第 1 讲: 算法问题与解题的正确性

姓名： 林方浩　　**学号：**

评分： _____　　**评阅：** _____

截止时间：2024 年 3 月 3 日

请独立完成作业，不得抄袭。 若得到他人帮助, 请致谢。 若参考了其它资料，请给出引用。 鼓励讨论，但需独立书写解题过程。

1　作业（必做部分）

题目 1 (TC Problem 2 − 1)

解答：

a. Every k-length sublist have the running time $\Theta(k^2)$, in terms of Θ-notation. As a result, $\frac{n}{k}$ sublists have total running time of $\Theta(nk)$.

b. The recursion tree has $\lg(\frac{n}{k})$ levels, while the cost of each level equals to the total length of the array, which is n. So we can merge the sublists in $\Theta(n \lg(\frac{n}{k}))$ worst-case time.

c. $k = \Theta(\lg(n))$.

d. We can choose k as an integer which is the most closet to $\lg(n)$.

题目 2 (TC Problem 2 − 2)

解答：

a. We alse need to prove that : the set of elements in A' is the same as the set of elements in A.

b. The loop invariant is :$A[j]$ is the smallest element in the array$A[j..n]$.
Initialization **:** $j = A.length$, then $A[j..n]$ is just $A[n]$, it is obviously right.
Maintenance **:** If $A[j] < A[j-1]$,then after exchange them, the number indexed by $j-1$ is the smallest element in $A[j-1..n]$ since all elements in $A[j+1..n]$ is greater than $A[j]$. If $A[j] \geq A[j-1]$, then the conclusion does not change.
Termination **:** We have $j = i$, according to the maintenance part, $A[i]$ is the smallest element in the array $A[i..n]$.

c. The loop invariant is : The array $A[1..i]$ is sorted.
Initialization **:** We have $i = 1$, according to the termination in part ***b*** ,it is

correct.

Maintenance **:** Since we have already proved that $A[i+1]$ is the smallest element in the array $A[i+1..n]$, and we suppose that the array $A[1..i]$ is sorted, thus $A[1..i+1]$ is sorted.

Termination **:** This time, i changes to $A.length$,which implies that the whole array is sorted.

d. The worst-case running time of bubblesort is $\Theta(n^2)$, which is the same as the insertion sort.

题目 3 (TC Problem 2 − 3)

解答:

a. $\Theta(n)$.

b. $y = 0$

for $i = 0$ ***to*** n

$y = y + a_i x^i$

The running time of this algorithm is $\Theta(n^2)$,the efficiency of the algorithm is less than Horner's rule.

c. $y = x\left(\sum_{k=0}^{n-(i+1)} a_{k+i+1}x^k\right) + a_i = \left(\sum_{k=1}^{n-i} a_{k+i}x^k\right) + a_i = \sum_{k=0}^{n-i} a_{k+i}x^k$. When it is at termination, which means $i = 0$, we have $y = \sum_{k=0}^{n} a_k x^k$.

d. According to part ***c***, it is correct when it terminates.

题目 4 (TC Problem 2 − 4)

解答:

a. $pair(1,5)$ $pair(2,5)$ $pair(3,5)$ $pair(4,5)$ $pair(3,4)$

b. $< n, n-1, n-2, ..., 1 >$ has the most inversions. It has $\frac{n(n-1)}{2}$ inversions.

c. Linear. Every time we compare the key with the elements on the left of the key, the inversion's number decresed by 1. As a result, the number of inversions correspond to the times of comparision.

d. By using the algorithm similar to the merge sort, we can count the inversion numbers in every sublists, and keep track on the inversions when merging the two sublists, we sum the two parts ,that is the answer.

题目 5 (TC Problem 3 − 2)

解答:

A	B	O	o	Ω	ω	Θ
$\lg^k n$	n^ϵ	yes	yes	no	no	no
n^k	c^n	yes	yes	no	no	no
$\sqrt{n}$	$n^{sin(n)}$	no	no	no	no	no
2^n	$2^{\frac{n}{2}}$	no	no	yes	yes	no
$n^{\lg c}$	$c^{\lg n}$	yes	no	yes	no	yes
$\lg(n!)$	$\lg(n^n)$	yes	no	yes	no	yes

题目 6 (TC Problem 3 − 4)

解答:

a. False, we can choose $f(n) = n$, $g(n) = n^2$.

b. False, we can choose $f(n) = n$, $g(n) = n^2$.

c. If $f(n) = O(g(n))$, there exist a positive constants c and n_0 such that $1 \leq f(n) \leq cg(n)$ for all $n \geq n_0$, thus $\lg(f(n)) \leq \lg(cg(n))= \lg(c) + \lg(g(n))=\mathrm{O}(\lg(g(n)))$.

d. Just as the problem above, $0 \leq f(n) \leq cg(n)$, thus $2^{f(n)} \leq 2^{cg(n)} = 2^c \cdot 2^{g(n)} = O(2^{g(n)})$.

e. As we suppose that when n increase to infinity, $f(n)$ must be unbounded, which implies $f(n) \geq 1$, thus $f(n) \leq f(n)^2$.

f. There exist a positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$, so we can choose $\frac{1}{c}$ and the n_0 such that $\frac{1}{c}f(n) \leq g(n)$ for all $n \geq n_0$.

g. False, we can choose $f(n) = 2^n$.

h. We let $g(n) = o(f(n))$, thus for any positive constant $c > 0$, there exists a constant $n_0 > 0$ such that $0 \leq g(n) < cf(n)$ for all $n \geq n_0$, which implies $f(n) \leq f(n) + g(n) \leq (1 + c)f(n)$, thus $f(n) + o(f(n) = \Theta(f(n))$.
