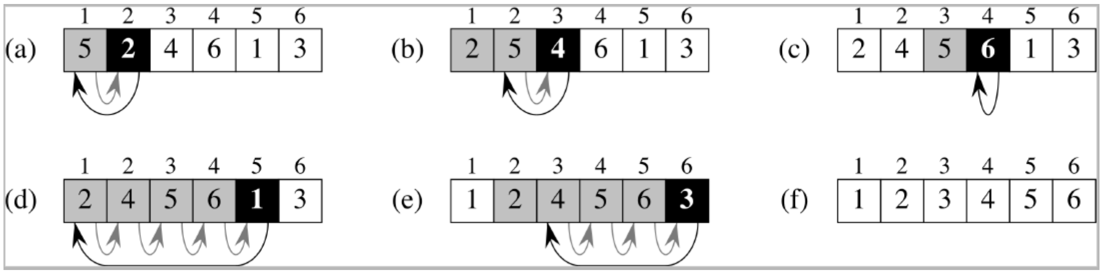
1. Use the following as a model,



illustrate the operation of INSERTION-SORT on the array *A* = {31, 41,59, 26, 41,58}. Rewrite the INSERTION-SORT procedure to sort into nonincreasing instead of nondecreasing order. you must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points each, 20 points in total.)

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

None decreasing:

1)

1	2	3	4	5	6
31	41	59	26	41	58

2)

1	2	3	4	5	6
31	41	59	26	41	58

3)

1	2	3	4	5	6
31	41	59	26	41	58

4)

1	2	3	4	5	6
26	31	41	59	41	58

5)

1	2	3	4	5	6
26	31	41	41	59	58

6)

1	2	3	4	5	6
26	31	41	41	58	59

Rewrite

```
INSERTION - SORT (A)
    for j = 2 to A.length
        key = A[j]
        i = j - 1
        while i > 0 and A[i] < key
            A[i+1] = A[i]
            i = i - 1
        A[i+1] = key
```

2. **Binary Addition of Integers**: Given two integers *a* and *b*, their binary expansions are shown below.

$$a = (a_{n-1}a_{n-2}\ldots a_1a_0)_2,\ b = (b_{n-1}b_{n-2}\ldots b_1b_0)_2$$

To compute the sum of *a* and *b* in binary form, add the corresponding pairs of bits with carries when they occur.

- First add their rightmost bits. This gives

$$a_0 + b_0 = c_0 \cdot 2 + s_0,$$

  - s_0 is the rightmost bit in the binary expansion of *a* + *b* and
  - c_0 is the *carry***.**
- Then add the next pair of bits and the carry.

$$a_1 + b_1 + c_0 = c_1 \cdot 2 + s_1$$

  - s_1 is the next bit (from the right) in the binary expansion of *a* + *b*, and
  - c_1 is the carry.
- Continue this process, adding the corresponding bits in the two binary expansions and the carry, to determine the next bit from the right in the binary expansion of *a* + *b*.
- At the last stage,

  $a_{n-1} + b_{n-1} + c_{n-2} = c_{n-1} \cdot 2 + s_{n-1}$

  The leading bit of the sum is $s_n = c_{n-1}$. This procedure produces the binary expansion of the sum,

$$\begin{array}{r} c_{n-1}c_{n-2}\ \ldots\ c_1\,c_0 \\ a_{n-1}a_{n-2}\ldots a_1a_0 \\ b_{n-1}b_{n-2}\ldots b_1b_0 \\ \hline s_{n-1}\,s_{n-2}\ \ldots\ s_1\,s_0 \end{array}$$

Write pseudocode for adding two integers in binary expansions **formally**. Store the two binary integers and their sum in arrays. Illustrate your algorithm using this following two integers: *a* = *(*1110*)*$_2$ and *b* = *(*1011*)*$_2$. You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points)

Binary - Add (A, B)

$n = A.length$

Let sum, C be new array sum$[0 \ldots n-1]$, C$[0 \ldots n-1]$

for i from 0 to $n-1$

if $i = 0$

$sum[i] = (A[i] + B[i]) \% 2$

$C[i] = (A[i] + B[i]) / 2$

else

$Sum[i] = (A[i] + B[i] + C[i-1]) \% 2$

$C[i] = (A[i] + B[i] + C[i-1]) / 2$

$a\ (1110)_2 + b\ (1011)_2$

$$
\begin{array}{rccccc}
 & 1 & 1 & 1 & 0 & \\
 & & 1 & 1 & 1 & 0 \\
+ & & 1 & 0 & 1 & 1 \\
\hline
 & 1 & 1 & 0 & 0 & 1
\end{array}
$$

$a_0 = 0,\ b_0 = 1,\ a_0 + b_0 = 0 \times 2 + 1 = C_0 \times 2 + S_0 \rightarrow S_0 = 1,\ C_0 = 0$

$a_1 = 1,\ b_1 = 1,\ a_1 + b_1 + C_0 = 1 \times 2 + 0 = C_1 \times 2 + S_1 \rightarrow S_1 = 0,\ C_1 = 1$

$a_2 = 1,\ b_2 = 0,\ a_2 + b_2 + C_1 = 1 \times 2 + 0 = C_2 \times 2 + S_2 \rightarrow S_2 = 0,\ C_2 = 1$

$a_3 = 1,\ b_3 = 1,\ a_3 + b_3 + C_2 = 1 \times 2 + 1 = C_3 \times 2 + S_3 \rightarrow S_3 = 1,\ C_3 = 1$

$C_3 = S_4 = 1$

$S = (11001)_2$

3. Express the following functions in terms of Θ- notation.

$(n^2 + 8)(n + 1)$, $(n \log n + n^2)(n^3 + 2)$, and $(n! + 2^n)(n^3 + \log(n^2 + 1)$

You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (5 points each, 15 points in total.)

a. $(n^2 + 8)(n + 1)$

$$(n^2+8)(n+1)$$

$$= n^3 + 8n + n^2 + 8$$

$$= n^3 + n^2 + 8n + 8$$

$$\theta(n^3+n^2+8n+8) = \theta(n^3)$$

b. $(n \log n + n^2)(n^3 + 2)$

$$(n\log n + n^2)(n^3+2)$$

$$= n^3\log n + n^5 + 2n\log n + 2n^2$$

$$\because n^5 > n^3\log n > 2n^2 > 2n\log n$$

$$\therefore \theta[(n\log n + n^2)(n^3+2)] = \theta(n^5)$$

c. $(n! + 2^n)(n^3 + \log(n^2 + 1))$

$$= n^3 n! + n^3 2^n + \log(n^2+1)n! + \log(n^2+1)2^n$$

$$\because n^3 n! > \log(n^2+1)n! > n^3 2^n > \log(n^2+1)2^n$$

$$\therefore \theta((n!+2^n))(n^3+\log(n^2+1)) = (n^3 n!)$$

4. We often use a *loop invariant* to prove that an algorithm gives the correct answer. To use a ***loop invariant*** to prove correctness, we must show three things about it:

   1)**Initialization:** It is true prior to the first iteration of the loop.

   2)**Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration.

   3)**Termination:** When the loop terminates, the invariant (usually along with the reason that the loop terminated) gives us a useful property that helps show that the algorithm is correct.

Let's look at the following Bubble sort algorithm.

```
        BUBBLESORT(A)
outer:  for i = 1 to A.length − 1
inner:      for j = A.length downto i + 1
                if A[j] < A[j − 1]
                    exchange A[j] with A[j − 1]
```

1) Bubble sort is a sorting algorithm that works by repeatedly exchanging adjacent elements that are out of order. Let *A'* denote the output of BUBBLESORT(A). To prove that BUBBLESORT is correct, we need to prove that it terminates and that $A'[1] \le A'[2] \le \ldots \le A'[n]$, where n = *A.length*. In order to show that BUBBLESORT actually sorts, what else do we need to prove?
2) State precisely a loop invariant for the for loop *inner*, and prove that this loop invariant holds using the above-mentioned structure of the loop invariant.
3) Using the termination condition of the loop invariant proved in part 2), state a loop invariant for the *for* loop *outer* that will allow you to prove $A'[1] \le A'[2] \le \ldots \le A'[n]$, where *n* = *A.length*. Prove that this loop invariant holds using the above-mentioned structure of the loop invariant. You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (5 points for #2), 5 points for each of the three parts (**Initialization, Maintenance, and Termination)** in #3). 20 points in total.)

(1) A' must contain all of the elements of array A.
It can't be less than or more than the numbers of A.

(2) Loop invariant: A[j] is the smallest one in subarray A[j...length]

Initialization: Only A[j] in subarray, it's the smallest one.

Maintenance: Before each iteration, A[j] is the smallest one.
After iteration, A[j-1] is the smallest one, j become to j-1,
so A[j] still the smallest one.

Termination : Terminate when $j = i$, $A[j] = A[i]$ is the smallest one of $A[i \ldots length]$

3) loop invariant: The subarray contains $(i-1)$ smallest elements

Initialization: $i-1=0$, the subarray is empty

Maintenance : Before iteration, subarray contains $(i-1)$ smallest elements
After iteration, $A[i]$ is the smallest one of $A[i \ldots length]$,
then $i{+}{+}$, subarray still contains $(i-1)$ smallest elements

Termination: Terminate when $i = length$, subarray contain $(length-1)$ smallest element, all elements in array are sorted decreasingly.

5. Suppose that a list contains integers that are in order of largest to smallest and an integer can appear repeatedly in this list.

   1) Devise an algorithm that locates all occurrences of an integer *x* in the list.

   2) Estimate the number of comparisons used.

You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points each, 20 points in total.)

Locate-Occurrence (A, x)

1)

sum = 0
n = A.length
for i from 0 to n
    if A[i] = x
        print A[i]

2) compare from 1st to last one

number of comparision : n

6. Prove that $n^3 - 91n^2 - 7n - 14 = \Omega(n^3)$. You must show/explain how you arrived at the constants, and clearly specify the positive constants c and n_0. Simply stating the answers will result in 0 points awarded. (10 points)

$n^3 - 91n^2 - 7n - 14 \geq 1 \times n^3$

when $91n^2 + 7n + 14 \geq 0$

For $n \geq 0$, $91n^2 + 7n + 14 > 0$

Therefore, $c = 1$, $n_0 = 0$

Hence $n^3 - 91n^2 - 7n - 14 = \Omega(n^3)$

7. Prove that $27n^2 + 18n = \Theta(0.5n^2 - 100)$. You must show/explain how you arrived at the constants, and clearly specify the positive constants c_1, c_2, and n_0. Simply stating the answers will result in 0 points awarded. (10 points)

step1:

$27n^2 + 18n$ and $0.5n^2 - 100 \geq 0$

$0.5n^2 - 100 \geq 0$

$0.5n^2 \geq 100$

$n^2 \geq 200$

$\because n > 0$

$\therefore n \geq \sqrt{200} = 10\sqrt{2}$

Step2: For the lower bound

$27n^2 + 18n \geq c_1(0.5n^2 - 100)$

For all $n \geq 10\sqrt{2}$, choose $c_1 = 54$, $27n^2 + 18n \geq 54(0.5n^2 - 100)$

$n_0 = 10\sqrt{2}$, $c_1 = 54$

step 3: For the upper bound

$27n^2 + 18n \leq c_2(0.5n^2 - 100)$

$27n^2 + 18n \leq \frac{1}{2}c_2n^2 - 100c_2$

when $c_2 = 54$, $27n^2 + 18n \leq 27n^2 - 5400$, not exists

when $c_2 = 60$, $27n^2 + 18n \leq 30n^2 - 3000$

$3n^2 - 18n - 3000 \geq 0$

$n^2 - 6n - 2000 \geq 0$

it's workable for $n \geq 0$

Therefore $c_1 = 54$, $c_2 = 60$, $n_0 = 10\sqrt{2}$

Hence $27n^2 + 18n = \Theta(0.5n^2 - 100)$

8. Write pseudocode for Strassen's algorithm and use Strassen's algorithm to compute the matrix product of *AB*. You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points)

$A = \begin{bmatrix} 2 & 1 \\ 3 & 2 \end{bmatrix}$, $B = \begin{bmatrix} 0 & 4 \\ 1 & 3 \end{bmatrix}$.

Strassen's Algorithm (A, B)

n = A.rows

let C be a new n×n matrix

if n == 1

$c_{11} = a_{11} \cdot b_{11}$

else

divide A into 4 sub-matrix

divide B into 4 sub-matrix

then

$S_1 = B_{12} - B_{22}$

$S_2 = A_{11} + A_{12}$

$S_3 = A_{21} + A_{22}$

$S_4 = B_{21} - B_{11}$

$S_5 = A_{11} + A_{22}$

$S_6 = B_{11} + B_{22}$

$S_7 = A_{12} - A_{22}$

$S_8 = B_{21} + B_{22}$

$S_9 = A_{11} - A_{21}$

$S_{10} = B_{11} + B_{12}$

then

$P_1 = A_{11} \cdot S_1$

$P_2 = S_2 \cdot B_{22}$

$P_3 = S_3 \cdot B_{11}$

$P_4 = A_{22} \cdot S_4$

$P_5 = S_5 \cdot S_6$

$P_6 = S_7 \cdot S_8$

$P_7 = S_9 \cdot S_{10}$

finally

$C_{11} = P_5 + P_4 - P_2 + P_6$

$C_{12} = P_1 + P_2$

$C_{21} = P_3 + P_4$

$C_{22} = P_5 + P_1 - P_3 - P_7$

return C

$A_{11} = 2$, $A_{12} = 1$, $A_{21} = 3$, $A_{22} = 2$

$B_{11} = 0$, $B_{12} = 4$, $B_{21} = 1$, $B_{22} = 3$

$S_1 = B_{12} - B_{22} = 1$

$S_2 = A_{11} + A_{12} = 3$

$S_3 = A_{21} + A_{22} = 5$

$S_4 = B_{21} - B_{11} = 1$

$S_5 = A_{11} + A_{22} = 4$

$S_6 = B_{11} + B_{22} = 3$

$S_7 = A_{12} - A_{22} = -1$

$S_8 = B_{21} + B_{22} = 4$

$S_9 = A_{11} - A_{21} = -1$

$S_{10} = B_{11} + B_{12} = 4$

$P_1 = A_{11} \cdot S_1 = 2$

$P_2 = S_2 \cdot B_{22} = 9$

$P_3 = S_3 \cdot B_{11} = 0$

$P_4 = A_{22} \cdot S_4 = 2$

$P_5 = S_5 \cdot S_6 = 12$

$P_6 = S_7 \cdot S_8 = -4$

$P_7 = S_9 \cdot S_{10} = -4$

then:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_5 + P_1 - P_3 - P_7 \end{bmatrix}$$

$$= \begin{bmatrix} 12 + 2 - 9 - 4 & 2 + 9 \\ 0 + 2 & 12 + 2 - 0 + 4 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 11 \\ 2 & 18 \end{bmatrix}$$

9. (Substitution method) Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$. You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points)

Guess: $T(n) = O(n^2)$

Induction: $T(m) \le cm^2$ for all $m < n$

$$T(n) = T(n-1) + n$$
$$\le c(n-1)^2 + n$$
$$\le c[n^2 - 2n + 1] + n$$
$$\le cn^2 - (2c-1)n + c$$

For $c \geq 1$, $2c - 1 \geq c$

Since $n \geq 1$, $(2c-1)n \geq c$, so $cn^2 - (2c-1)n + c \le cn^2$

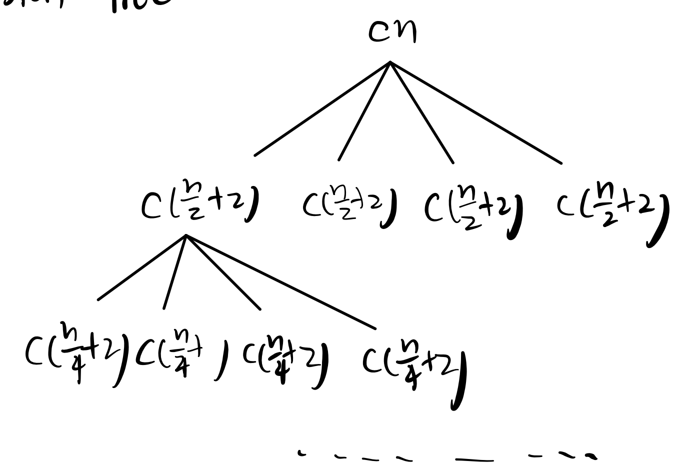
Therefore choose $c = 1$, $T(n) = O(n^2)$

10. Use a recursion tree to determine a good asymptotic upper bound on the following recurrence.

$$T(n) = 4T(n/2 + 2) + n$$

Use the substitute method to verify your answer. You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points)

Recursion Tree :



$$T_n = \sum_{i=0}^{\lg n} 4^i \left(\frac{n}{2^i} + 2\right) - 2$$

$$= \sum_{i=0}^{\lg n} 4^i \cdot \frac{n}{2^i} + \sum_{i=0}^{\lg n} 4^i \cdot 2 - 2$$

$$= n \sum_{i=0}^{\lg n} \frac{4^i}{2^i} + 2 \sum_{i=0}^{\lg n} 4^i - 2$$

$$= n \sum_{i=0}^{\lg n} 2^i + 2 \sum_{i=0}^{\lg n} 4^i - 2$$

$$= n \frac{2^{\lg n+1} - 1}{2-1} + 2 \frac{4^{\lg n+1} - 1}{4-1} - 2$$

$$= n(2^{\lg n+1} - 1) + \frac{2}{3}(4^{\lg n+1} - 1) - 2$$

$$= n(2 \cdot 2^{\lg n} - 1) + \frac{2}{3}(4 \cdot 4^{\lg n} - 1) - 2$$

$$= 2n^2 - n + \frac{8}{3}n^2 - \frac{2}{3} - 2$$

$$= \frac{14}{3} n^2 - n - \frac{8}{3}$$

$$= \Theta(n^2)$$

Assume $T(m) \le cm^2$, for $m < n$

$$T(n) = 4T(n/2+2)+n$$
$$\le 4c(n/2+2)^2+n$$
$$\le cn^2+(8c+1)n+16c$$

$\because c > 0$

$\therefore$ we can't prove the assumption

Assume $T(m) \le c(n^2-dn)$

$$T(n) = 4T(n/2+2)+n$$
$$\le 4c[(n/2+2)^2-d(n/2+2)]+n$$
$$= 4c(n^2/4+2n+4-dn/2-2d)+n$$
$$= c(n^2-dn)-(cd-8c-1)n-(d-2)8c$$

when $\begin{cases} cd-8c-1 \ge 0 \\ d-2 \ge 0 \end{cases}$

it's $\le c(n^2-dn)$

it means $\begin{cases} c \ge \frac{1}{d-8} \\ d \ge 2 \end{cases}$

$\because c \ge 0$

$\therefore d > 8$

Therefore, the upper bound is $T(n) = O(n^2-dn)$, where $d > 8$

11. Use the master method to give tight asymptotic bounds for the following recurrences.

1) $T(n) = 2T(n/4) + 1$
2) $T(n) = 2T(n/4) + \sqrt{n}$

You must show all the steps to arrive at the answers. Simply stating the answers will result in 0 points awarded. (10 points each, 20 points in total.)

(1) $T(n) = 2T(n/4) + 1$

$a = 2,\ b = 4,\ f(n) = 1$

$\log_b a = \log_4 2 = \frac{1}{2}$

$f(n) = O(n^{\log_4 2 - \varepsilon}),\ \varepsilon = \frac{1}{2}$

Hence, $T(n) = \Theta(n^{\frac{1}{2}})$

(2) $T(n) = 2T(n/4) + \sqrt{n}$

$a = 2,\ b = 4,\ f(n) = \sqrt{n} = n^{\frac{1}{2}}$

$\log_b a = \log_4 2 = \frac{1}{2}$

$n^{\log_b a} = f(n) = n^{\frac{1}{2}}$

$f(n) = \Theta(n^{\frac{1}{2}})$

Hence, $T(n) = n^{\frac{1}{2}} \lg n$