Data Structures - Solution no. 2

2. (a) If $f \prec h_1$ and $g \prec h_2$ then $f \cdot g \prec h_1 \cdot h_2$.
   TRUE: Follows since if $f(n) < c_1 h_1(n)$ for all $n > n_1$, and $g(n) < c_2 h_2(n)$ for all $n > n_2$, then $f(n)g(n) < c_1 c_2 h_1(n) h_2(n)$ for all $n > \max\{n_1, n_2\}$.

   (b) If $f = O(h_1)$ and $g = O(h_2)$ then $f^g = O(h_1^{h_2})$.
   FALSE: For example $n \prec n$ and $2 \prec 1$ but $n^2 \not\prec n^1$.

3. The complexity clearly follows the recursion

$$T(n) = 3T(2n/3) + C$$

   where n is the length of the sorted fragment and C is a constant. According to the first case in the Master Method, we have $T(n) = \Theta(n^k)$ where $k = \log_{3/2} 3$.

4. (b) Use divide and conquer. For n poeple to pay $T(n)$, each person should pay 1, and then they should part into two groups and repeat by recursion. The depth of the recursion is of order $\Theta(\log n)$, hence each person ends up paying $\Theta(\log n)$ times 1, which totals to $T(n) = \Theta(n \log n)$. Use induction to actually prove this.

   (c) Using the iteration method one sees that $T(n)$ equals n times the number of recursive itterations. Since each iteration halves the *logarithm* of the argument, the number of iterations is of order $\log\log n$. Hence $T(n) \sim n \log\log n$.

5. Use induction.