2-1

a.

n個元素使用Insertion sort worst case = $\theta(n^2)$

現在每個 sublist 都有 k 個元素，總共有$\lceil n/k \rceil$個 sublist

所以時間$T(n) = \lceil n/k \rceil * \theta(k^2) = \theta(nk)$

b.

```
            n
           / \
        n/2   n/2
        / \   / \
       ...............
    k    k   ...   k   }⌈n/k⌉個
```

每一層做 merge 的時間是 $\theta(n)$

總共有$\lceil \lg(n/k) \rceil$層

(可以想成最底層有 n/k 群

往上 merge 成 1 個總共需要$\lceil \lg(n/k) \rceil$層)

$T(n) = \lceil \lg(n/k) \rceil * \theta(n)$

$\quad = \theta(n \lg(n/k))$

c.

merge sort $\rightarrow \theta(n \lg n)$

modified $\rightarrow \theta(nk + n(\lg(n/k)))$

if $\theta(n \lg n)$ bound $\theta(nk)$ then $k \le \theta(\lg n)$

if $\theta(n \lg n)$ bound $\theta(n(\lg(n/k)))$ then $k \ge \theta(1)$

所以 largest $k = \theta(\lg n)$

d.

若從 θ(理論上)來看，k 取 1 到 $\theta(\lg n)$之間的整數，時間都是 $\theta(n \lg n)$

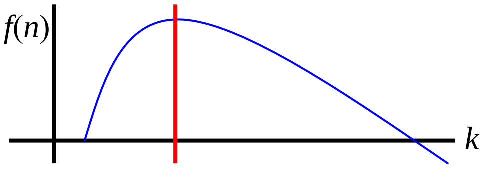
但從實際上來看的話，k 就和兩者前面的常數有關

假設insertion sort前面的常數是c_1，merge sort前面常數是c_2

modified後和原本的時間差 $= f(k) = c_2(n \lg n) - c_1(nk) - c_2(n \lg(n/k))$

$= c_2 n(\lg k) - c_1(nk)$，我們的目標是找個k讓這$f(k)$最大(省最多時間)

由於$n > 0$，$c_2 > c_1 > 0$，k爲正整數，圖大概長這形狀



理想的k是紅線的位置所在，我們可以由微分得到$k = c_2/((c_1)(\ln 2))$

由於 k 必須是整數，因此可以取最接近的兩個整數來比較看哪個好

2-4

a.

(1, 5), (2, 5), (3, 4), (3, 5), (4, 5)

b.

array < n, n-1, ..., 2, 1>

Total inversion = (n-1) + (n-2) + ... + 2 + 1 = $n(n-1) / 2$

c.

Insertion sort 每個數字 insert 後都至少往前比一次

所以總共至少比 $\theta(n)$次

而其中每多一個 inversion 就會讓 while 多執行一次

所以若總共有 k 個 inversion 就會多執行 k 次 while

Insertion sort running time → $T(n) = \theta(n + k)$

d.

直接 modify MERGE_SORT 中的 MERGE，當後面的數字往前搬的時候，同時計算他越過多少人，此數量即更正的 inversion 數量，最後全部加起來即是答案

Initially InversionNum := 0

```
MERGE (A, p, q, r)
    T1 ← p                                  //第一個sorted array起頭
    T2 ← q + 1                              //第二個sorted array起頭
    while (T1 ≤ q) or (T2 ≤ r) do
        If ((A[T1] > A[T2]) and (T2 ≤ r)) or (T1 > q)
            T3 ← A[T2]
            T2 ← T2 + 1
            InversionNum ← InversionNum + (q - T1 + 1)
        Else    T3 ← A[T1]
            T1 ← T1 + 1
        TempA[T1 - p + T2 - q - 1] ← T3     //放入暫存array
    for i ← 1 to (r - p + 1) do
        A[p + i - 1] ← TempA[i]             //複製回去
```

3-3

a.

$$2^{2^{n+1}} > 2^{2^n} > (n+1)! > n! > e^n > n \cdot 2^n > 2^n > (3/2)^n >$$
$$[n^{\lg\lg n} = \lg n^{\lg n}] > (\lg n)! > n^3 > [4^{\lg n} = n^2] > [n\lg n = \lg(n!)] >$$
$$[2^{\lg n} = n] > (\sqrt{2})^{\lg n} > 2^{\sqrt{2\lg n}} > \lg^2 n > \ln n > \sqrt{\lg n} > \ln\ln n >$$
$$2^{\lg^* n} > [\lg^* n = \lg^* \lg n] > \lg(\lg^* n) > [n^{1/\lg n} = 1]$$

$f(n) > g(n)$ 表示 $f(n) = \Omega(g(n))$

[ ]內的 $f(n) = g(n)$ 表示 $f(n) = \theta(g(n))$

很明顯的就不說明了，幾個比較不明顯的說明一下

$n^{\lg\lg n} = \lg n^{\lg n}$ 左右都取 lg 就會發現左右一模一樣

$\lg n^{\lg n} > (\lg n)! > n^3$ $n = 2^x$帶入，$x^x > x! > 8^x$很明顯

$n\lg n = \lg(n!)$ 因爲$(n/2)^{n/2} \le n! \le n^n$，因此可以左右夾擠$n\lg n$

$(\sqrt{2})^{\lg n} > 2^{\sqrt{2\lg n}} > \lg^2 n$ $n = 2^x$帶入就很明顯

$\lg^2 n > \ln n > \sqrt{\lg n} > \ln\ln n$ $\log_a b = (\lg b)/(\lg a) = \theta(\lg b)$，因此可當同底來比較

$\ln\ln n > 2^{\lg^* n} > \lg^* n$ 前兩項同時取 lg 或是靠感覺應該就可以知道大小
中間項取 lg 都還和第三項一樣大
所以這順序應該很明顯

$\lg^* n = \lg^* \lg n$ 注意 lg*的定義，前面那項只會比後面那項多 1

$\lg^* \lg n > \lg(\lg^* n)$ 如果覺得不好比
拿 lg*取代 lg*lg n 來和 lg(lg* n)比就很明顯
lg*之後再度取一次 lg 顯然就小很多

$n^{1/\lg n} = 1$ $n^{1/\lg n} = 2$，因爲 $1/\lg n = \log_n 2$

b.

符合的解答很多

其中一例 $f(n) = (2^{2^{n+2}})^{\sin n}$

when sin = 1 and $n \to \infty$

⇨ $f(n) = 2^{2^{n+2}} = \Omega(2^{2^{n+1}})$

when sin = -1 and $n \to \infty$

⇨ $f(n) = 1/2^{2^{n+2}} = O(1)$

$f(n)$有時候比上列所有$g_i(n)$都大，有時候又比所有$g_i(n)$都小

⇨ $f(n)$ is neither $O(g_i(n))$ nor $\Omega(g_i(n))$ for all $g_i(n)$ in part (a)