Exercise 2-1:

a. Answer:

For an array of size k, sorting it using insertion sort will cost $\Theta(k^2)$ time.

Thus for n / k such sublists, the total running time is $\Theta(n/k \cdot k^2) = \Theta(nk)$

b. Answer:

During the merging process, we pick the smallest element and put it in the merged list.

That's the "n" in $\Theta(n \lg(n/k))$

By using a minimal heap, we can find the smallest element in O(1) time.

But adjusting the heap requires $O(\lg(n/k))$ time. Thus the total running time will be $\Theta(n \lg(n/k))$ in the worst case.

c. Answer:

$Let\ n \lg n = nk + n \lg(n/k)$

$\therefore n \lg n = nk + n \lg n - n \lg k$

$\therefore n \lg k = nk$

$\therefore k = \lg k,\ no\ solution.$

$\because But\ there\ is\ actually\ a\ constant\ c \neq 1,\ s.t.\ cn \lg n = nk + n \lg(n/k)$

$\therefore There\ could\ be\ a\ k\ that\ make\ them\ equal,\ but\ it\ will\ be\ fairly\ small.$

d. Answer:

When k=1, T(n)=$\Theta(n \lg n)$, it's degenerated to heap sort, which is good e-nough.

When k=n, T(n)=$\Theta(n^2)$, it's degenerated to insertion sort, which is good e-nough for small data sets.

So, make it small. How about 5?