UNIVERSITY OF TORONTO, DEPARTMENT OF ECE

ECE1762 - Homework 2

Xinyun Lv(1001091178), Yang Wang(1001319227)

October 14, 2015

1 PROBLEM I

(a) **Solution:**

To get the expected number of recursive calls, we let x_k be the indicator random variable associated with the event in which the kth element is chosen as x^*. Thus,

$$x_k = \begin{cases} 1 & \text{if the } k\text{th smallest element is chosen as } x^* \\ 0 & \text{otherwise} \end{cases}$$

Let $T(n)$ denotes the expected number of recursive calls of a set of numbers with size of n. Thus,

$$T(n) = \begin{cases} T(0) & \text{if the largest element is chosen as } x^* \\ T(1) + T(0) & \text{if the second largest element is chosen as } x^* \\ T(2) + T(0) & \text{if the third largest element is chosen as } x^* \\ . \\ . \\ . \\ T(n-1) + T(0) & \text{if the smallest element is chosen as } x^* \end{cases}$$

Then we have,

$$T(n) = x_0 T(0) + \sum_{k=1}^{n-1} x_k (T(k) + T(0))$$

$$\begin{aligned}
E(T(n)) &= \frac{1}{n} \cdot T(0) + E(\sum_{k=1}^{n-1} x_k(T(k) + T(0))) \\
&= \frac{1}{n} \cdot T(0) + \sum_{k=1}^{n-1} \{[E(x_k T(k))] + E[x_k T(0)]\} \\
&= \frac{1}{n} \cdot T(0) + \frac{1}{n} \sum_{k=1}^{n-1} E[T(k)] + \frac{n-1}{n} \cdot T(0) \\
&= T(0) + \frac{1}{n} \sum_{k=1}^{n-1} E[T(k)]
\end{aligned}$$

Where $T(0) = 1$. So,

$$E(T(n)) = T(0) + \frac{1}{n} \sum_{k=1}^{n-1} E[T(k)]$$

(b) **Solution:**
According to the result we got from (a), we have:

$$\sum_{k=1}^{n-1} E[T(k)] = n \cdot (E(T(n)) - T(0))$$

Also, we have,

$$\sum_{k=1}^{n-2} E[T(k)] = (n-1) \cdot (E(T(n-1)) - T(0))$$

Substract two equations above we have:

$$\begin{aligned}
E(T(n-1)) &= n \cdot (E(T(n)) - T(0)) - (n-1) \cdot (E(T(n-1) - T(0)) \\
nE(T(n)) - nE(T(n-1)) &= T(0) = 1 \\
E(T(n)) &= E(T(n-1)) + \frac{1}{n}
\end{aligned}$$

Thus, we have,

$$\begin{aligned}
E(T(n)) &= E(T(n-1)) + \frac{1}{n} \\
E(T(n-1)) &= E(T(n-2)) + \frac{1}{n-1} \\
\ldots &= \ldots \\
E(T(2)) &= E(T(1)) + \frac{1}{2}
\end{aligned}$$

Which could be simplified as,

$$E(T(n)) = E(T(1)) + \sum_{i=2}^{n} \frac{1}{i}$$

Since $E(T(1)) = 1$,

$$E(T(n)) = \sum_{i=1}^{n} \frac{1}{i}$$

(c) **Solution:**
Similar with (a), let $T(n)$ denotes the expected running time of a set of numbers with size of n. Thus,

$$T(n) = \begin{cases} \theta(n) & \text{if the largest element is chosen as } x^* \\ T(1)+\theta(n) & \text{if the second largest element is chosen as } x^* \\ T(2)+\theta(n) & \text{if the third largest element is chosen as } x^* \\ \cdot & \\ \cdot & \\ \cdot & \\ T(n-1)+\theta(n) & \text{if the smallest element is chosen as } x^* \end{cases}$$

Then we have,

$$T(n) = x_0\theta(n) + \sum_{k=1}^{n-1} x_k(T(k)+\theta(n))$$

$$\begin{aligned} E(T(n)) &= \frac{1}{n}\cdot\theta(n) + E(\sum_{k=1}^{n-1} x_k(T(k)+\theta(n))) \\ &= \frac{1}{n}\cdot\theta(n) + \sum_{k=1}^{n-1}\{[E(x_k T(k))] + E[x_k\theta(n)]\} \\ &= \frac{1}{n}\cdot\theta(n) + \frac{1}{n}\sum_{k=1}^{n-1} E[T(k)] + \frac{n-1}{n}\cdot\theta(n) \\ &= \theta(n) + \frac{1}{n}\sum_{k=1}^{n-1} E[T(k)] \end{aligned}$$

So,

$$E(T(n)) = \theta(n) + \frac{1}{n}\sum_{k=1}^{n-1} E[T(k)]$$

(d) **Solution:**
Here, we can make a guess, $E(T(n)) \le cn = O(n)$, where c is a constant number. By induction we have,

$$\begin{aligned} E(T(n)) &\le \theta(n) + \frac{1}{n}\sum_{k=1}^{n-1} ck \\ &\le \theta(n) + \frac{c}{n}\sum_{k=1}^{n-1} k \\ &\le \theta(n) + \frac{cn}{2} = O(n) \end{aligned}$$

(e) **Solution:**
To compute the probability that Find-Max(X) compares the i-th and the j-th largest

items in X, we can think about when two items are not compared. For ease of analysis, we rename the elements of X as $z_n, z_{n-1}, ..., z_1$, with z_i being the ith largest element. We can also define the set $Z_{ij} = \{z_j, z_{j-1}, ..., z_i\}$ to be the set of elements between z_i and z_j, inclusive. because we assume that element values are distinct, once a x^* is chosen with $z_j < x < z_1$, we know that z_i and z_j cannot be compared at any subsequent time. If, on the other hand, z_i is chosen as x^* before any other item in Z_{1j}, then z_i will be compared to each item in Z_{1j} including z_j, except for itself. Similarly, if z_j is chosen as x^* before any other item in Z_{1j}, then z_j will be compared to each item in Z_{1j} including z_i, except for itself. Thus z_i and z_j are compared if and only if the first element to be chosen as x^* from Z_{1j} is either z_i or z_j. Meanwhile, prior to the point at which an element from Z_{1j} has been chosen as x^*, the whole Z_{1j} is together in the same partition, so any element of Z_{1j} is equally likely to be the first one chosen as x^*. Because the set Z_{1j} has j elements, and because x^* is chosen randomly and independently, the probability that any gievn element is the first one chosen as x^* is $1/j$. Thus, we have

$$Pr(z_i \text{ compared with } z_j) = \frac{2}{j}$$

2 Problem II

Solution:

To show that number of comparisons is at least $n - k + log\binom{n}{k-1}$ is equivalent to showing that number of outcome boxes is at least $2^{n-k} \times \binom{n}{k-1}$.

Let $V(n,k)$ denote the number of comparisons of finding the kth largest element of n-element set.

When $k = 1$, $V(n,1)$ represent the number of comparisons for finding the largest element of n-element set. We could easily observe that $V(n,1) \geq n-1$, since every element except the largest must lose at lease one comparison. This observation implies that in any comparison tree to find the largest element, every leaf has depth at least $n-1$, which implies that there must be at least 2^{n-1} leaves. We can generalize this argument to prove a lower bound for $V(n,k)$ for arbitrary values of k.

Let T be a comparison tree that identifies the kth largest element $x_{(k)} \in X$.

Suppose we are at some outcome box for determing the kth largest element $x_{(k)}$. Assume there are some elements not yet directly known to be bigger or smaller than $x_{(k)}$.

- Set U is set of elements not yet directly known to be bigger or smaller than $x_{(k)}$.
- Set L is set of those known to be larger than $x_{(k)}$
- Set S is set of those known to be smaller than $x_{(k)}$

Suppose that there is an element $x_* \in U$ such that $x_* > x_{(k)}$, which implies that $\mid L \mid = k-1+1 = k$ and $x_{(k)}$ become the $k+1$ largest element. However, we have known that $x_{(k)}$ is the kth largest element which is a contradiction. So we can conclude that when a decision reaching the right outcome of finding the kth largest element also decide correctly which the $k-1$ larger elements are.

Now suppose that the set of $k-1$ largest elements of X:

$$L = \{x_{(1)}, x_{(2)}, ..., x_{(k-1)}\}$$

Since those elements in set L must bigger than those elements in set S, we could remove those comparisons from T. Call the reduced tree R. Since the reduced tree R also identifies the largest element of S, based on the conclusion we got from base case $k = 1$, R must have at least 2^{n-k} leaves. Since there are $\binom{n}{k-1}$ choices for set L, we conclude that T has at least $\binom{n}{k-1} \cdot 2^{n-k}$ leaves which also implies that the number of comparisons is at least $n - k + log\binom{n}{k-1}$.

3 Problem III

Solution:

Since finding the kth largest element in the union of the two arrays is equivalent of finding the $n-k+1$th smallest element. To easier analyze the algorithm we are going to apply, we denote $k^* = n-k+1$.

Binary search is a good example of achieving logarithmic complexity by halving its search space in each iteration. As a good hint, to achieve complexity of $O(logn)$ running time, we must halved the search space of A and B in each iteration.

Base case is pretty strait forward: If length of one of the array is 0, then the answer is $k*$th smallest element of the second array.

Array A and array B could be described as follows:
where $mida$ is the mid index of the first array A and $midb$ is the mid index of the second array B.

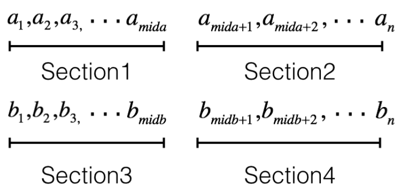
$a_1, a_2, a_3, \cdots a_{mida}$
$a_{mida+1}, a_{mida+2}, \cdots a_n$
Section1
Section2
$b_1, b_2, b_3, \cdots b_{midb}$
$b_{midb+1}, b_{midb+2}, \cdots b_n$
Section3
Section4

Figure 3.1: Picture of putting balls into bins

If $mida + midb < k^*$, then it means that the k^* smallest element is at {Section1, Section2, Section4} or {Section2, Section3, Section4}, the reason is as follows. If $A[mida] > B[midb]$, we can safely discard section3 since every element of section3 is smaller than $A[mida]$ and cannot be the k^*th smallest element of the union array. Similarly, If $A[mida] < B[midb]$ then we can discard section1.

However, if $mida + midb > k^*$, then it means that the k^* smallest element is at {Section1, Section2, Section3} or {Section1, Section3, Section4}, the reason is as follows. If $A[mida] < B[midb]$, we can safely discard section4 since every element of section4 is bigger than $A[mida]$ and cannot be the k^*th smallest element of the union array. Similarly, If $A[mida] > B[midb]$ then we can discard section2.

The pseudo code is as follows:

Algorithm 1 Find(A, B, k^*), $k^* = n - k + 1$

```
if length(A) = 0 then return $B[k^*]$
if length(B) = 0 then return $A[k^*]$
$mida \leftarrow length(A)/2$
$midb \leftarrow length(B)/2$
if $mida + midb < k^*$ then
    if $A[mida] > B[midb]$ then $Find(A, B[midb+1:], k^* - midb - 1)$
    else$Find(A[mida+1:], B, k^* - mida - 1)$
else
    if $A[mida] > B[midb]$ then $Find(A[:mida], B, k^*)$
    else$Find(A, B[:midb], k^*)$
```

Since we halved the search space of A and B in each iteration, so the time complexity is $O(logn + logn) = O(logn)$

4 Problem IV

Solution: