The natural logarithm of \((1+x)\), i.e. \(\ln(1+x)\) for \((-1 < x < 1)\) can be evaluated by the approximation:

\[ p(x) = \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots + \frac{x^n}{n} \]

Consider the variable \(x\) to be of type float. The value of \(x^i\) is computed by a function `pow(x,i)` using \((i -1)\) float multiplications. The algorithm is:

```c
float p = 0; float s = -1.0;
for (int i = 1; i <= n; i++)  { s = -s ;    p = p + pow(x,i) / i * s ; }
```

(a) What is the number of float arithmetic operations for a single iteration \((i)\) of the loop?

(b) What is the total number of such operations \(T(n)\) done by the algorithm, and what is its complexity (Big-O)?

(c) A faster algorithm is:

```c
float p = 0; float s = -1.0;
for (int i = 1; i <= n; i++)  { s = -s * x ;   p = p + s / i  ; }
```

Why is this algorithm faster than the direct one? (explain by comparing the two Big-O's).

{The sum of integers from 1 to \(n\) is equal to \(n(n+1) / 2\)}

---

Find the Big-O for the following number of operations:

1. \(T(n) = n^3 + 100n \log n + 500\)
2. \(T(n) = 4^n + n^3\)
3. \(T(n) = 0.01n \log n + 8 \log n\)
4. \(T(n) = 1 + 3 + 9 + 27 + \ldots + 3^{n-1}\)

The multiplication of two square matrices \(A_{n \times n}\) and \(B_{n \times n}\) produces a matrix \(C_{n \times n} = A*B\) whose elements are given by:

\[ C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj} \quad i, j = 0,\ldots,n-1 \]

Write the algorithm to receive \(A, B\), and return \(C\) using the above definition. Find the number of arithmetic operations done by this algorithm as a function of \(n\).

Suppose program (A) takes \(2^n/16\) units of time and program (B) takes \(16n^2\) units:

1. For what values of \((n)\) does program (A) take less time than (B)?
2. For each of these programs, how large a problem can be solved in \(2^{20}\) time units?