Having a general idea about the running time of an algorithm is very important for both programmers and the users. Big-O notation is designed to capture the worst-case running time of an algorithm as a function of the size of the input.

**Definition: Big-Oh Notation**

Let $f, g : \mathbb{N} \to \mathbb{R}^+$. We say that $f$ is “big-oh” of $g$, written $f = O(g)$, or $f \in O(g)$, if ...

**Remark 1:** A useful way of determining big-O of a function:

**Remark 2:** The big-O notation is not sensitive to multiplicative constants, lower order terms, or the basis of a logarithm.

**Example:**

a) $f(n) = 2n^3 + 3n^2 + 100$

b) $f(n) = n + 10\sqrt{n} + \log(n)$

c) $f(n) = 2^n + n^7 + 10^3$

**Question:** Suppose $f(n)$ is $O(g(n))$ and $g(n)$ is $O(h(n))$. Is it true that $f(n)$ is $O(h(n))$?

**Question:** What is $O(1)$? What is $O(n)$?

**Example 1:** What is the best-case, worst-case and average case running time of the sequential search algorithm?

**Example 2:** What is the best-case, worst-case and average case running time of the binary search algorithm?

**Example 3:** What is the number of steps to solve the towers of Hanoi puzzle?
**Example 4:** What is the running time of the bubble sort algorithm? Is there any difference between the best-case and worst case?

```c
for i ∈ {1, 2, 3, ..., n - 1} do
    for j ∈ {1, ..., n - i} do
        if (x_j > x_{j+1}) then swap(x_j, x_{j+1})
```

**Example 5:** Matrix multiplication. The following code multiplies two $n \times n$ matrices $A$ and $B$, and stores the result in another matrix $C$. Determine its running time in Big-Oh notation.

```c
void matrixmult(int n, const int A[][n], const int B[][n], int C[][n])
{
    int i, j, k;
    for (i=1; i<=n; i++) {
        for (j=1; j<=n; j++) {
            C[i][j] = 0;
            for (k=1; k<=n; k++)
                C[i][j] = C[i][j] + A[i][k] * B[k][j];
        }
    }
}
```

**Polynomial Time Algorithms:** An algorithm is called a polynomial time algorithm if

**Size of the Input and Number Theoretic Algorithms** Consider the brute-force algorithm to determine whether a given integer is prime? PRIMES is in P.

**Remark:** If the input for a number theoretical algorithm is integer $n$, then the size of the input is taken to be ................................................................. which is ..........

**Example:** Computational Complexity of Addition, Multiplication and Division