Homework 3 Solution

Professor Liu Yong
EL 9343 - Data Structure and Algorithm

October 22, 2016

Exercise 1. CLRS 7-3

a. $E[X_i]$ is simply $\sum_{j=1}^{n} Pr(selectj \wedge jisthei^{th}smallestelement)$. Since these two events are independent, so we can break the product apart and get $\sum_{j=1}^{n}$Pr { select j } $\cdot$ Pr {j is the i^{th} smallest element } $= \sum_{j=1}^{n} \frac{1}{n} \cdot \frac{1}{n}$, since each element is equally to be the i^{th} smallest element, we could get $E[X_i] = \frac{1}{n}$
b. The expected running time of quicksort depends on the sizes of the sub array after partitioning. We can sum over the size resulting from all possible pivots multiplied by the probability of that pivot. If the pivot is at q, then the resulting subarrays will have size q 1 and n q. In addition, it takes $\Theta(n)$ time to do the partitioning. So, the total expected running time is:$E[T(n)] = E[\sum_{q=n}^{n} X_q(T(q-1) + T(n-q) + \Theta(n))]$
c. By linearity of expectation, we can move the summation outside of the expectation. Because X_q is independent at the time it takes to do the recursive call, we can split the expectation of the production into the production of expectation.

$$\begin{aligned} E[T(n)] &= \sum_{q=1}^{n} E[X_q(T(q-1) + T(n-q) + \Theta(n))] \\ &= \sum_{q=1}^{n} E[x_q] \cdot E[T(q-1) + T(n-q) + \Theta(n)] \end{aligned}$$

Substitute the $\sum_{q=1}^{n} E[x_q]$ with $\frac{1}{n}$, we have

$$
\begin{aligned}
E[T(n)] &= \frac{1}{n}\sum_{q=1}^{n}(E[T(q-1)] + E[T(n-q) + \Theta(n)]) \\
&= \frac{1}{n}(\sum_{q=1}^{n} E[T(q-1)] + \sum_{q=1}^{n} E[T(n-1)] + \sum_{q=1}^{n} \Theta(n)) \\
&= \frac{1}{n}(E[T(0)] + E[T(1)] + ... + E[T(n-1)] \\
&+ E[T(n-1)] + E[T(n-2)] + ... + E[T(0)]) + n\Theta(n) \\
&= \frac{1}{n}(2\sum_{q=2}^{n-1} E[T(q)]) + \Theta(n)
\end{aligned}
$$

since we considered $E[T(0)]$ and $E(T(1))$ to be $\Theta(1)$

d.

$$
\begin{aligned}
\sum_{k=2}^{n-1} klgk &= \sum_{k=2}^{[n/2]-1} klgk + \sum_{k=[n/2]}^{n-1} klgk \\
&\leq \sum_{k=2}^{[n/2]-1} klg(n/2) + \sum_{k=n/2}^{n-1} klgk \\
&= lg(n/2)\sum_{k=2}^{[n/2]-1} k + lgn\sum_{k=n/2}^{n-1} k \\
&= lg(n/2)(\frac{(n/2-1)(n/2)}{2} - 1) + lg(n)(\frac{(n-1)(n)}{2} - \frac{(n/2-1)(n/2)}{2}) \\
&\leq (lgn - lg2)(\frac{(n/2)^2}{2}) + lg(n)(\frac{n^2}{2} - \frac{(n/2)^2}{2}) \\
&= (lgn)\frac{n^2}{8} - \frac{n^2}{8} + (lgn)\frac{n^2}{2} - (lgn)\frac{n^2}{8} \\
&= \frac{1}{2}n^2 lgn - \frac{1}{8}n^2
\end{aligned}
$$

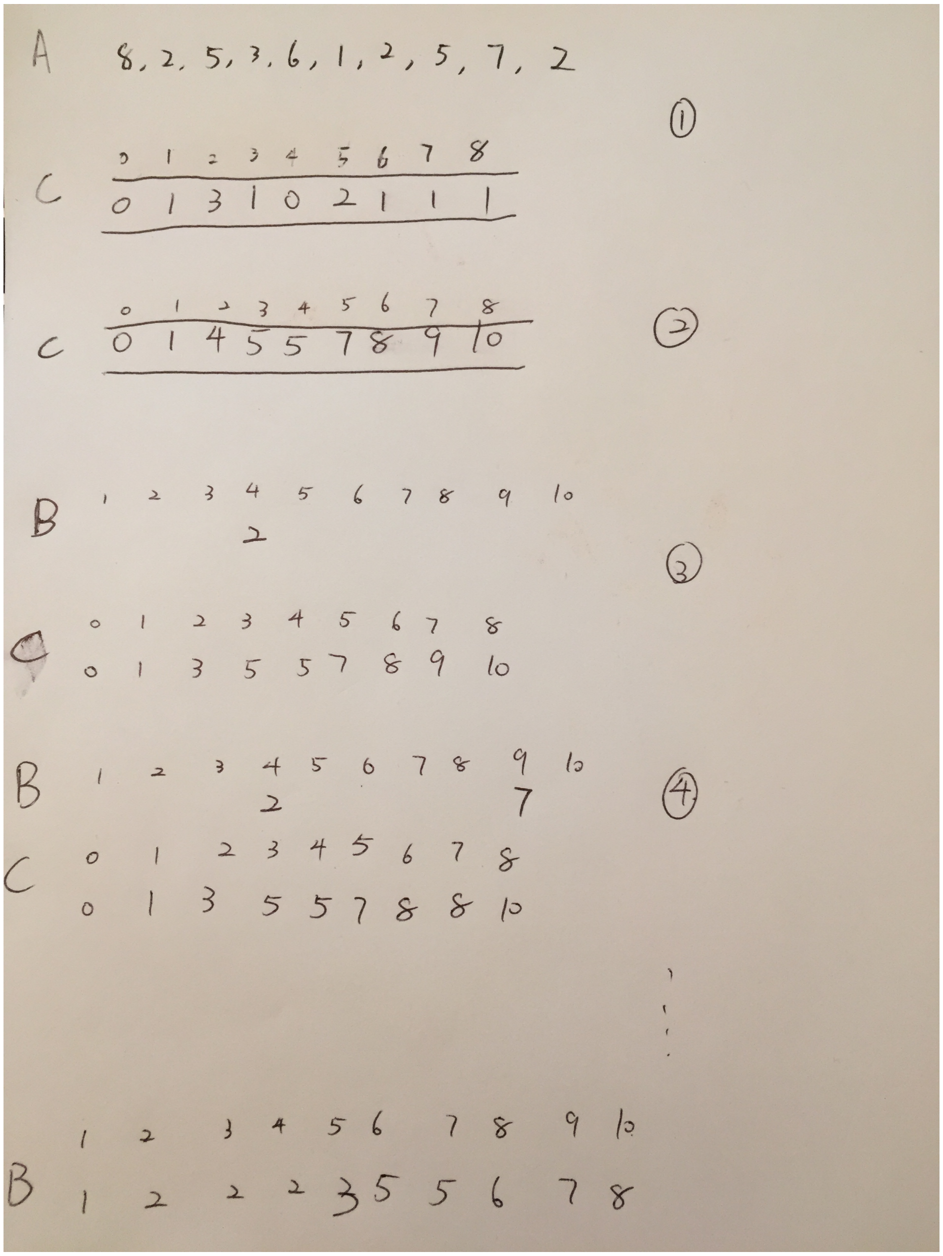
e.

$$\begin{aligned}
E[T(n)] &= \frac{2}{n}\sum_{q=2}^{n-1} E[T(q)] + kn \\
&\leq \frac{2}{n}\sum_{q=2}^{n-1} cqlgq + kn \\
&\leq \frac{2}{n}c(\frac{1}{2}n^2 lgn - \frac{1}{8}n^2) + kn \\
&= 2c(\frac{1}{2}nlgn - \frac{1}{8}n) + kn \\
&= cnlgn - \frac{1}{4}cn + kn \\
&\leq cnlgn
\end{aligned}$$

Exercise 2. CLRS 7-5

a.Lex $x_1, x_2 and x_3$ be the three values that are picked at random. Since there are $3! = 6$ ways to arrange different sequences of these three elements, we can calculate the probability assuming $x_1 < x_2 < x_3$ and multiply by 6: $p_i = 6PR[x_1 < i, x_2 = i, x_3 > i] = 6 \cdot \frac{2}{n} \cdot \frac{1}{n-1} \cdot \frac{n-i}{n-2} = \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$
b.We assume n is even, since it doesn't affect the ratio in the limit, $p_{n/2}\frac{6(\frac{n}{2}-1)(\frac{n}{2})}{n(n-1)(n-2)} = \frac{3(n^2-2n)}{2(n^3-3n^2+2n)}$. For the ordinary implementation, $p = \frac{1}{n}$. So $\lim_{n\to\infty} \frac{p_{\frac{n}{2}}}{p} = \lim \frac{3(1-2/n)}{2(1-3/n+2/n^2)} = \frac{3}{2}$
c.For the new method, the probability of a good split is $Pr[\frac{n}{3} \leq i \leq \frac{2n}{3}] = \sum_{i=n/3}^{2n/3} \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$. To approximate by an integral, I'll use the variable t to mean the fraction of the way from 1 to n. Then n, n-1, n-2 are nearly 1. So the approximating integral is $\int_{1/3}^{2/3} \frac{6t(1-t)}{1\cdot 1\cdot 1} = \int_{1/3}^{2/3} 6(t-t^2) = \frac{13}{27}$. For the old method, the probability of a good split is just $Pr[\frac{n}{3} \leq i \leq \frac{2n}{3}] = \sum_{i=n/3}^{2n/3} \frac{1}{n} = \frac{1}{n} \cdot \frac{n}{3} = \frac{1}{3} < \frac{13}{27}$
d. The best possible running time for any version of quicksort would be achieved when the median is picked for the pivot every time, but this is still on $\omega(nlgn)$ since the array is divided in half every time and the recursion handles both halves. Thus any scheme for picking the pivot, such as the median-of-three considered in this problem, cannot affect the asymptotic running time, but only the constant factor in the running time.

Exercise 3. Illustrate the operation of Count Sort.



Exercise 4. CLRS Exercise 8.2-4

Same as the counting sort, with the auxiliary array we get after step (b), we could get the number of items in range(x,y) = C[b]-C[a-1]

Exercise 5. CLRS Exercise 8.3-4

Firstly, we convert the number to n-based, since the number range in $[0, n^3-1]$, the converted number will be three digits long, so we can use Radix sort to sort them in $O(n)$ time.

Exercise 5. CLRS 9-1

a. Sort the numbers using the merge sort or heap sort, which take $\Theta(nlgn)$ worst-case time. Put the i largest elements with index based lookup operation and output the result, taking constant time, total worst-case time is $\Theta(nlgn)$

b. Implement the priority queue as a heap. Build the heap using the BUILD-HEAP, which takes $\Theta(n)$ time, then call HEAP-EXTRACT-MAX i times to get the i largest elements, in $\Theta(ilgn)$ worst-case time, and store them in reverse order to extraction in the output array. The worst-case extraction time is $\Theta(ilgn)$. Total worst-case running time: $\Theta(n + ilgn)$

c. Use the SELECT algorithm of Section 9.3 to find the ith largest number in $\Theta(n)$ time. Partition around that number in $\Theta(n)$ time. Sort the i largest numbers in $\Theta(ilgi)$ worst-case time. Total worst-case time: $\Theta(n + ilgi)$

Exercise 6. CLRS 11-2

a. There are (n choose k) sets of k keys that could map to that slot, each of them has probability (1/n) of actually being mapped to that slot, and each of the set of keys mapped to that slot is exactly S. Different choices of S give us disjoint events so we can add these probabilities over all possible choices of S, giving us the stated bound.

b. P_k = Pr[at least one slot contains exactly k keys and no slot contains more than k keys] $\leq$ Pr[at least one slot contains exactly k keys] $\leq nQ_k$. The final inequality is not an equality because the probabilities are not disjoint: it is possible for two or more different slots to contain exactly k keys.

c.Stirling;s approximation is:$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + O(\frac{1}{n}))$

we can use Stirling's approximation to expand (n choose k) = $\frac{n!}{k!(n-k)!}$

$\approx \sqrt{2\pi n}/(\sqrt{2\pi k}\sqrt{2\pi n - k})\frac{n^n}{(k^k(n-k)^{n-k})}(1+O(\frac{1}{n}))$ ¡ $\frac{n^n}{k^k(n-k)^(n-k)}$, here we've dropped the square roots and at the same time used them to cancel out the final $(1 + O(1))$ term. we're left only with $Q_k < \frac{n^n}{k^k(n-k)^(n-k)}\frac{1}{n}^k\ (1 - \frac{1}{n})^{n-k}$. Ignoring the $(1 - \frac{1}{n})^{n-k}$ part of this, splitting the n^n into $n^{n-k}n^k$ and regrouping. Finally, we need the inequality that, for any x and y ≥ 1, $(1 + \frac{x}{y})^{\frac{1}{y}} < e^y$ and substituting that in gives $Q_k < (\frac{e}{k})^k$ as desired.

d. We can rewrite our inequality from (c) as $lnQ_k < -klnk$. Plugging in $k_0 = clnn/lnlnn$, we get $lnQ_{k0} \leq -c(\frac{lnn}{lnlnn})ln(\frac{lnn}{lnlnn}) = -c(\frac{lnn}{lnlnn})(lnlnn - lnlnlnn)$. For sufficiently large n, $lnnlnn \geq 2lnlnlnn$, and this will simplify to $-\frac{c}{2}lnn$. Letting c = 6 gives $lnQ_{k0} < \frac{1}{n^3}$. From part (b), $P_{k0} \leq nQ_{k0} < \frac{1}{n^2}$ and the same holds for larger values of k.

e.

$$
\begin{aligned}
E[M] &= \sum_{i=1}^{n} iPr[M=i] \\
&= \sum_{i=1}^{clogn/loglogn} iPr[M=i]) + (\sum_{i=clogn/loglogn}^{n} iPr[M=i] \\
&\le (\sum_{i=1}^{clogn/loglogn} (clogn/loglogn)Pr[M=i]) + (\sum_{i=clogn/loglogn}^{n} nPr[M=i]) \\
&= (clogn/loglogn)(\sum_{i=1}^{clogn/loglogn} Pr[M=i]) + n(\sum_{clogn/loglogn}^{n} Pr[M=i]) \\
&\le (clogn/loglogn) + n(\sum_{clogn/loglogn}^{n} Pr[M=i]) \\
&\le (clogn/loglogn) + n(\sum_{clogn/loglogn}^{n} 1/n^3) \\
&\le (clogn/loglogn) + 1 \\
&= O(logn/loglogn)
\end{aligned}
$$