Analysis of simple algorithms

Count the number of steps executed for each algorithm.

Algorithm 1: input array b of size n

```
s = 0
t = 1
for i = 1 to n
  s = s + t * b[n-i+1]
  t = 2 * t
```

Answer: $3n + 2$

Algorithm 2: input n

```
s = 0
for i = 1 to n
  for j = 1 to n
    s = s + i*j
```

Answer: $2n^2 + n + 1$

Algorithm 3: input 3-dim array A of size n × n × n

```
for i = 1 to n
  for j = 1 to n
    for k = 1 to n
      A[i, j, k] = A[i, j, k] + A[j, k, i] * A[k, i, j]
```

Answer: $2n^3 + n^2 + n$

Algorithm 4: input n

```
s = 0
t = 1
for i = 1 to n
  for j = 1 to i
    t = t*j
    s = s + t
```

When i = 1, j goes from 1 to 1	1
When i = 2, j goes from 1 to 2	2
When i = 3, j goes from 1 to 3	3
etc.	
When i = n, j goes from 1 to n	n

Answer: $3n(n + 1)/2 + n + 2$

Algorithm 5: input n

```
s = 0
while (n > 1)
  s = s + n
  n = n / 2
```

Count 3 statements (guard and two statements in the body) **each** time through the loop
repeatedly divides n by 2 until n <= 1
how many times through the loop?
Let's say k represents the number of times through the loop.
continues until n / (2^k) = 1. Solve for k.
k = **floor** (lg n)

The guard is actually executed one extra time
Total numbers of steps = 1 + 3*k + 1

Answer: 2 + 3 floor (lg n)

Algorithm 6: input n

```
p = 1
for i = 1 to n
  for j = 1 to i
    for k = 1 to j
      p = p * (i + j + k)
```

Answer: 1 + n + n(n+1)/2 + 3 floor (lg n) + 2* $(\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n)$

Count 1 for the very first assignment statement
The outside for loop is executed once for each value of i (1, 2, … n) n times
The middle for loop is executed 1 + 2 + 3 + … + n times
The inside for loop is executed…something like n^3 since it is triply nested, right??

Example: If n = 5:

i = 1
 j = 1
  k = 1

i = 2

j = 1	j = 2
k = 1	k = 1
	k = 2

i = 3

j = 1	j = 2	j = 3
k = 1	k = 1	k = 1
	k = 2	k = 2
		k = 3

etc.
NOTE:
We need to count the number of triples (i, j, k) where $1 \le k \le j \le i \le n$
Consider the following example to illustrate a way of counting these triples:

Suppose n = 5: The three vertical bars stand for k, then j, then i (in that order). Count the number of stars from the **beginning** of the string up to each bar to determine the values of k, j and i.

* * | * | * * |	stands for k = 2, j = 3, i = 5
* * || * * | *	stands for k = 2, j = 2, i = 4

We want to count how many strings there are like this (5 stars and 3 bars). But we need to modify this counting…

Since $k \ge 1$, the first section must <u>**always**</u> have at least one *. So we will take the first * out, but just pretend it is always there (to be sure that k >= 1).

Our stars and bars are now like this…

* | * | * * |	stands for k = 2, j = 3, i = 5 (with an invisible * in front)
* || * * | *	stands for k = 2, j = 2, i = 4 (with an invisible * in front)

Now the question is how many strings have 4 stars and 3 bars?

_ _ _ _ _ _ _

If n = 5, we can count the number of i, j, k triples by laying down a sequence of 4 + 3 = 7 blanks (4 blanks for the *'s and 3 blanks for the |'s to represent k, then j, then i.) Then choose any 3 blanks to represent the |'s.

To generalize this for any n, we need n-1 stars (the 1 in the front is "invisible") and 3 bars. Lay down a sequence of n – 1 + 3 = n + 2 blanks and choose any 3 of them to be the |'s.

$$\binom{n+2}{3} = \frac{(n+2)!}{3!(n-1)!} = \frac{(n+2)(n+1)n}{6} = \tfrac{1}{6}n^3 + \tfrac{1}{2}n^2 + \tfrac{1}{3}n \quad = O(n^3)$$