Problem 2-2

a.
You need to show that the elements of A' form a permutation of the elements of A.

b.
Loop invariant: At the start of each iteration of the for loop A[j] is the smallest value in the subarray A[j:n] is a permutation of the values that were in A[j:n] at the time that the loop started.

Initialization:
Initially, j=n and the subarray A[j:n] consist of the single element A[n]. The Loop invariant trivially holds

Maintenance:
Consider an itteration for a given value of j. By the loop invariant, A[j] is the smallest value in A[j:n] A[j] and A[j-1] if A[j] is less than A[j-1] if A[j] is less than A[j-1], and so A[j-1] will be the smallest value in A[j-1:n] afterward. Since the only change to the subarray A[j-1:n] is this possible exchange, and the subarray A[j:n] is a permutation of the values that were in A[j:n] at the time that the loop started, we see that A[j-1:n] is a permutation of the values that were in A[j-1:n] at the time that the loop started. Decrementing j for the next iteration maintains the invariant.

Termination:
The loop terminates when j reaches i. By the statement of the loop invariant, A[i] is the smallest value in the subarray A[i:n], and A[i:n] is a permutation of the value in the subarray A[i:n], and A[i:n] is a permutation of the values that were in A[i:n] at the time that the loop started

c.

Initilization: Before the first iteration of the loop, $i=1$, the subarray $A[1 : i-1]$ is empty, and so the loop invariant vacouously holds.

Maintenance:

Consider an iteration for a given value of $i$. By the loop invariant, $A[1 : i-1]$ consists of the $i-1$ smallest values in $A[1 : n]$, in sorted order. Therefore, $A[i : n] \leq A[i]$. part (b) shows that after executing the for loop, $A[i]$ is the smallest value in $A[i : n]$, in sorted order. Moreover, since for loop of permutes $A[i : n]$, the sub array $A[i+1 : n]$ consists of the $n-i$ remaining values originally in $A[1 : n]$

termination:

The for loop of the lines terminates when $i=n$, so that $i-1 = n-1$. By the statement of the loop invariant, $A[i : n-1]$ is the subarray $A[1 : n-1]$, and it consists of the $n-1$ smallest values originally in $A[1 : n]$, in sorted order. The remaing element must be the largest value in $A[1 : n]$, and it is in $A[n]$. therefore, the entire array $A[1 : n]$ is sorted.

d. for given value of $i$, this loop makes $n-i$ iterations, and $i$ takes on the valus, $1, 2 \ldots n-1$. The total number of iterations, therefore, is

$$\sum_{i=1}^{n-1}(n-i) = \sum_{i=1}^{n-1} n - \sum_{i=1}^{n-1} i = n(n-1) - \frac{n(n-1)}{2} = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2}.$$

Thus, the running of bubble sort is $\Theta(n^2)$ in all cases.

The worst case running time is same as insertion sort.

Problem 2-4

a. The inversions are $(1,5),(2,5),(3,4),(3,5),(4,5)$

note that inversions are specified by indices rather than by the values in the array.

b. the array with elements drawn from $\{1,2,\cdots n\}$ with the most inversions is $\langle n, n-1, n-2, \cdots 2, 1\rangle$. ~~For~~ $\forall\ 1 \le i < j \le n$, there is inversion $(i,j)$. The number of such inversion is $\binom{n}{2} = n(n-1)/2$.

c. suppose that the array A starts out with an inversion $(k,i)$, Then $k < i$ and $A[k] > A[i]$. A the time that the outer for loop of line 1-8 set $key = A[i]$, the value that started in $A[k]$ is still somewhere to the left of $A[i]$. That is, it's in $A[j]$, where $1 \le j < i$, and so the inversion has become $(j,i)$. Some iteration of the while loop of lines 5-7 moves $A[j]$ one position to the right. Line 8 will eventually drop key to the left of this element, thus ~~[illegible]~~ eliminating the inversion. Because line5 moves only elements that are greater than key it moves only elements that correspond to inversions. In other word, each iteration of the while loop of lines 5-7 corresponds to the elimination of one inversion.

d.

```
Merge Inversions (A,p,q,r)
    n_L = q - p + 1
    n_R = r - q
    let L[0:n_L - 1] and R[0:n_R - 1] be new arrays
    for i = 0 to n_L - 1
        L[i] = A[p + i - 1]
    for j = 0 to n_R - 1
        R[j] = A[q + j]
    i = 0
    j = 0
    k = p
    inversions = 0
    while i < n_L and j < n_R
        if L[i] ≤ R[j]
            inversions = inversions + n_L - i
            A[k] = L[i]
            i = i + 1
        else A[k] = R[j]
            j = j + 1
        k = k + 1
    while i < n_L
        A[k] = L[i]
        i = i + 1
        k = k + 1
    while j < n_R
        A[k] = R[j]
        j = j + 1
        k = k + 1
    return inversions
```

↓ more code

```
COUNT-Inversions (A,p,r)
    inversions = 0
    if p < r
        q = ⌊(p+r)/2⌋
        inversions = inversions + Count-Inversions(A,p,q)
        inversions = inversions + Count-Inversions(A,q+1,r)
        inversions = inversions + Merge-Inversions(A,p,q,r)
    return inversions
```

problem 3-n

a. $f_0(n) = n$. Since $f(n)$ just subtracts 1, the answer is how many times you subtract 1 from $n$ before reaching 0, which is just $n$.

b. $f_1(\lg n) = \lg^* n$. This answer comes directly from the definition of the iterated logarithm function.

c. $f_1(n/2) = \lceil \lg n \rceil$. This result is easily shown by induction for $n$ a power of 2. The ceiling function handles values of $n$ between power of 2.

d. $f_2(n/2) = \lceil \lg n \rceil - 1$. Take the answer from part c, but one fewer time.

e. $f_2(\sqrt{n}) = \lceil \lg \lg n \rceil$. Define $m = \lg n$, so that $n = 2^m$. The problem then become determining $f_1((2^m)^{1/2})$. (It is $f_1(2^{m/2})$ instead of $f_2(2^{m/2})$ because $n=2$ implies $m=1$. By part c, the answer is

$\lceil \lg m \rceil = \lceil \lg \lg n \rceil$.

f. $f_1(\sqrt{n})$ is undefined. No matter how many times you take the square root of $n > 1$, you will never reach 1.

g. $\lceil \log_3 \log_3 n \rceil \leq f_2(n^{1/3}) \leq \lceil \log_3 \log_3 n \rceil + 1$. Similiar to the solution to part (e). Let $n = 3^m$ and $m = \log_3 n$, so that the problem becomes finding $f_{\log_3 2}(3^{m/3})$. As in part (c), the number of times you divide by 3 before reaching 1 is $\lceil \log_3 m \rceil = \lceil \log_3 \log_3 n \rceil$. Since $\log_3 2 < 1$, however, you might need to iterate one more time to reach $\log_3 2$.

Probelem 3.3

$2^{2^{n+1}}$

$2^{2^n}$

$(n+1)!$

$n! \to \Theta(n^{n+1/2}e^{-n})$

$e^n \to 2^n(e/2)^n = \omega(n2^n) \Rightarrow \omega(n)$

$n \cdot 2^n$

$2^n$

$(3/2)^n$

$(\lg n)^{\lg n} = n^{\lg\lg n} \to a^{\log_b c} = c^{\log_b a}$

$(\lg n)! \Rightarrow \omega(n^3)$

$n^3$

$n^2 = 4^{\lg n}$

$n\lg n, \lg(n!) \to \lg(n!) \to \Theta(n \lg n)$

$n = 2^{\lg n}$

$(\sqrt{2})^{\lg n} (= \sqrt{n})$

$2^{\sqrt{2\lg n}}$

$\lg^2 n$

$\ln n$

$\sqrt{\lg n}$

$\ln\ln n$

$2^{\lg^* n}$ $\quad$ $\lg^*(\lg n) = (\lg^* n) - 1$

$\lg^* n, \lg^*(\lg n)$

$\lg(\lg^*)n \to (1 \sim 5)$

$n^{1/\lg n} = 2$, ~~or~~ 1