Taylor Whitlock
Section 002
HW4

2.4)

Algorithm A: O(n^2.32192809489)
Algorithm B: O(2^n)
Algorithm C: O((n^2)log(n))

I would choose Algorithm C

2.5)

a. Θ(n^0.631); A/(B^D) → 2/(3^0) → Θ(n^0.631)
b. Θ(n^1.161); A/(B^D) → 5/(4^1) → Θ(n^(log4(5))) → Θ(n^1.161)
c. Θ(nlog(n)); A/(B^D) → 7/(7^1) → Θ((n^1)log(n)) → Θ(nlog(n))
d. Θ((n^2)log(n)); A/(B^D) → 9/(3^2) → Θ((n^2)log(n))
e. Θ((n^3)log(n)); A/(B^D) → 8/(2^3) → Θ((n^3)log(n))

2.17)

```
def div_conq(A, offset):    # A is a sorted array of integers, offset is to keep track of the index in the original array
    if len(A) == 1:
        return True if A[0] == offset else False
    i = len(A) // 2
    if A[i] == i + offset:
        return True
    elif A[i] < i + offset:
        return div_conq(A[:i], offset)
    else:
        return div_conq(A[i:], offset + i)
```