Definition: $(n) = O(g(n))$

$if\ f(n) \le C * g(n) \quad for\ all\ n \ge k \qquad rewrite\ to\ find\ C{:}\ \frac{f(n)}{g(n)} \le C$

$where\ C > 0\ and\ k > 0$

1) $T(n) = 5n^4 + 6n^2 + 2n + 4 \le 5n^4 + 6n^4 + 2n^4 + 4n^4$
   $C = 16\ and\ n_0 = 1$ so $T(n)\ is\ O(n^4)$

2) $T(n) = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{3n^3+3n^2+n}{6} < \frac{3n^3+3n^3+n^3}{6}$
   $C = \frac{7}{6}\ and\ n_0 = 1$ so $T(n) is\ O(n^3)$

3) $T(n) = 2^{n+1}$ Show $T(n)\ is\ O(2^n)$ $f(n) < C *\ g(n)$
   $\frac{2^{n+1}}{2^n} \le C$ $\frac{2^n * 2}{2^n} \le C$ $2 \le C$
   $C = 2\ and\ n_0 = 1$ So $T(n)\ is\ O(2^n)$

4)

	Finishes in:
$n = 10$	
$(\log n) = 3 * 10^{-12}$	less than one second
$(n) = 10^{-11}$	less than one second
$(n \log n) = 3 * 10^{-11}$	less than one second
$(n^2) = 10^{-10}$	less than one second
$(n^3) = 10^{-9}$	less than one second
$(2^n) = 10^{-9}$	less than one second
$O(n!) = 10^{-6}$	less than one second
$n = 10^3$	
$(\log n) = 10^{-12}$	less than one second
$(n) = 10^{-9}$	less than one second
$(n \log n) = 10^{-8}$	less than one second
$(n^2) = 10^{-6}$	less than one second
$(n^3) = 10^{-3}$	less than one second
$(2^n) = 10^{288}$	hopeless
$O(n!) = 4 * 10^{2567}$	hopeless
$n = 10^6$	
$(\log n) = 2 * 10^{-11}$	less than one second
$(n) = 10^{-6}$	less than one second
$(n \log n) = 2 * 10^5$	two days
$(n^2) = 1$	one seconds

$(n^3) = 10^6$	about a day
$(2^n) = 10^{299988}$	hopeless
$O(n!) = 8 * 10^{5565708}$	hopeless

$n = 10^9$

$(\log n) = 3 * 10^{-11}$	less than one second
$(n) = 10^{-3}$	less than one second
$(n \log n) = 3 * 10^2$	5 minutes
$(n^2) = 10^6$	about a day
$(n^3) = 10^9$	almost two days
$(2^n) = 10^{299999988}$	hopeless
$O(n!) = 10^{8565705523}$	hopeless

$n = 10^{12}$

$O(\log n) = 4 * 10^{-11}$	less than one second
$O(n) = 1$	one second
$O(n \log n) = 40$	less than a minute
$O(n^2) = 10^{12}$	some tens of thousands of years
$O(n^3) = 10^{24}$	hopeless
$O(2^n) = 10^{2999999999988}$	hopeless
$O(n!) = 10^{11565705518104}$	hopeless

$n = 10^{15}$

$O(\log n) = 5 * 10^{-11}$	less than one second
$O(n) = 10^3$	less than an hour
$O(n \log n) = 5 * 10^4$	less than a day
$O(n^2) = 10^{18}$	hopeless
$O(n^3) = 10^{33}$	hopeless
$O(2^n) = 10^{299999999999988}$	hopeless
$O(n!) = 10$^10^16.163	hopeless

$n = 10^{18}$

$(\log n) = 6 * 10^{-11}$ less than one second

$(n) = 10^{6}$ ten days

$(n \log n) = 6 * 10^{7}$ less than 2 years

$(n^2) = 10^{24}$ hopeless

$(n^3) = 10^{42}$ hopeless

$(2n) = 10^{2999999999999998 8}$ hopeless

$O(n!) = 10\hat{}10\hat{}19.244$ hopeless

$n = 10^{21}$

$(\log n) = 7 * 10^{-11}$ less than one second

$(n) = 10^{9}$ about 3 decades

$(n \log n) = 7 * 10^{10}$ about 3 centuries

$(n^2) = 10^{30}$ hopeless

$(n^3) = 10^{51}$ hopeless

$(2n) = 10^{29999999999999999988}$ hopeless

$O(n!) = 10\hat{}10\hat{}22.3131$ hopeless

6) *Included under 5)*

$$\sum_{i=0}^{n} c_i x^n = c_n x^n + c_{n-1} x^{n-1} + \cdots + c_2 x^2 + c_1 x + c_0$$

7)

```
// The output of this is p ,
// the polynomial evaluated at x.
p = c_0
for i = 1 to n
        term = c_i
        for j = 1 to i
                term = term * x
        p = p + term
```

8)

```
// The output of this is p ,
// the polynomial evaluated at x.
p = 0
for (i = n; i > 0; i - -)
        p = x · ( p + c_i)
return p + c_0
```

9) Horner's rule is more efficient for polynomials

10) The best case is O(n) and the worst case is O(n^2). The only time the best case will happen, as this is a sorting algorithm, will be when already sorted data is evaluated. Otherwise the runtime will be represented as O(n^2) and since there is no real reason to sort data that is already ordered the worst case will also be the average/usual case.

11)

```
//assume you are sorting an array of length n. First element has index 1.
//selection sort produces an array of sorted integers in ascending order
1    for k = 1 to n - 1
2            indexOfMin = k
3            for i = k + 1 to n
4                    if (a[i] < a[indexOfMin])
5                            indexOfMin = i
6            if (indexOfMin ≠ k)
7                    tmp = a[k]
8                    a[k] = a[indexOfMin]
9                    a[indexOfMin] = tmp
```

Line Number	**Cost**	**Best Case**	**Worst Case**
1	C_1	(n – 2 + 1) + 1 = n	n
2	C_2	n – 1	n – 1
3	C_3	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
4	C_4	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$
5	C_5	0	$\frac{n(n-1)}{2}$
6	C_6	n – 1	n – 1
7	C_7	0	n – 1
8	C_8	0	n – 1
9	C_9	0	n – 1

Big O of this algorithm for both best case and worst case is $O(n^2)$ so the average case will be the same