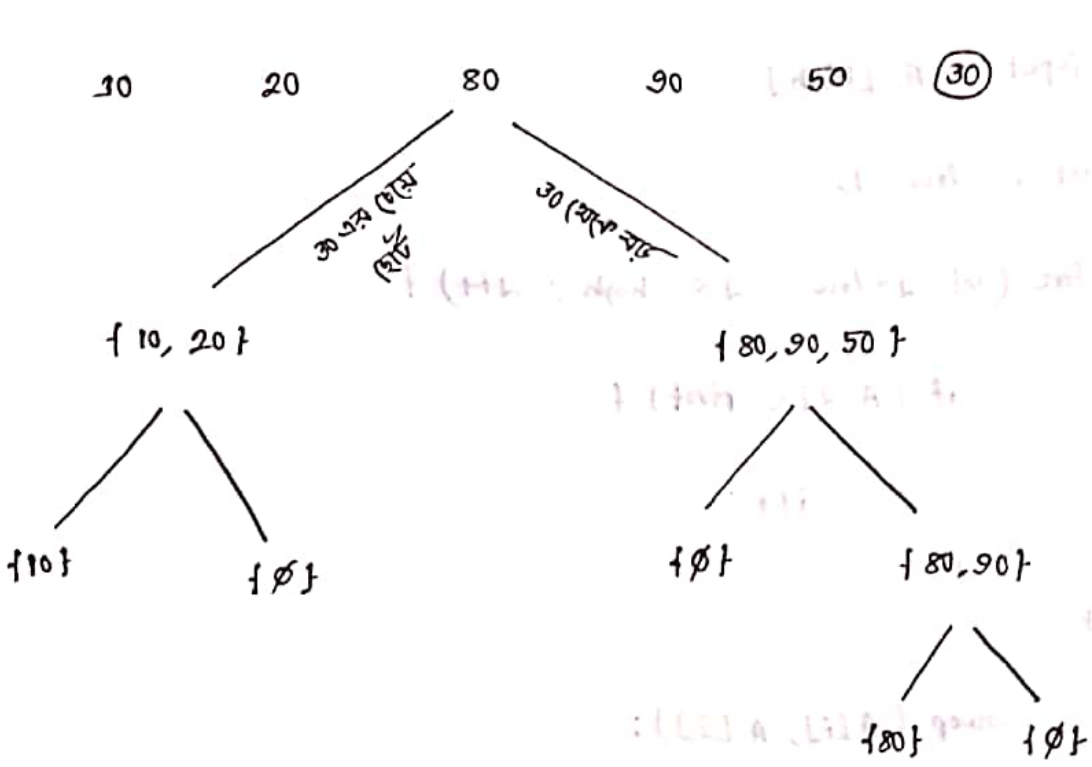


i
10 30 40 100

j
15 20 35 38 42 105 200

Complexity : $n\log_2 n$

Quick sort



pivot
↓
$x_1, x_2 <$ (>=)

1) last
2) first
3) rand
4) medium

```
void qs (A[], int low, int high) {
    if (low > high)  return;
    if (low < high) {
        int pi = PARTITION (A, low, high);   // low=0, high=5
        qs (A, low, pi-1);
        qs (A, pi+1, high);
        qs (A, pi+1, high);
    }
}
```

```
int PARTITION ( int A [], int low, int high) {

    input = A [high] ;

    int i = low - 1;

    for (int j = low ; j <= high ; j++) {

        if (A[j] < pivot) {

            i++;

        }

    }

    swap (A[i], A [j]);

    return (i+1);

}
```

Complexity:

$n \log n$

20-02-20

Radix Sort :

348, 143, 361, 423, 538, 128, 321, 543, 366

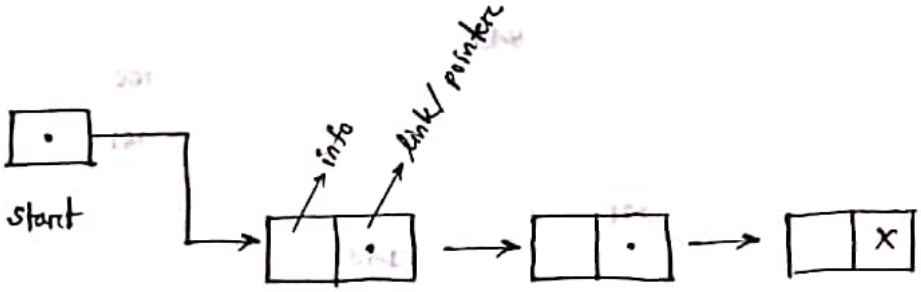
	0	1	2	3	4	5	6	7	8	9
348 143 361 423 538 128 321 543 366		361 321	1	143 423 543			366		348 538 128	
361 321 143 423 543 366 348 538 128			321 423 128	538	143 543 348		361 366			
321 423 128 538 143 543 348 361 366		128 143		321 348 361 366	423	538 543				

128
143
321
348
361
366
423
538
543

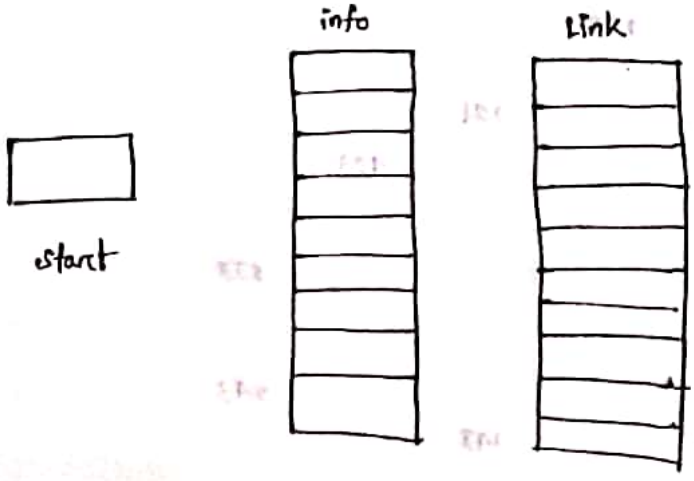
Complexity

$O(n^2)$

Linked List : (Dynamic)



Representation of linked list in Memory



Perfect Square :

Step 1 : 8

1 _____ 8

```
{
  if (x * x = key)        // x = (hi+lo)/2
      found, return true ;
  else if (x*x > key)
      hi = x-1 ;
  else
      lo = x+1 ;
}
return false ;
```

Step 2:

```
p ← input [MAX RANGE]
for (int i=1; i<=p; i++) {
    bool res = binSer (i);
    if (res)
        cout << i << endl;
}
```

Assignment :

- Lower Bound

theoretically / graphically

(ii) Quick sort / Merge sort implement

Traversing a Linked List :

1. Set Ptr := start
2. Repeat step 3 and 4 while Ptr ≠ Null
3. Apply process to Info [Ptr]
4. Set Ptr := Link [Ptr]
5. Exit

searching in a Linked List :

* Unsorted
* Sorted

must question আসবে

Linked List থেকে Minimum Value খুঁজে বের করা

1. Set Ptr := start
2. Set min = ∞
3. Repeat step 4 to 6 while Ptr ≠ Null
4. If (min > Info [Ptr])
5. Set min = Info [Ptr]
6. Set Ptr = Link [Ptr]
7. Exit

min = ∞/a [0]
↓
Info [start]

if (a[i] < min)
min = a[i]

if (info [Ptr] < min)
min := info [Ptr]

```
if [ Info [Ptr] == key ]
    printf ("found");
else
    Ptr := [Link [Ptr]
```

[Linked List use করে যদি key খুঁজে বের করতে হবে]

8 12 16 22 50 100 [14 search করতে হবে]

⇒ Linked List এ Binary Search use করা যায় না। Cause আমরা কোনো mid বের করতে পারবো না।

Effective way → যে number search করতে হবে Linked List এ তার equal পর্যন্ত search করবো। কারন List sorted আছে। 14 (number) এর থেকে বড় হয়ে গেলে Not Found।

```
if [key == info [Ptr]
    Found, Exit
else if [key < info [Ptr]
    Not Found, Exit
else
    Ptr := Link [Ptr]
```

Memory Allocation, Garbage Collection

List (Info, Link, start, Avail)

Overflow, Underflow:

Avail ⇝ value null → overflow

start ⇝ value null → underflow

25-02-20

Inserting into a Linked List :

Beginning

1. If Avail = Null then write overflow and Exit
2. Set New : = Avail and Avail = Link [Avail]
3. Set Info [New] : = Item
4. Set Link [New] : = Start
5. Set Start : = New
6. Exit

Start

5

Avail

7

0	100	8
1		4
2		11
3	50	10
4		9
5	A0	0
6		X
7		1
8	60	3
9		2
10	120	X
11		6

Inserting after a given node

1. overflow ??
2. set New := Avail and Avail := Link [Avail]
3. set Info [New] := Item
4. if LOC = Null then :

   set Link [New] := start

   set start = New

   Else :

   set Link [New] := Link [LOC]

   and Link [LOC] := New

5. Exit

Quiz # 1 ka Syllabus

Q. Pattern

① Algorithm + Modification

② Implementation

③ Conceptual Question

Deleting a node from a Linked List

1. if LOCP = Null then

   set Start := Link [Start]

   Else

   set Link [LOCP] := Link [LOC]

2. set Link [Avail] := LOC

   LOC := Avail

3. Exit

Start

5

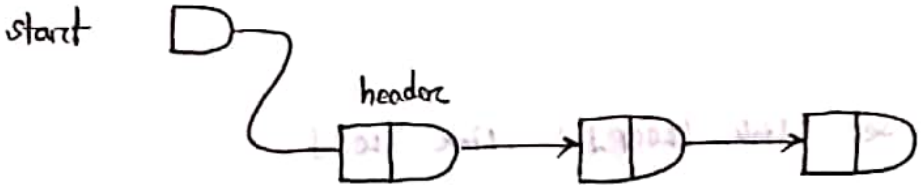
Avail

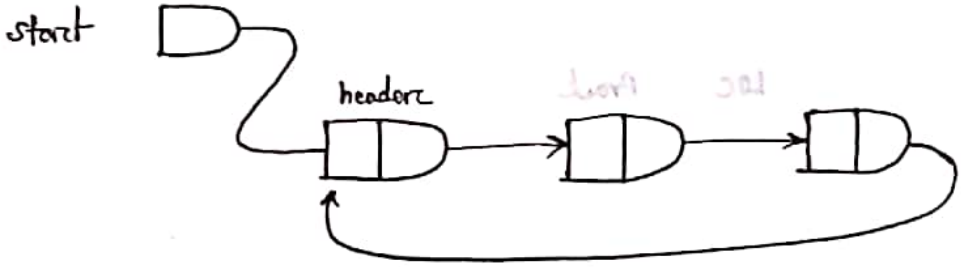
6

0	260	3
1		X
2		8
3	50	7
4		1
5	100	0
6		2
7	90	X
8	.	4

Header and Circular Linked List

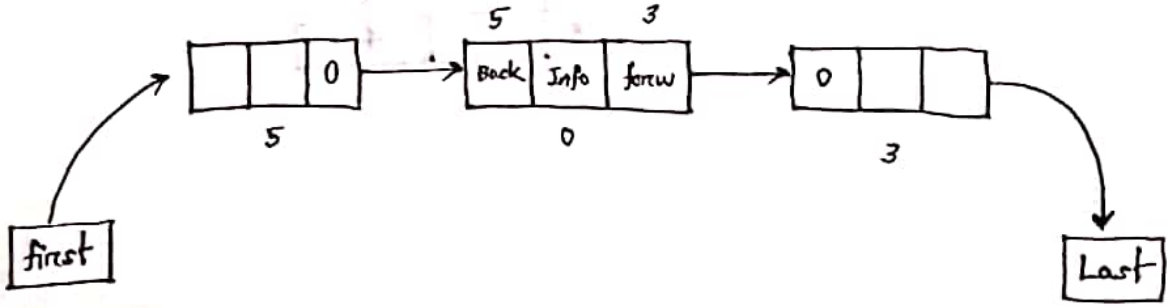
* Grounded
* Circular





Two-way / Doubly Linked List

* Forw
* Info
* Back



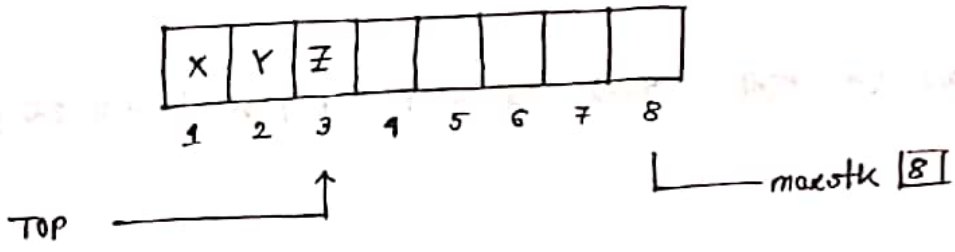
***

must 1টি question আসবে (insertion/ deletion from doubly linked list)

Stack

* TOP
* PUSH
* POP

Array Representation of Stack



Overflow → Top = maxstk & কিছু push করবো

Underflow → Top = NULL & কিছু pop করবো

LIFO

Last in First Out

PUSH

1. If Top = Maxstk then write overflow and Exit
2. set Top : = Top + 1
3. set stack [Top] : = Item
4. Exit

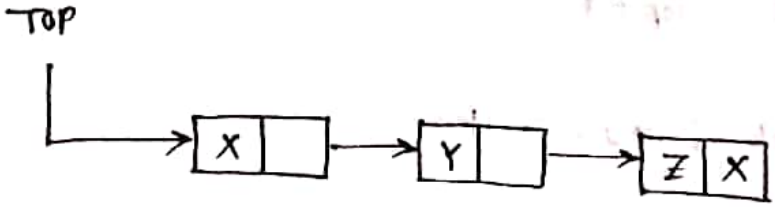
POP

1. If Top = 0 or NULL then write underflow and Exit.
2. Set Item : = stack [Top]
3. set Top : = Top - 1
4. Return

. . . . . .

☐ Avail কে কেনো stack এর মতো implement করা হয় ?

Linked Representation of stack



PUSH

1. overflow ??
2. set New := Avail and Avail := Link [Avail]
3. set Info [New] := Item
4. set Link [New] := TOP
5. set TOP := New
6. Exit

POP

1. underflow ??
2. set Item := Info [TOP]
3. set Temp := TOP

   TOP = Link [TOP]
4. set Link [Temp] := Avail

   Avail := Temp
5. Exit

Arithmatic Expression : Polish Notation

→ prefix

prefix (+AB)

infix (A+B)

postfix (AB+)

→ Reverse Polish Notation

***

Evaluating a postfix Expression :

1. Add a right parenthesis ")" at the end of P
2. Scan P from left to right and repeat step 3 & 4
3. If an operand is found, put it on stack
4. If an operator is found, then :
   - (a) Remove the top two elements of stack, where A is the top
   - (b) Evaluate $B \otimes A$ ← any operator
   - (c) Place the result of (b) back to stack
5. set value equal to the top element of stack
6. Exit.

5, 6, 2, +, *, 12, 4, /, -)

Symbol	stack.
5	5
6	5 6
2	5 6 2
+	5 8
*	40
12	40 12
4	40 12 4
/	40 3
-	37
)	

B+A

6+2 = 8

5*8 = 40

12/4 = 3

40 - 3 = 37

∴ The value of expression = 37

3, 1, +, 2, ↑, 7, 4, -, 2, *, +, 5, -

Symbol	stack
3	3
1	3 1
+	4
2	4 2
↑	16
7	16 7
4	16 7 4

Symbol	stack
-	16 3
2	16 3 2
*	16 6
+	22
5	22 5
-	17
)	

***

Transforming Infix to Postfix Expression :

① push "(" onto stack, and add ")" to the end of Q.

② Scan Q from left to right and repeat steps 3 to 6 for each element of Q until the stack is empty.

③ If an operand is found, add it top

④ If a left parenthesis is found, push it onto stack.

⑤ If an operator ⊗ is found, then :

   ⓐ Repeatedly POP from stack and add to P each operator which has the same or higher precedence than ⊗

   ⓑ Add ⊗ to stack

⑥ If a right parenthesis is found, then :

   ⓐ Repeatedly POP from stack and add to P each operator until a left parenthesis is found.

   ⓑ Remove the left parenthesis.

⑦ Exit.

$A + (B * C - (D / E \uparrow F) * G) * H)$

Symbol Scanned	Stack	Expression P
A	(	A
+	(+	A
(	(+(	
B		AB
*	(+(*	
C		ABC
-	(+(-	ABC*
(	(+(-(	
D		ABC*D
/	(+(-(/	
E		ABC*DE
↑	(+(-(/↑	
F		ABC*DEF
)	(+(-	ABC*DEF↑/
*	(+(-*	~~ABC*DEF↑/~~
G		ABC*DEF↑/G
)	(+	ABC*DEF↑/G*-
*	(+*	
H		ABC*DEF↑/G*-H
)	Empty	ABC*DEF↑/G*-H*+

Precedence :

↑

/ *

+ −

$(A-B) / ((D+E) * F))$

Symbol Scanned	Stack	Expression P
	(	
(	((	
A		A
−	((−	
B		AB
)	(	AB −
/	(/	
(	(/(	
(	(/((	
D		AB − D
+	(/((+	
E		AB − DE
)	(/(	AB − DE +
*	(/(*	
F		AB − DE + F
)	(/	AB − DE + F *
)	Empty	AB − DE + F * /

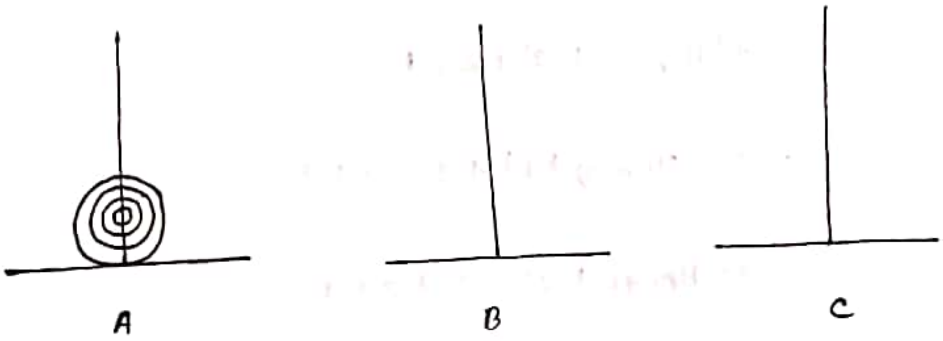
10-03-20

Towers of Hanoi :

① Move the top n-1 disks from pos A to B

② Move the top disk from A to C : A → C

③ Move the top n-1 disks from B to C



Towers (N, Beg, Aux, End)

① A → C (H_1)

② A → B, A → C, B → C (H_2)

③ 3 + 1 + 3 (H_3)

$H_3 + 1 + H_3$ (H_4) $= 2H_3 + 1$

$H_n = 2H_{(n-1)} + 1$

$H_1 = 1$ (Base Condition)

Non Recursive solution :

$$H_n = 2H_{(n-1)} + 1$$

$$= 2(2H_{(n-2)} + 1) + 1$$

$$= 2^2 H_{(n-2)} + 2 + 1$$

$$= 2^2 (2H_{(n-3)} + 1) + 2 + 1$$

$$= 2^3 H_{(n-3)} + 2^2 + 2 + 1$$

$$= 2^3 (2H_{(n-4)} + 1) + 2^2 + 2 + 1$$

$$= 2^4 H_{(n-4)} + 2^3 + 2^2 + 2 + 1$$

$$= 2^{n-1} H_{n-(n-1)} + 2^{n-2} + 2^{n-3} + \_\_\_ + 2^2 + 2 + 1$$

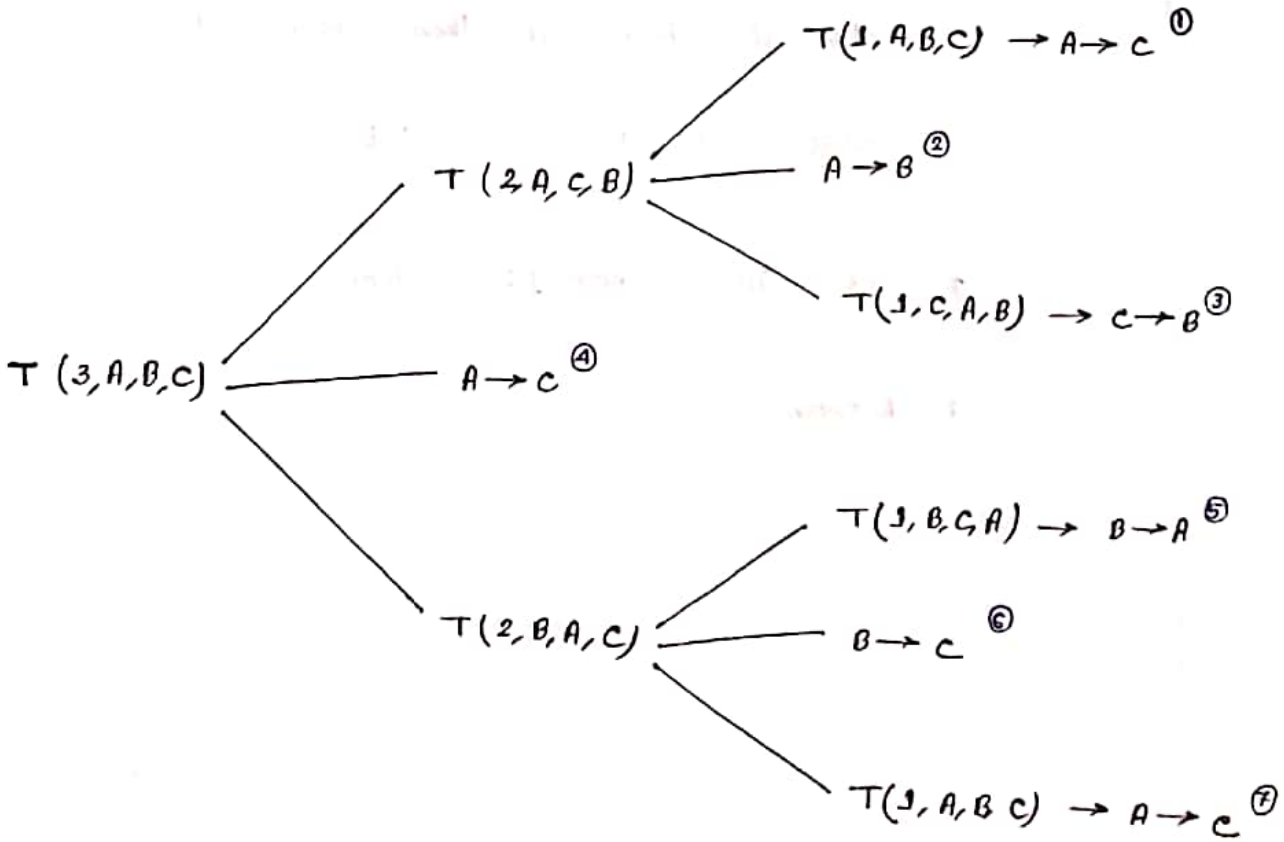
$$= 2^{n-1} + 2^{n-2} + \_\_\_ + 2^2 + 2 + 1$$

$$= 2^{n-1+1} - 1$$

$$= 2^n - 1$$

Recursive solution :

```
tower (N, A, B, C)
{
    if (n == 1)  A → C
    else
        tower (N-1, A, C, B)   // A থেকে B তে যাবে C-এর help নিয়ে
        tower (1, A, B, C)     // A → C
        tower (N-1, B, A, C)   // B → C
}
```



Quiz # 02

12-03-20

Queues :

* Front
* Rear

Qinsert :

1. If Front = 1 and Rear = N or
   if Front = Rear + 1 then write : overflow
2. If front : = Null then :
   set Front : = 1 and Rear : = 1
   else if Rear = N then : Rear : = 1
   else set Rear = Rear + 1
3. set Queue [Rear] : = Item
4. Return

Qdelete :

1. If Front = Null then write : underflow
2. set Item = Queue [Front]
3. If Front = Rear then :

   set Front = Null and Rear = Null,

   else if Front = N then

   set Front : = 1

   else set Front : = Front + 1
4. Return

Deques / Dequeues

* input - restricted
* output - restricted

Priority queues

① An element of higher priority is processed before any element of lower priority.

② Two elements with the same priority are processed according to the order in which they were added to the queue.