Question 8

1. What does this algorithm compute?

   Answer: It computes $1 + 2^2 + 3^2 + ... + n^2$

2. What is the input size?

   Answer: n

3. What is the basic operation?

   Answer: Multiplication

4. Set up a recurrence and a initial condition and find the number of times the basic operation is executed.

   Answer:
   $M(n) = M(n-1) + 1$, $M(1) = 0$
   Solution: $M(n) = n - 1$

5. What is the efficiency class of this algorithm.

   Answer: $\Theta(n)$

Question 9

a. The recurrence relation with initial condition is
$Q(n) = Q(n-1) + 2n - 1$ for $n > 1$, $Q(1) = 1$

Computing the first few terms of the sequence yields the following:
$Q(2) = Q(1) + 2 \times 2 - 1 = 1 + 2 \times 2 - 1 = 4;$
$Q(3) = Q(2) + 2 \times 3 - 1 = 4 + 2 \times 3 - 1 = 9;$
$Q(4) = Q(3) + 2 \times 4 - 1 = 9 + 2 \times 4 - 1 = 16;$
It appears that $Q(n) = n^2$. (We check this hypothesis by substituting this formula into the recurrence equation and the initial condition. The left hand side yields $Q(n) = n^2$. The right hand side yields $Q(n-1) + 2n - 1 = (n-1)^2 + 2n - 1 = n^2$.)

b. The recurrence relation with initial condition is
$M(n) = M(n-1) + 1$ for $n > 1$, $M(1) = 0$.

Solving it by backward substitutions (it is almost identical to the factorial example – see Example 1 in the section) generates solution $M(n) = n - 1$.

c. Let $C(n)$ be the number of additions and subtractions made by the algorithm. The recurrence with initial condition for $C(n)$ is $C(n) = C(n-1) + 3$ for $n > 1$, $C(1) = 0$. Solving it by backward substitutions generates solution $C(n) = 3(n-1)$.

Note: If we dont include in the count the subtractions needed to decrease n, the recurrence will be $C(n) = C(n-1) + 2$ for $n > 1$, $C(1) = 0$. The solution is $C(n) = 2(n-1)$. This is also correct.

Question 10

1. What does the algorithm compute?

   Answer: Search (binary search).

2. How is the input size n expressed in terms of the parameters?

   Answer: $n = r - l + 1$

3. Assume that after comparison of K with $A[m]$, the algorithm can determine whether K is smaller than, equal to, or larger than $A[m]$. Set up a recurrence (with an initial condition) for the worst case of this algorithm. Solve the recurrence for $n = 2^k$, and determine the Θ efficiency class.

   Answer: Recurrence for the worst case: $C(n) = C(\lfloor n/2 \rfloor) + 1$, for $n > 1$, $C(1) = 1$

   When $n = 2^k$, the recurrence is $C(2^k) = C(2^{k-1}) + 1$. By using the backward substitution method, we have $C(2^k) = C(2^{k-1}) + 1 = C(2^{k-2}) + 2 = ... = C(2^{k-i}) + i$.

   Or we can write as $C(n) = C(n/2) + 1 = C(n/2^2) + 2 = ...$ and have $C(n) = C(n/2^i) + i$.

   Let $i = k$. Use the initial condition, and use $k = \log_2 n$. We have
   $C(n) = 1 + k = 1 + \log_2 n \in \Theta(\log n)$

4. What is the Θ efficiency class when $n \neq 2^k$? Why?

   Answer: According to the Smoothness Rule, the efficiency class is also $\Theta(\log n)$ when $n \neq 2^k$.