1.

a)

$$\forall n \geq 1 : 5n^2 - 2n + 26 \leq 29n^2 \implies 5n^2 - 2n + 26 \in O(n^2)$$

b)

$$\forall n > a : \frac{a^n}{n!} < \frac{a}{1} * \frac{a}{1} \dots \frac{a}{a} * 1 \dots 1 * \frac{a}{n} = \frac{a^a}{n} \implies \lim_{n \to \infty} \frac{a^n}{n!} = 0 \implies a^n \in O(n!)$$

c)

$$\forall n \geq 1 : 2^{n+a} = 2^a * 2^n = C * 2^n \implies 2^{n+a} \in O(2^n)$$

d)

$$\forall a \geq 1 : \log_a n = \frac{\log_2 n}{\log_2 a} \implies f(n) \in O(\log_a n)$$

e)

$$\begin{aligned} 2^n &\leq n^{\log^2 n} * C \\ n &\leq \log_2(n^{\log^2 n}) + \log_2(C) \\ n &\leq \frac{\log_2(n^{\log^2 n})}{\log_2 n} * \log_2 n + \log_2(C) \\ n &\leq \log^2 n * \frac{\log_2 n}{\log_2 10} * \log_2 10 + C' \\ n &\not\leq \log^3 n * \log_2 10 + C' \implies 2^n \notin O(n^{\log^2 n}) \end{aligned}$$

f) assume there is some constant such that $2^{2^{n+1}} \leq C * 2^{2^n}$ then

$$\begin{aligned} \log_2(\log_2(2^{2^{n+1}})) &\leq \log_2(\log_2(C * 2^{2^n})) \\ n + 1 &\leq \log_2(\log_2(C) + 2^n) \\ n + 1 &\leq n + \log_2(\frac{\log_2 C}{2^n} + 1) \\ n + 1 &\leq n + \log_2(\frac{C'}{2^n} + 1) \\ 1 &\not\leq \log_2(\frac{C'}{2^n} + 1) \implies 2^{2^{n+1}} \notin O(2^{2^n}) \end{aligned}$$

2.

a)

$$\sum_{i=0}^{n-1} i = \frac{(n-1)(n-1+1)}{2} \implies O(n^2)$$

b)

the best case time complexity is $O(n)$, and i happens when the median of the array is the first element. the worst case time complexity is $O(n^2)$ and that happens when the last element in the array is the median since it will loop through the entire array n times.

3.

```
start = 0;
end = n;
i = end/2;
flag = false;
while(!flag){
    flag = A[i] == 1 && A[i-1] == 0;
    if(!flag){
        if(A[i]==0){
            start = i;
            i = start + (end-start)/2;
        } else{
            end = i;
            i = start + (end-start)/2;
        }
    }
}
return i;
```

This continually divides the remaining length in half, resulting in $\log_2 n$ elements being checked. which makes $O(\log n)$ the worst case time complexity.

4.

for i in [1,k]
    $O(i * n)$

$$n * k + n * (k - 1) + \ldots + n * 2 + n = \sum_{i=1}^{k} ni = n\frac{k(k+1)}{2} \implies O(k^2 n)$$

5.

a) 314,606,891, and 817,504,243 are the numbers i chose the GCD ran in 2085 milliseconds and the fastGCD ran in less than 1 millisecond
b) 5,915,587,277, and 5,463,458,053 are the numbers i chose. GCD ran in 36,177 milliseconds and the fastGCD ran in less than 1 millisecond
c)
Case 1) $b \geq \frac{a}{2}$ after an iteration $a = b, b = a\%b$ making b at most $\frac{b}{2}$
Case 2) $b \leq \frac{a}{2}$ after an iteration $a = b, b = a\%b$ making a at most $\frac{a}{2}$
since both cases end it a 50% reduction the algorithm is $O(log_2 max(a,b))$ but since $log_2 a, log_2 b$ = number of bits in a and b the algorithm is $O(n)$ where n is the number of bits in a and b.