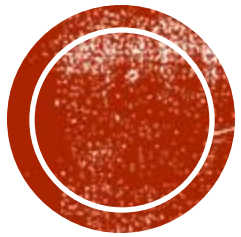

COMPLEXITY ANALYSIS

WORST, BEST AND AVERAGE CASE ANALYSIS

Worst Case Analysis:

Maximum running time an algorithm will take for input size n.

Consider an input for which the algorithm takes the longest time

Informative and can be usually done.

Best Case Analysis:

Minimum running time that an algorithm takes for an input size n

Consider an input for which the algorithm takes the shortest time

This is not very informative

Average Case Analysis:

Average running time that an algorithm will take for input size n

Consider all possible inputs and compute average.

Generally very difficult. Because all combinations of the input are equally likely.

COUNTING SIMPLE INSTRUCTIONS

- Count the simple instructions
  - Assignments have cost of 1
  - Comparisons have a cost of 1
  - Let's count all parts of the loop

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

    - j=0 has a cost of 1, j<n executes n+1 times, and j++ executes n times for a total cost of 2n+2
  - Each statement in the repeated part of a loop have a cost equal to number of iterations

EXAMPLES

```
sum = 0;                                  -> 1
sum = sum + next;                         -> 1
```

Cost

Total Cost: 2

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

Cost

Total Cost: 3n + 2

```
k = 0                                     -> 1
for (int i = 0; i < n; i++)               -> 2n+2
  for (int j = 0; j < n; j++)             -> n(2n+2) = 2n² +2n
    k++;                                  -> n²
```

Cost

Total Cost: $3n^2 + 4n + 3$`

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
Assignment
for(i=n-2;i>=1;i--)
      Arithmetic Operation
Output
for(i=0;i<n;i++)
      Comparison
for(i=0;i<6;i++)
      Assignment
}
```

Cost

1

1 + (n-1) + (n-2) = **2n-2**

n-2

1

1+(n+1)+n = **2n+2**

n

1+7+6 = **14**

6

6n+20

O(n)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm( int n )
{
        Input
        for(i=n;i>=1;i=i-4)
                Arithmetic Operation
        Output
}
```

1

1 + (n/4+1) + n/4 = 2+n/2

n/4

1

(3n+16)/4

O(n)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Input
        for(i=1;i<=n;i=i+5)
                Comparison
        Output
}
```

1

Loop runs Approximately n/5 times

1

O(n)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Assignment
        for(i=n-1;i>=1;i/=2)
                Arithmetic Operation
        Output
}
```

When n=51

Iterations of I every step becomes:

50, 25, 12, 6, 3, 1, 0

Reduces by half every time

=> Can be represented as a logarithmic function of base 2.

T(n) is approximately $[\log_2 n]$

T(n) is O(logn)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Assignment
        for(i=1;i>=n-1;i*=2)
                Arithmetic Operation
        Output
}
```

When n=51

Iterations of I every step becomes:

1,2,4,8,16,32,64

Doubles every time

=> Can be represented as a logarithmic function of base 2.

T(n) is approximately $[\log_2 n]$

T(n) is O(logn)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Input
        for(i=0;i<n;i++)
        {
                Assignment
                for(j=0;j<n;j++)
                        Comparison
        }
        Arithmetic Operation
        Output
}
```

1

1+(n+1)+n = **2n+2**

n

n(2n+2)= **$2n^2$+2n**

n^2

1

1

T(n) = $3n^2$+5n+5

T(n) is O(n^2)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Input
        for(i=0;i<n;i++)
        {
                for (i=n-1;i>=1;i/=2)
                        Arithmetic Operation
                for(j=0;j<n;j++)
                        Comparison
        }
        Arithmetic Operation
        Output
}
```

Loop 1 : n

Loop 2 : n * logn highest term of n in T(n)

Loop 3: n * n

n^2 is the highest term of n in T(n)

T(n) is O(n^2)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Input
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        for(k=0;k<n;k++)
                                Comparison
        Arithmetic Operation
        for(i=0;i<n;i++)
                for(j=10;j>=1;j--)
                        Assignment

}
```

Loop 1: n

Loop 2: n*n

Loop 3: n*n*n

Loop 4: n

Loop 5: 10n

The highest term of n in T(n) is n^3

T(n) is O(n^3)

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Assignment
        for(i=0;i<n;i++)
                for(j=0;j<=i;j++)
                        Operation
        Output
}
```

Outer loop	Value of i	Inner loop Iterations
1	0	1
2	1	2
3	2	3
n-1	n-2	n-1
n	n-1	n

Total Iterations = 1+2+3+…n

$\Rightarrow n(n+1)/2$

T(n) is approximately $n^2/2 + n/2$

=> T(n) is $O(n^2)$

BIG O ANALYSIS OF ALGORITHMS

```
someAlgorithm(int n)
{
        Assignment
        for(i=0;i<n;i++)
                for(j=i+1;j<=n;j++)
                        Operation
        Output
}
```

Outer loop	Value of i	Inner loop Iterations
1	0	n-1
2	1	n-2
3	2	n-3
n-2	n-3	2
n-1	n-2	1

Total Iterations = (n-1)+(n-2)+(n-3)+….+2+1

$\Rightarrow n(n-1)/2$

T(n) is approximately $n^2/2 - n/2$

=> T(n) is $O(n^2)$