# *Design and Analysis of Algorithms*

**Noor-ul-Ain**

National University of Computer and Emerging Sciences,
Islamabad

## Algorithm Characteristics

The necessary features of an algorithm:

- *Definiteness*
  - The steps of an algorithm must be precisely defined.
- *Effectiveness*
  - Individual steps are all do-able.
- *Finiteness*
  - It won’t take forever to produce the desired output for any input in the specified domain.
- *Output*
  - Information or data that goes out.

# Algorithm Characteristics

Other important features of an algorithm:

- *Input*.
    - Information or data that comes in.
- *Correctness.*
    - Outputs correctly relate to inputs.
- *Generality.*
    - Works for many possible inputs.
- *Efficiency.*
    - Takes little time & memory to run.

## *Complexity Analysis*

**Want to achieve platform-independence**

Use an abstract machine that uses *steps* of time and *units* of memory, instead of seconds or bytes

- ✓ each elementary operation takes **1 step**
- ✓ each elementary instance occupies **1 unit of memory**

## *Standard Analysis Techniques*

- Constant time statements
- Analyzing Loops
- Analyzing Nested Loops
- Analyzing Sequence of Statements
- Analyzing Conditional Statements

# *Analysing an Algorithm*

- **Simple statement sequence**

  ```
  s1; s2; …. ; sk
  ```

  - Basic Step = $1$ as long as $k$ is constant

- **Simple loops**

  ```
  for(i=0; i<n; i++) { s; }
  ```

  where `s` is Basic Step = $1$

  - Basic Steps : $n$

- **Nested loops**

  ```
  for(i=0; i<n; i++)
      for(j=0; j<n; j++) { s; }
  ```

  - Basic Steps : $n^2$

## *Analysing an Algorithm*

- **Loop index depends on outer loop index**

```
for(j=0;j<=n;j++)
    for(k=0;k<j;k++){
       s;
       }
```

- **Inner loop executed**
    - **1, 2, 3, …., n times**

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$\therefore$ **Basic Steps :** $n^2$

# *Analysing an Algorithm*

**// Input: int A[N], array of N integers**
**// Output: Sum of all numbers in array A**

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

**How should we analyse this?**

# Analysing an Algorithm

```
// Input: int A[N], array of N integers
// Output: Sum of all numbers in array A

int Sum(int A[], int N){
    int s=0;      1

    for (int i=0; i< N; i++)
    2                  3       4
  5   s = s + A[i];
              6       7
   return s;
}                 8
```

1,2,8: Once

3,4,5,6,7: Once per each iteration of for loop, N iteration

Total: 5N + 3

The *complexity function* of the algorithm is : $f(N) = 5N + 3$

## *Constant time statements*

- **Simplest case: O(1) time statements**

- **Assignment statements of simple data types**
  **int x = y;**

- **Arithmetic operations:**
  **x = 5 * y + 4 - z;**

- **Array referencing:**
  **A[j] = 5;**
- **Most conditional tests:**
  **if (x < 12) ...**

## *Analyzing Loops*

- **Any loop has two parts:**
  - **How many iterations are performed?**
  - **How many steps per iteration?**

```
int sum = 0,j;
for (j=0; j < N; j++)
  sum = sum +j;
```

## *Analyzing Loops*

- **Any loop has two parts:**
  - **How many iterations are performed?**
  - **How many steps per iteration?**

```
int sum = 0,j;
for (j=0; j < N; j++)
  sum = sum +j;
```

  - **Loop executes N times (0..N-1)**
  - **O(1) steps per iteration**
- **Total time is N * O(1) = O(N*1) = O(N)**

## *Class Activity*

```
int sum = 0;
  for (i=1;i<n; i=i+2)
    sum = sum +i;
```

## *Class Activity*

```
int sum = 0,j;
   for (i=1;i<n; i=i+2)
     sum = sum +i;
```

$F(n) = n/2;$

$F(n) = O(n)$

## *Analyzing Loops*

- **What about this for loop?**

```
int sum =0, j;
 for (j=0; j < 100; j++)
    sum = sum +j;
```

- **Loop executes 100 times**
- **O(1) steps per iteration**
- **Total time is 100 * O(1) = O(100 * 1) = O(100) = O(1)**

## *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<n; j++)
   for (k=0; k<n; k++)
      sum += k+j;
```

## *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<n; j++)
   for (k=0; k<n; k++)
      sum += k+j;
```

n+1 times

(n) * n+1 times

n * n times

$F(n) = 2n^2+2n+1$

$F(n) = O(n^2)$

# *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<n; j++)
   for (k=0; k<n; k++)
      sum += k+j;
```

- **Start with outer loop:**
  - **How many iterations? N**
  - **How much time per iteration? Need to evaluate inner loop**

- **Inner loop uses O(N) time**
- **Total time is N * O(N) = O(N*N) = $O(N^2)$**

## *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<N; j++)
   for (k=N; k>0; k--)
      sum += k+j;
```

?

## *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<N; j++)
   for (k=N; k>0; k--)
      sum += k+j;
```

- **Start with outer loop:**
  - **How many iterations?  N**
  - **How much time per iteration? Need to evaluate inner loop**

- **Inner loop uses O(N) time**
- **Total time is N * O(N) = O(N*N) = $O(N^2)$**

## *Analyzing Nested Loops*

- **Treat just like a single loop and evaluate each level of nesting as needed:**

```
int j,k;
for (j=0; j<N; j++)
   for (k=N; k>0; k--)
      sum += k+j;
```

- **Start with outer loop:**
  - **How many iterations?  N**
  - **How much time per iteration? Need to evaluate inner loop**

- **Inner loop uses O(N) time**
- **Total time is N * O(N) = O(N*N) = O($N^2$)**

## *Analyzing Nested Loops*

- **What if the number of iterations of one loop depends on the counter of the other?**

```
int j,k;
for (j=0; j < N; j++)
   for (k=0; k < j; k++)
      sum += k+j;
```

- **Analyze inner and outer loop together:**
- **Number of iterations of the inner loop is:**
- **$0 + 1 + 2 + ... + (N-1) = O(N^2)$**

## *Class Activity*

```
void add( int A[ ], int B[ ], int n)
{
    for (i=0; i<n; i++)
     {
        for (j=0; j< n; j++)
          {
                c[i,j]  =A[i][k]  + B[k][j];


           }
        }
}
```

# *Class Activity*

```
void multiply( int A[ ], int B[ ], int n)
{
   for (i=0; i<n; i++)
     {
       for (j=0; j< n; j++)
          {
            c[i,j]= 0;
                for (k=0; k< n; ++)
                   {
                    c[i,j]+=A[i][k]*b[k][j];
                    }
            }
       }
}
```