1、 For worst case $T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1) + (n-1) \end{cases}$

For general case $T(n) = \begin{cases} 1 \text{ or other constant} & \text{if } n=1 \\ T(n-1) + O(n) \end{cases}$

2 a、 (0,4) (1,4) (2,4) (3,4) (2,3)

b、 The array with descending order has the most inversions

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

c、 The running time is linear to the number of inversions.

Pseudo code of insertion sort as follows

```
Insertion sort ( arr[ ])   // Index of arr start with 1.
    n = arr.size
    for i = 2 to n
        for j = i to 2
            if arr[j] < arr[j-1]   // comparison
            then arr[j], arr[j-1] ← arr[j-1], arr[j]   // swap
```

we know every swap makes inversions reduced by 1.

The code will stop till inversions are 0, namely, arr is sorted.

So the number of swap is equal to inversions.

The number of comparison is $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$.

We can say when the size of arr is fixed, the running time is linear to the number of inversions.

d. countInversions (arr[ ], start, end) // Index of arr start with 1.

```
inversions = 0
If start < end
  then  mid = ⌊(start+end)/2⌋
        inversions += countInversions ( arr[ ] , start , mid )
        inversions += countInversions ( arr[ ] , mid+1 , end )
        inversions += countAndMerge ( arr[ ], start , mid , end )
return inversions
```

```
countAndMerge ( arr[ ], start, mid , end )
    l = mid - start + 1
    r = end - mid

    L[1,2 ... , l+1]
    R[1,2, ... , r+1]
    L[l+1] = ∞
    R[r+1] = ∞
    i = j = 1
    inversions = 0
    counted = false
    for k = start to end
        if counted == false and R[j] < L[i]
            inversions += ( l - i + 1 )
            counted = true

        if L[i] ≤ R[j]
            arr[k] = L[i]
            i++
        else A[k] = R[j]
            j++
            counted = false
    return inversions
```

The recurrence of worst case can be considered as

$T(n) = 2T(\lfloor \frac{n}{2} \rfloor) + cn$ for fixed positive integer c , by master theorem ,

we know $cn = \Theta(n^{\log_2 2}) \Rightarrow T(n) = \Theta(cn\log n)$

(i) $f(n)=\Theta(g(n)) \Rightarrow f(n)=O(g(n))$ and $f(n)=\Omega(g(n))$

3. From $f(n)=\Theta(g(n))$, we have

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for } n > n_0$$

$\because$ "$0 \le c_1 g(n) \le f(n)$ for $n > n_0$" meets the definition of $f(n)=\Omega(g(n))$

$\therefore f(n)=\Omega(g(n))$

$\because$ "$0 \le f(n) \le c_2 g(n)$ for $n > n_0$" meets the definition of $f(n)=O(g(n))$

$\therefore f(n)=O(g(n))$

(ii) $f(n)=O(g(n))$ and $f(n)=\Omega(g(n)) \Rightarrow f(n)=\Theta(g(n))$

From $f(n)=O(g(n))$, we have

$$0 \le f(n) \le c_3 g(n) \text{ for } n > n_1 \quad \cdots ①$$

From $f(n)=\Omega(g(n))$, we have

$$0 \le c_4 g(n) \le f(n) \text{ for } n > n_2 \quad \cdots ②$$

From ① ②, we have

$$0 \le c_4 g(n) \le f(n) \le c_3 g(n) \text{ for } n > \max(n_1, n_2)$$

this meets the definition of $f(n)=\Theta(g(n))$

$\therefore f(n)=\Theta(g(n))$

4. false $\quad n^2+n=\Theta(n^2) \ne \Theta(n) = \Theta(\min(n^2, n))$

5. true $\quad f(n)=O(g(n)) \Leftrightarrow \exists c, n_0 \ \forall n \ge n_0 \ 0 \le f(n) \le c g(n) \Rightarrow$

$0 \le \lg f(n) \le \lg c + \lg g(n) \le \lg c \times \lg g(n) + \lg g(n) = (\lg c + 1) g(n)$

since $\lg g(n) \ge 1$ for sufficiently large n.

This meets the definition of $\lg f(n) = O(\lg g(n))$

6. we guess $T(n) \le cn\lg n$

$$T(n) \le 2c\lfloor \tfrac{n}{2} \rfloor \lg \lfloor n/2 \rfloor + n$$
$$\le cn\lg \tfrac{n}{2} + n$$
$$= cn\lg n - cn\lg 2 + n$$
$$\le cn\lg n + n \quad \text{for } c \ge 1$$

we guess $T(n) \ge c(n+2)\lg(n+2)$

$$T(n) \ge 2c(\lfloor n/2 \rfloor + 2)(\lg \lfloor n/2 \rfloor + 2) + n$$
$$\ge 2c(n/2 - 1 + 2)(\lg \tfrac{n}{2} - 1 + 2) + n$$
$$= (cn + 2c)\lg \tfrac{n+2}{2} + n$$
$$= c(n+2)\lg(n+2) - c(n+2)\lg 2 + n$$
$$= c(n+2)\lg(n+2) + (1-c)n - 2c$$
$$\ge c(n+2)\lg(n+2) \quad \text{for } n \ge \tfrac{2c}{1-c},\ 0 \le c < 1$$

$$\therefore \begin{cases} T(n) = \Omega(n\lg n) \\ T(n) = O(n\lg n) \end{cases} \Rightarrow T(n) = \Theta(n\lg n)$$

7. $a = 1 \quad b = 2$

$f(n) = \Theta(n^{\log_2 1}) = \Theta(1)$

$T(n) = \Theta(\lg n)$. by master theorem

8. By master theorem

a. $T(n) = \Theta(n^4)$

b. $T(n) = \Theta(n)$

c. $T(n) = \Theta(n^2 \lg n)$

d. $T(n) = \Theta(n^2)$

e $T(n) = \Theta(n^{\log_2 7})$

f. $T(n) = \Theta(\sqrt{n} \lg n)$

g. Let $d = n \bmod 2$

$$T(n) = \sum_{j=1}^{j=\lfloor n/2 \rfloor} (2j+d)^2$$

$$= \sum_{j=1}^{\lfloor \frac{n}{2} \rfloor} 4j^2 + 4jd + d^2$$

$$= \Theta\left(\frac{n(n+2)(n+1)}{6} + \frac{n(n+2)d}{2} + \frac{d^2 n}{2}\right)$$

$$= \Theta(n^3)$$