6.5-9

初始化一个空的最小堆H，取出$L_1 \dots L_k$中每个列表的最小元素，并放入H中，注意放入每个元素u的同时记录从下标为E_u的列表中取出。

接下来，每次从从H中执行`Extract-Min`，将取出的元素v放入结果列表的末尾，然后从L_{E_v}中，取出最小元素并加入H中；若L_{E_v}已经为空，则不需执行此操作。不断重复此过程，直到H为空。

由于每个元素被从列表中取出一次，并被加入、取出大小为k的堆各一次，因此时间复杂度为$\Theta(n)+\Theta(nlgk)=\Theta(nlgk)$

7-5

1. $p_i = \frac{1}{n}\frac{i-1}{n-1}\frac{n-2-(i-2)}{n-2} \times 3! = \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$
2. $\lim_{n\to\infty} p_{\lfloor \frac{n+1}{2} \rfloor} = \frac{3n}{2}$

   若随机选择一个元素，则$\lim_{n\to\infty} p_{\lfloor \frac{n+1}{2} \rfloor} = \frac{1}{n}$，

   因此此做法可以将选中排序后中位数的概率增大为原来的1.5倍。
3. $ans = \lim_{n\to\infty} \sum_{i=\frac{n}{3}}^{\frac{2n}{3}} p_i$

   $= \lim_{n\to\infty} \frac{6}{n(n-1)(n-2)} \sum_{i=\frac{n}{3}}^{\frac{2n}{3}} (i-1)(n-i)$

   $= \lim_{n\to\infty} \frac{6}{n(n-1)(n-2)} \sum_{i=\frac{n}{3}}^{\frac{2n}{3}} in - i^2 - n - i$

   利用积分，并近似处理，可得：

   $ans = \frac{6}{n^3} \int_{\frac{n}{3}}^{\frac{2n}{3}} -x^2 + (n-1)x - n\, dx$

   近似得$ans = \frac{13}{27} - \frac{2}{n}$，n趋于无穷时，落在区间内的概率为$\frac{13}{27}$，朴素算法落在该区间中的概率显然为$\frac{1}{3}$，改进的算法落在该区间内的概率约为朴素算法的1.44倍。

4. 快速排序算法的运行时间主要取决于递归的深度、每次分区的长度。由前几问的分析可以得知，采用三中值取`pivot`的算法，仅仅可以将“好的区间”出现的概率提高常数倍。然而，即使每次都可以将区间长度减半，算法的时间复杂度仍然为$\Theta(nlgn)$，所以采用三中值的方法不会比这个界更优，同时，也不会比朴素取中值的界更劣，因此此方法仅能影响算法的常数因子。

7.2-5

递归树上，叶子节点的区间长度为1。

设最浅的叶子节点的深度为h，最浅的叶子节点的祖先显然是每次都被分成了较短的区间，则有$n\,\alpha^h = 1$。

$\therefore h = log_\alpha \frac{1}{n} = \frac{lg\frac{1}{n}}{lg\alpha} = \frac{-lgn}{lg\alpha}$

设最深的叶子节点深度为H，最深的叶子节点的祖先显然是每次都被分成了较长的区间，则有$n\,(1-\alpha)^H = 1$。

$\therefore H = log_{1-\alpha} \frac{1}{n} = \frac{lg\frac{1}{n}}{lg(1-\alpha)} = \frac{-lgn}{lg(1-\alpha)}$