Outline # 3

Algorithm

Algorithm is the step by step procedure to solve a problem

Pseudocode

Pseudocode is the simple English like discription of the steps of a procedure and a specification of this procedure using an actual programming language.

Example of Algorithm & Pseudocode

Q: Describe an algorithim for finding maximum value in a finite sequence of integer

Sol:

Algorithim

(i) Temporarily set the max equal to first integer of sequence
(ii) Compare the max with the next value of sequence and if the next integer is greater then max value, then set max equal to 2nd integer of sequence.
(iii) Repeat the (2nd step) if there are more integers in the sequence.
(iv) Stop when there is no integer left in the sequence. The temp max at this point is the largest integer in the sequence.

Pseudocode

$(a_1, a_2, a_3, \ldots, a_n)$ integers sequence
max = a_1
for i = 2 to n
    if max < a_i then
        max = a_i

return max (max is the largest value)

Searching Algorithms

The procedure of locating a specific value from a list of values is called searching algorithm.

(i) Linear Search

Linear Search is also called sequential search.

Linear searching is done by compairing the value, which we want to search, with all values in a sequence or list. The search begins with compairing the value with the first integer. If a_1 (first integer) = x (searched value) the solution is the location of a_1, namely 1. If $a_1 \neq x$, then a_2 is compared with x. If $a_2 = x$ then solution is a_2's location. If $a_2 \neq x$, continue the process by compairing x with each term in list. If the entire list has been searched without locating x, the solution is 0.

Algo

Procedure : Linear Search)
x from (a1, a2, a3, ... an)

```
for (i = 1, i<=n, i++)
{
  if (a[i] == x)
  {
    loc = i
    break;
  }
  else {loc = 0}
}
return loc;
```

```
loc=-1
for(i=1; i<=n; i++)
{
  if (x == a[i])
  {loc = i
   break }
}
if (loc==-1)
{ S.O.P ("x not found");
else if (loc != 1)
{S.O.P("x found at i index")}
```

Binary Search

Binary Search algo can be used when the list has values occuring in the order of increasing size (ascending order)

It procedes by compairing the element to be located in the middle term of list.
First the list is split into 2 parts.
by compairing the value with the middle value of list.
If value ≠ middle value then we check if value is greater then middle value or if value is smaller then middle value.

eg

if value is greater then middle value, then we ignore the value on left of middle value of list and compare the value with the middle value of remaining list.

e.g (1 2 3 4 5 | 6 7 (8) 9 10) value = 9
ignore | middle value
compare 9 with middle value.

This process continues until we found the value at the middle value.

Algo

```
int l=left=0 , r=high = size-1
mid = (l+r)/2;

while ( l<=r) {
  if (n==a[mid])
  { S.O.P ("value found");
    break;
  }
else if( n>a[mid])
  {l= mid+1 ; }

else if ( n<a[mid])
  {r= mid-1 ;
  }
  if (l>r)
    { S.O.P ( value not found);
}  }
```

Sorting

Sorting is the process of putting the element into a list in increasing order

eg

unsorted array : 7, 2, 5, 3, 1

Sorted array = 1, 2, 3, 5, 7

Bubble Sort

Bubble sort is the simplest sorting algorithm.

It puts a list into increasing order by succesively comparing adjacent element, interchanging them if they are in wron order.

Algo:

```
for (i=5; j>0; j--)
{ for (i=0; i<5; i++)
    if (a[i] < a[i+1])
    { temp = a[i];
      a[i] = a[i+1]
      a[i+1] = temp
    }
}
```

eg { 5, 2, 3, 4, 5

1, 2, 3, 4, 5 ①

2, 1, 3, 4, 5 ②

2, 3, 1, 4, 5 ③

2, 3, 4, 1, 5 ④

2, 5, 4, 5, 1

2, 3, 4, 5, 1 ①

3, 2, 4, 5, 1 ②

3, 4, 2, 5, 1 ③

3, 4, 5, 2, 1

Insertion Sort

To sort the list with n element, the insertion sort begins with the 2nd element. The insertion sort compares this second element with the first element and inserts it before the first element, if the 2nd element is smaller then first element. And it inserts after the ~~first 2nd~~ element if 2nd is greater then 1st element.

Example { 3, 2, 4, 1, 5 }

```
int i, key, j;
for(j=1 ; j<5 ; j++)
{
    key = arr[j]
    i = j-1;

while( i>=0 && arr[i] > key )
    {
    arr[i+1] = arr[i]
    i = i-1;
    }
    arr[i+1] = key ;
}
```

Quick Sort

Rules

=> First you have to chose a pivit element out of all an array.

pivot element can be any element from array.

e.g (35) 50 15 25 80 20 90 45

pivot element

=> Next you have to chose Two pointers as pointers p & q respectively

(35) 50 15 25 80 20 90 45

p q

=> 'p' element will move to its right side and stops when it will find an element greater than "pivot"

=> 'q' element will move to its L.H.S and stops when it will find the lesser element than pivot value.

Why we do it?

Just to bring all the greater elements to the right of pivot value and lesser value to its left side

through this we can sort an array through quick sort method.

35 5 4 3 2 1 +∞ → will stop there if there is no small elem

(35) 50 15 25 80 20 90 45 +∞

p ↓ q ↓

=> This is to stop the p traversing in case of it doesn't find greater element

=> p will check >=

=> q will check <=

1st pass

(35) (50) 15 25 80 20 90 45

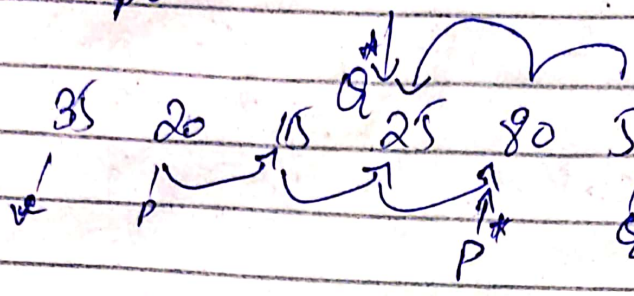
v ↓ p ↑↓ q ↑ ↑ ↑ ↓

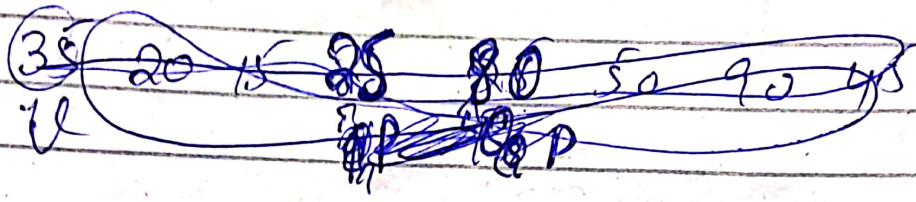
=> Now p is at 50 and q is at 20, Check that they both crossed each other or not, if not then swap both the value

35 20 15 25 80 50 90 45

v ↑ p ↓ q ↓

Repeat





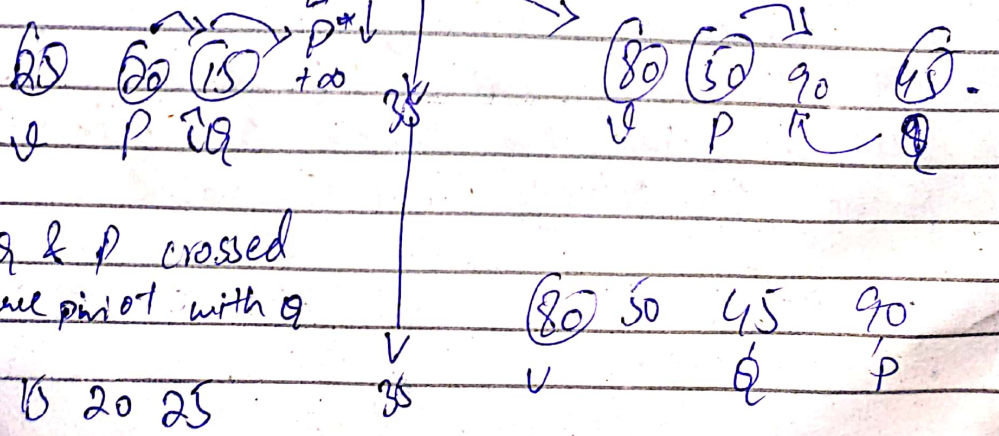
P & Q crossed each other

⇒ Now check whether P & Q have crossed each other. or not, if crossed then replace pivot value with Q.

⇒ This condition also holds when both P and Q will be at same position

25 20 15 35 80 50 90 45

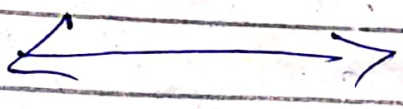
Now 35 is at its rig8 position.



Now both P & Q crossed

so replace P & Q

10 20 25 35 45 50 80 90



Integers:

Integers are the set of numbers from 0 to $+\infty$, or 1 to ∞

Division

When one integer is divideed by a second non zero integer, the quotient may or may not be an integer.

For Example:

$12/3 = 4$ is an integer

$11/4 = 2.75$ is not an integer.

Division Algorithm

Let 'a' be an integer and d a positive integer. Then there are unique integers q and r such that

$$a = dq + r$$

∵ a = dividend
d = divider
q = quotient
r = remainder

$\Rightarrow$ 'q' = 'a' div 'd' $\Rightarrow$ 'r' = 'a' mod 'd'

Example

Q: What are the quotient & remainder when 101 is divided by 11?

q = a div d
= 101 / 11
q = 9

r = a mod d
= 101 % 11
= 2

Matrices:

$$A = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 5 & 0 \end{bmatrix}$$

Order of matrix
= m x n
A = 2 x 3

=> Scalar matrices = which have same number of rows & columns in both

=> Rectangular matrices = which have not same number of rows & columns.

$$A+B = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

$$= \begin{bmatrix} 7 & 9 \\ 11 & 13 \end{bmatrix}$$

$$A.B = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ 2 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1+8+6 & 2+6+0 \\ 2+4+8 & 4+3+0 \end{bmatrix}$$

$$= \begin{bmatrix} 15 & 8 \\ 14 & 7 \end{bmatrix}$$

⇒ Transponse of a matrics

$$A = \begin{bmatrix} 1 & 2 & 3 \\ a & b & c \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & a \\ 2 & b \\ 3 & c \end{bmatrix}$$

∴ convert rows into column and column into rows It is transponse.

⇒ Zero. one matrix

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \quad , B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

=> Join of zero-one matrices

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$A \vee B$ (OR)

$$= \begin{bmatrix} 1 \vee 1 & 0 \vee 1 & 0 \vee 1 \\ 0 \vee 0 & 1 \vee 1 & 1 \vee 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

=> Meet of zero-one matrices

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$A \wedge B = \begin{bmatrix} 1 \wedge 1 & 0 \wedge 1 & 1 \wedge 0 \\ 0 \wedge 1 & 0 \wedge 1 & 1 \wedge 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

=> Boolean product zero-one matrices:

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$A \odot B = \begin{bmatrix} (0 \wedge 1) \vee (1 \wedge 1) & (0 \wedge 1) \vee (1 \wedge 1) \\ (1 \wedge 1) \vee (0 \wedge 1) & (1 \wedge 1) \vee (0 \wedge 1) \end{bmatrix}$$

$$= \begin{bmatrix} (0 \vee 1) & (0 \vee 1) \\ (1 \vee 0) & (1 \vee 0) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

=> Boolean power of a square zero-one matrice

Find B^2 $\quad \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

$$B \odot B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} (1 \wedge 1) \vee (1 \wedge 1) & (1 \wedge 1) \vee (1 \wedge 1) \\ (1 \wedge 1) \vee (1 \wedge 1) & (1 \wedge 1) \vee (1 \wedge 1) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$