CMPUT 204 — Problem Set 3

(Partial solutions provided by Lisheng)

Topics covered in Part I are heapsort and priority queues; in Part Ⅱ are quicksort and lower bounds for comparison based sorting.

It is highly recommended that you read pages **135**–**168** very carefully and do **all** the exercises. The following are some of them that you are **REQUIRED** to practice on.

Quiz questions are mostly based on this list, with some minor modifications necessary. Consult your instructor and TAs if you have any problem with this list.

Part Ⅱ

1. Consider a Quick-sort of n distinct keys.
   (a) What does "equiprobable input distribution" mean?
   (b) Assuming an equiprobable input distribution, what is the probability that the initial splitter is the k'th largest key (among the n possible keys)? Justify your answer.

   Hints:

   (a) "Equiprobable input distribution" means every input instance has the same probability.
   (b) $\frac{1}{n}$
   Suppose the key set is $\{1, 2, \ldots, n\}$. Then there are $(n-1)!$ distinct permutations in which the last key is k. So, the probability is $\frac{(n-1)!}{n!} = \frac{1}{n}$.

2. Suppose Quick-sort is called on a sorted array with n distinct keys. How many
   (a) key comparisons
   (b) interchanges

   are performed?

   Hints:

   (a) $\sum_{k=1}^{n-1} k$
   (b) $\sum_{k=1}^{n-1}(k+1)$

3. Give a permutation of $\{1, 2, 3, \ldots, 14, 15\}$, such that for a Quick-sort of this input, every partition splits the list into two equal size sublists.

   Hints:

   There are a lot of such permutations. One of them is:

   $A[1..15] = (1, 3, 2, 6, 5, 7, 4, 12, 9, 11, 10, 14, 13, 15, 8)$.

4. P148, Ex 7.1-1.

5. P149, Ex 7.1-4.

   Hints:

   Modify `Partition`(A, p, r) such that $A[p..i]$ contains keys that are NOT less than the split key and $A[(i+2)..r]$ contains keys that are less than the split key.

6. P153, Ex 7.2-1.

   Hints:

   You can replace $\Theta(n)$ in the recurrence by some function $f(n)$ with $c_1 n \leq f(n) \leq c_2 n$. Using the iterated substitution method to guess a lower bound and an upper bound for $T(n)$, and prove by induction respectively.

7. P153, Ex 7.2-2.

   Hints:

   $\Theta(n^2)$ (trace `Partition`).

8. P153, Ex 7.2-3.

   Hints:

   You can derive the exact recurrence for the running time (key comparisons):

   $$T(n) = \begin{cases} 0, & \text{if } n = 0, 1 \\ (n-1) + T(0) + T(n-1), & \text{if } n \geq 2 \end{cases}$$

9. P153, Ex 7.2-4.

   Hints:

   Notice that the best case number of key comparisons for `Insertion sort` is $\Theta(n)$, where the

best case is the input array is sorted. Therefore, given that the input array is almost sorted, `Insertion sort` has a running time that is very close to the best case.

However, the worst case number of key comparisons for `Quicksort` is $\Theta(n^2)$, and a sorted array is one of the worst cases. Therefore, given that the input array is almost sorted, `Quicksort` has a running time that is very close to the worst case.

In summary, depending on how almost sorted the input array is, `Insertion sort` could beat `Quicksort`.

10. P159, Ex 7.4-2.

    Hints:

    Check lecture slides for the best case running time recurrence relation.

    You can prove the lower bound $\Omega(n \log n)$ by assuming $n = 2^k - 1$ for some k, using iterated substitution and proof by induction.

11. P159, Ex 7.4-5.

    Hints:

    When the top-level `Quicksort` call returns, the array is almost sorted in the sense that the array can be chopped into blocks (of length at most k) and these blocks are "sorted" — their relative order is the same as in the sorted array.

    In the average running time analysis for `Quicksort`, $T(n) = c$ when $n \leq k$. Therefore, the average running time for `Quicksort` is $T(n) \in \Theta(n \log(\frac{n}{k}))$. The average running time for `Insertion sort` on an array of length k is $\frac{k^2}{2}$ (why? on average every iteration needs to make $\frac{k}{2}$ comparisons). Therefore in total the running time for `Insertion sort` is $\frac{n}{k} \times \frac{k^2}{2} = \frac{nk}{2}$. Summing them up gives the average running time $\Theta(nk + n \log(\frac{n}{k}))$.

    Notice that when k increases, nk increases and $n \log(\frac{n}{k})$ decreases; when k decreases, nk decreases while $n \log(\frac{n}{k})$ increases. The idea on picking the right value(s) for k is to make a balance.

12. P161, Prob 7-3.

    Hints:

    (a)

    (b) $T(n) = 3T(\frac{2}{3}n)$

    (c) No, they don't deserve tenure.

13. P167, Ex 8.1-1.

    Hints:

    $(n - 1)$.

14. P167, Ex 8.1-3.

    Hints:

    (a) $\log(\frac{n!}{2}) \in \Theta(n \log n)$;

    (b) $\log(\frac{n!}{n}) \in \Theta(n \log n)$;

    (c) $\log(\frac{n!}{2^n}) \in \Theta(n \log n)$.

15. P167, Ex 8.1-4.

    Hints:

    Prove that there are $(k!)^{\frac{n}{k}}$ distinct permutations and use this number to derive the lower bound.