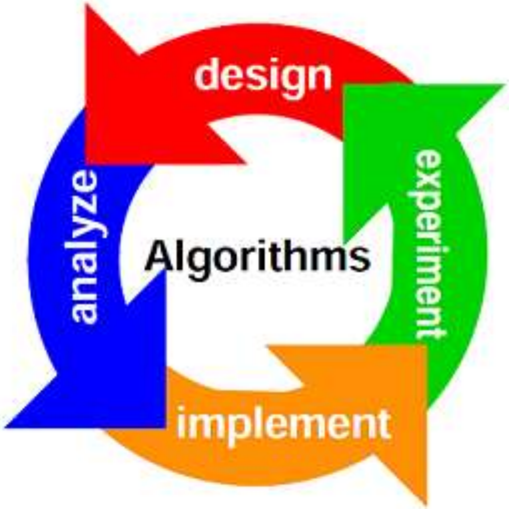
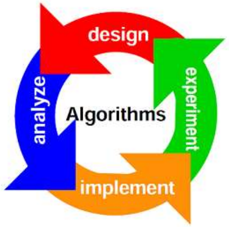


DESIGN AND ANALYSIS OF ALGORITHMS

Introductory Session

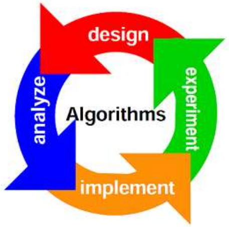
4/9/2024

Module I



- **Introduction:** Algorithm, Concepts in performance analysis – space complexity and time complexity, Asymptotic Notations
- **Sorting:** Analysis of - Bubble sort, Selection sort and Insertion sort
- **Searching:** Analysis of - Linear Search, Binary Search and Interpolation Search.
- **Hashing Techniques**: Different hashing functions, methods for collision handling.

Algorithm

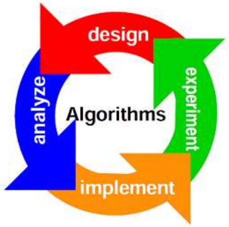


A step by step procedure for solving Computational Problems.

Example

- Sorting Problem
  - Input:A sequence of n numbers(a1,a2,...an)
  - Output:A permutation (reordering)(a1,a2,...an) of the input sequence such that a1<=a2<=...an

Algorithm Vs Program



Algorithm	Program
Design	Implementation
Requires Domain Knowledge	Programmer
Any Language	Specific programming language
Hardware and Operating System Independent	Hardware and Operating System Dependent
Analysis	Testing

Priori Analysis

- Algorithm
- Language Independent
- Hardware Independent
- Time and Space function

Posteriori Testing

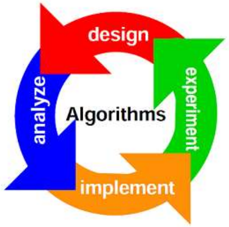
- Program
- Language Dependent
- Hardware and OS dependent
- Watch Time and Bytes

Characteristics Of Algorithm

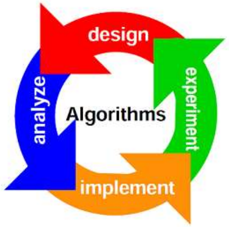
Algorithm

- **Input (Range)**
- **Output**
- **Definiteness**
- **Finiteness**
- **Effectiveness**
- **Efficiency(Speed)**

Facts…



- An algorithm is said to be correct if for every input instance, it halts with correct output
- Algorithms should be designed such that it is efficient in terms of time and space
- Computing time and space is a bounded resource



Issues In Algorithms

- ✓ How to devise algorithms
- ✓ How to validate algorithms
- ✓ How to analyze algorithms

Analysis of Algorithm- Factors

- Time Efficiency/Time Complexity
- Space Efficiency/Space Complexity
- Simplicity
- Generality
- Range of input(Measuring Input size)
- Computing best case, worst case and average case efficiencies
- Computing Order of growth

Performance Analysis of an Algorithm

Performance Analysis Measures

Space/Time Complexity

- The space complexity of an algorithm is the amount of memory it needs to run to completion.
- The time complexity of an algorithm is amount of computer time it needs to run to completion

Space Analysis Of An Algorithm

- Space required to store the data values
- The space needed is the sum of the component factors:
  - Fixed part : Space of inputs & outputs, Instruction
  - Variable part: Space dependent upon instance characteristics
- The space requirement S(P) of any algorithm P can be written as S(P)=c + Sp where c is a constant

Time Analysis Of An Algorithm

- The time complexity of an algorithm is amount of computer time it needs to run to **completion -running time** + compile time
- Difficult to compute the running time complexity in terms of physically clocked time, why?
- Time complexity is given in terms of frequency count of the basic operation
- **Frequency count** denotes the no. of times of execution of statement(basic operation).

Time Analysis Of An Algorithm

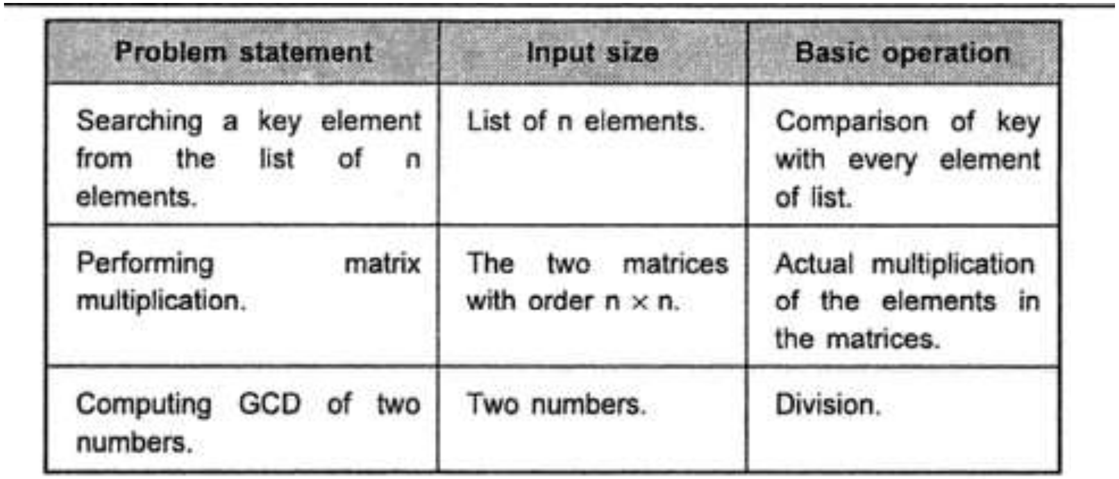
- The estimate of the time is calculated by isolating a particular operation called as a basic operation
- Example:

```
sum=0
Repeat for I=1,2 ….n
sum=sum+V[I]
Exit
```

- In the above example the operation to be isolated is addition , the other operation is assignment
- The rest of the operations are called book keeping operations and generally not counted

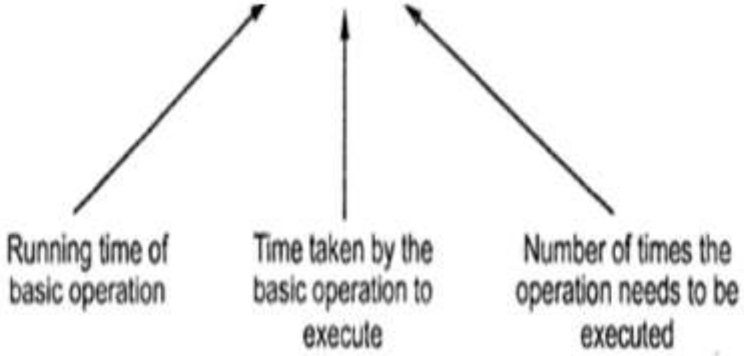
Basic Operation

- Time consuming operation of an algorithm
- Normally, located in *inner loop*



Problem statement	Input size	Basic operation
Searching a key element from the list of n elements.	List of n elements.	Comparison of key with every element of list.
Performing matrix multiplication.	The two matrices with order n × n.	Actual multiplication of the elements in the matrices.
Computing GCD of two numbers.	Two numbers.	Division.

$T(n) = t(B_{op})$*No. of Basic Operations



Running time of basic operation

Time taken by the basic operation to execute

Number of times the operation needs to be executed

Other Factors

- Network Consumption
- Power Consumption

Analyzing Algorithms

Analyzing the Algorithms-I

- Frequency Count Method
  - Swapping two Elements

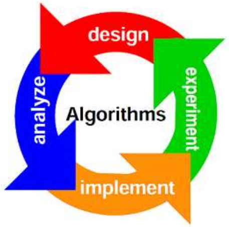
```
Algorithm Swap(a,b)
{
        temp=a;
        a=b;
        b=temp;
}
```

Analyzing the Algorithms-II

- Sum of elements of an array

```
Algorithm Sum(A,n)
{
      s=0;
      i=0;
      While(i<n)
      {
            s=s+A[i];
            i++;
      }
      return s;
}
```

Analyzing the Algorithms-II



- Sum of elements of an array

```
Algorithm Sum(A,n)
{
     s=0;
     for(i=0;i<n;i++)
     {
          s=s+A[i];
     }
     return s;
}
```

Analyzing the Algorithms-III

- Matrix Addition

```
Algorithm Add(A,B,n)
{
        for(i=0;i<n;i++)
        {
        for(j=0;j<n;j++)
                {
                        C[i,j]=A[i,j]+b[i,j];
                }
        }
}
```

Analyzing the Algorithms-IV

- Matrix Multiplication

```
Algorithm Multiply(A,B,n)
{
        for(i=0;i<n;i++)
        {
        for(j=0;j<n;j++)
                {
                        C[i,j]=0;
                        for(j=0;j<n;j++)
                        {
                                C[i,j]=C[i,j]+A[i,k]*B[k,j];
                        }

                }
        }
}
```

Problems: Find the Time Complexity

- Algorithm for finding the factorial

```
function(int n)
{
    if (n==1)
       return;
    for (int i=1; i<=n; i++)
    {
        for (int j=1; j<=n; j++)
        {
            printf("*");
            break;
        }
    }
}
```

Thank you!