论题 2-1 作业

姓名：陈劭源　　　　学号：161240004

1 [TC] Problem 2-1

a. For every sublist of length k, insertion sort can sort it in $\Theta(k^2)$ worst-case time, and there are n/k sublists, so these sublists can be sorted by insertion sort in $\Theta(k^2)n/k = \Theta(nk)$ worst-case time.

b. Apply the divide-and-conquer approach. Divide these sublists into two groups, each contains $n/(2k)$ sublists, and merge these sublists recursively, and finally merge the two groups. Let m denote the number of the sublists, i.e. n/k, and $T(m)$ denote the total running time of merging m sublists. The "divide", "conquer" and "combine" steps take a running time of $\Theta(1)$, $2T(m/2)$, $\Theta(km)$, so the recurrence is

$$T(m) = \begin{cases} \Theta(1) & m = 1 \\ 2T(m/2) + \Theta(km) & m > 1 \end{cases}.$$

Solve this recurrence, we obtain $T(m) = \Theta(km\log(m)) = \Theta(m\log(k))$.

c. A standard merge sort takes a running time of $\Theta(n\log n)$. When $k = \Theta(\log(n))$, $\Theta(nk + n\log(n/k)) = \Theta(n\log n)$. For every $k = \omega(\log(n))$, $\Theta(nk + n\log(n/k)) = \Theta(nk) = \omega(n\log n)$. Therefore, the largest value of k is $\Theta(\log n)$.

d. It mainly depends on the constant factors of merge sort and insertion sort. Let c_1 be the constant factor of merge sort, c_2 be the constant factor of insertion sort. We can rewrite the total running time as $T = c_1nk + c_2n\log(n/k)$. Minimize T with respect to k. Since $T'_k = nc_1 - nc_2/k$, $k = c_2/c_1$ is a minimum point of T. So we can choose $k = c_1/c_2$ in practice.