- **Algorithm:** It is a combination of sequence of finite steps to solve a particular problem.

Properties of Algorithm: • output should be generated after finite time.

- There should be at least one input.
- It's independent from programming language.

Difference between Algorithm and Program:

Algorithm	Program
• Written at Design stage	• Written at implementation stage
• Need domain expert	• Need programer
• Written in any language	• Written in programming language
• H/W or OS independent	• H/W or OS dependent
• Can be Analyze	• Can be tested

Pseduo Code: • It's a description of an algorithm that is more structured than usual prose but less formal than a Programming language.

- Pseduo Code is our preferred notation for describing algorithms.

Measuring the running time of an algorithm:

App1: Experimental study: Write a program that implements the algorithm.

App2: Frequency Count Method:

Prob 1:

```
Main()
{
   x=y+z;  → 1
}
                 O(1)
```

Prob 2:

```
main()
{
 x=y+z; → 1
 for(i=1; i<=n; i++)
   { x=y+z; → n
   }
}           n+1
            ----
            O(n)
```

Prob 3:

```
main()
{ x= y+z; → 1
 for(i=1; i<=n; i++)
 {  x=y+z; → n
 }
 for(i=1; i<=n; i++)
  {
   for(j=1; j<=n; j++)
    {  x= y+z; → $n^2$
    }
  }                $n^2+n+1$
}                  ---------
                   $O(n^2)$
```

Prob 4:

```
main()
{
 while (n>=1)
  {
   n= n-10 → $\frac{n}{10}$
  }
}             ⇒ O(n)
```

Prob 5:

```
main()
{ i=0
  while (i<=n)
   { i= i+5 → $\frac{n}{5}$
   }
}               ⇒ O(n)
```

Prob 6:

```
main()
{
 while (n>=1)
  {  n= $\frac{n}{2}$;
  }
}
```

$$n \;:\; k=\log_2 n$$
$$\frac{n}{2} \;:\; O(\log_2 n)$$
$$\frac{n}{2^2} \;\; \frac{n}{2^k}=1$$

Prob7:

```
main ()
{ i=1
  while (i<=n)
  { i = 2*i;
  }
}
```

$i=1$

$2*1=2$

2^2

$\vdots$

$2^k=n$

$k=\log_2 n$

$O(\log_2 n)$

Prob8:

```
main ( )
{
  while (n>2)
  {
     $n = n^{1/2}$
  }
}
```

n

$n^{1/2}$

$n^{1/2^2}$

$\vdots$

$n^{1/2^k}$

$$n^{\frac{1}{2^k}} = 2$$

$$\frac{1}{2^k}\log_2 n = \log_2 2$$

$$\log_2 n = 2^k$$

$$k = \log_2 \log_2 n$$

$$\boxed{O(\log_2 \log_2 n)}$$

Prob 9:

```
main()
{
while(n>23)
{
   $n = n^{\frac{1}{255}}$
}
}
```

$$n^{\frac{1}{255^k}} = 23$$

n

$n^{\frac{1}{255}}$

$n^{\frac{1}{255^2}}$

$\vdots$

$n^{\frac{1}{255^k}}$

$$\frac{1}{255^k}\log_{23} n = \log_{23} 23$$

$$k = \log_{255} \log_{23} n$$

$$\boxed{O(\log_{255} \log_{23} n)}$$

Prob 10:

```
main()
{
    while(n >= 15)
    {
        n = $n^{1/5}$
    }
}
```

n

$n^{1/5}$

$n^{1/5^2}$

$\vdots$

$n^{\frac{1}{5^k}}$

$$n^{\frac{1}{5^k}} = 15$$

$$\frac{1}{5^k} \cancel{\log} \log_{15} n = \log_{15} 15$$

$$\log_{15} n = 5^k$$

$$k = \log_5 \log_{15} n$$

$$\boxed{O(\log_5 \log_{15} n)}$$

Prob 11:

```
main()
{
    i = 2
    while (i < n)
    {   i = $i^2$
    }
}
```

2

2^2

$(2^2)^2$

$\vdots$

$$(2^2)^k = n$$

$$2^k = \log_2 n$$

$$k = \log_2 \log_2 n$$

$$O(\log_2 \log_2 n)$$

Prob 12:

```
main()
{   i = 3
    while (i < n)
    {   i = $i^2$
    }
}
```

3

3^2

$(3^2)^2$

$\vdots$

$$(3^2)^k = n$$

$$2^k = \log_3 n$$

$$k = \log_2 \log_3 n$$

$$\boxed{O(\log_2 \log_3 n)}$$