CS217 - Algorithm Design and Analysis
Homework Assignment 2

Group Name: Static
Group Members: Zhenjia Xu, JiaCheng Yang
Zhuolin Yang, Wenda Qiu

22nd Oct, 2016

2 Sorting Algorithms

Exercise 2.1.

- First, we divide the elements of A into $n/2$ pairs.
- Second, we make a comparison in each pair. Put the bigger one into array B, and put the smaller one into array C (If the two elements are equal, put one of them into B, and the other into C).

  It is obvious that the maximum is in B, and the minimum is in C.

  In this step, we get two arrays of size $n/2-1$ by $n/2$ comparisons.
- Third, find the maximum in B ($n/2-1$ comparisons) and the mimimum in C ($n/2$ comparisons).

  these two elements are the maximum and minimum in A.

In all, the total number of comparisons is $N = n/2+n/2-1+n/2-1 = \frac{3}{2}n-2$

Exercise 2.2.

- Step1, Let $B_0 = A$ (B_0 is of size n).
- Step2, divide B_0 into $n/2$ pairs, make a comparison in each pair and put the bigger one into B_1 (B_1 is of size $n/2$).
- Step3, do the same thing in Step2 to generate B_2 by B_1, generateB_3 by $B_2 \cdots$ until we get $B_{\log_2 n}$(containing only one element, which is the maximum).

The above three steps use $n-1$ comparisons in all.
Because we need $\frac{n}{2^i}$ comparions to generate $B_i (1 \leq i \leq \log_2 n)$

$$\sum_{i=1}^{\log_2 n} \frac{n}{2^i} = n - 1$$

The maximum compares with $\log_2 n$ elements in above steps, and the second largest element is obviously in these $\log_2 n$ elements. So we only need another $\log_2 n - 1$ comparisons to find the second largest element.

In all, the total number of comparisons is $N = n-1+\log_2 n-1 = n+\log_2 n-2$

Exercise 2.3. Consider the maximum is based on the **larger than**, a relation between two numbers. As you can see, this relation is transitive, and every comparison of two numbers can build a relationship. To make a number **maximum**, you need to build a directed tree-like structure which its root is the maximum number. In order to build such a structure, you need at least $n-1$ edges. thus, you need to compare at least $n-1$ time to construct $n-1$ relations.

Exercise 2.4. Consider constructing this special sequence by Recursive Process. Let $n = 2^d$, We define $F(d)$ is a process which can give us a acceptable sequence whose length is 2^d. when d equals to zero seems trivial, just return the only one element. when $d > 0$, consider making the number of comparisons as much as possible, after sorting this sequence, we split the sequences in two parts. choose the index of 1, 3, 5, .. $2^d - 1$ into one part, while the index of 2, 4, 6 .. 2^d into another part. then put these two parts into $F(d-1)$ individually. After that, merge these two new parts(one part is first, then another part), thus we get the sequence which $F(d)$ need.

Now, we prove the total comparison of sequence constructed by $F(d)$ is $n \log n - n + 1$ by induction.

Proof. the total comparison of sequence which $F(d)$ provides is $n \log n - n + 1$.

1. Basic step
   When $d = 0, n = 1$, it's clearly right because:

   $$n \log n - n + 1 = 0$$

   while the only element don't need to compare.

2. Induction step
   Assume that the comparisons fit when $F(d-1)$ works, consider $F(d)$.

   Consider the process of merge sort, first it split the sequences into two parts, which are the first half of the sequence and the rest. the two parts are all constructed by $F(d-1)$, so while doing this, it costs $2 \times (n \log n - n + 1) = 2 \times (2^{d-1} \times (d-1) - 2^{d-1} + 1)$ comparisons, where $n = 2^{d-1}$.

   After that, we can easily find that merge these two parts need $2^d - 1$ comparisons, because the first parts contains the $1st, 3rd, 5th..$ large number of the whole sequences, when the other part contains the $2nd, 4th, 6th..$, so the total comparisons is:

   $$2 \times (2^{d-1} \times (d-1) - 2^{d-1} + 1) + 2^d - 1 = 2^d \times d - 2^d + 1$$

3. Conclusion
   By induction, we have proved that the total comparison of sequence which $F(d)$ provides is $n \log n - n + 1$, where $n = 2^d$.

□

Exercise 2.5. By the linearity of expectaton, the expected number of comparisons of quicksort on a random input is:

$$\begin{aligned}
\mathbb{E}[comparisons] &= \sum_{i \neq j} \frac{1}{|i-j|+1} && (1)\\
&= 2 * \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} && (2)\\
&= 2 * \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} && (3)\\
&= 2 * \sum_{k=2}^{n} \frac{n-k+1}{k} && (4)\\
&= 2 * [(n+1) * \sum_{k=2}^{n} \frac{1}{k} - (n-1)] && (5)\\
&= 2 * [(n+1) * (H_n - 1) - (n-1)] && (6)\\
&= 2nH_n + 2H_n - 4n && (7)
\end{aligned}$$

Exercise 2.6. The main difference between QUICKSELECT and QUICKSORT is that the QUICKSELECT only deal with one of the sequences parted by the pivot while the QUICKSORT process both.

Here is an figure about an example of choosing the 7-th element in the sequence $[4, 2, 5, 6, 8, 1, 7, 3]$(The nodes colored red are considered to be visited).

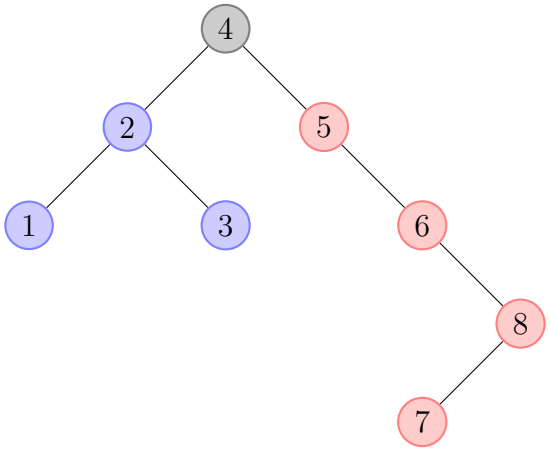

Figure 1: The example figure

Exercise 2.7. Let's define $[i, j]$ as a set which contains the integral numbers between i and j. Recalling the proof that $A_{i,j} = 1$ iff the number in $[i, j]$ in the input sequence that occurs first must be i, we can obvirusly make an inferrence that $B_{i,j,k} = 1$ iff the number in $[i, j] \cup [i, k]$ in the input sequence that occurs first must be i.

Notice that every number has no difference among other numbers except their pronunciations. In other words, we can choose the number which occurs first with equal probability.

Hence

$$\mathbb{E}[B_{i,j,k}] = \frac{1}{|[i,j] \cup [i,k]|}$$

Lemma 1. $|[i,j] \cup [i,k]| = \max\{i,j,k\} - \min\{i,j,k\} + 1$

Proof. We will prove the lemma with three cases below.

- If $i = \min\{i,j,k\}$, then

$$|[i,j] \cup [i,k]| = \max\{j,k\} - i + 1 = \max\{i,j,k\} - \min\{i,j,k\} + 1$$

- If $i = \max\{i,j,k\}$, then

$$|[i,j] \cup [i,k]| = i - \min\{j,k\} + 1 = \max\{i,j,k\} - \min\{i,j,k\} + 1$$

- If $j < i < k$ or $k < i < j$, then

$$|[i,j] \cup [i,k]| = \max\{j,k\} - \min\{j,k\} + 1 = \max\{i,j,k\} - \min\{i,j,k\} + 1$$

□

According to the lemma, the following formula holds.

$$\mathbb{E}[B_{i,j,k}] = \frac{1}{\max\{i,j,k\} - \min\{i,j,k\} + 1}$$

Exercise 2.8. Since A is a permutaion of size n, the answer for $QUICKSELECT(\pi = \{A,k\})$ is k. Similar like quicksort tree, quickselect tree is a quicksort tree without the nodes which haven't node k in its subtree.

So, elements i(i is pivot) and j($j \neq i$) will have a comparasion if and only if in quicksort tree, i is a common ancestor of k and j.

Thus, the total number of comparasions is

$$C(\pi) = \sum_{i \neq j} B_{i,j,k}$$

Exercise 2.9. We need to calculate the fomula in Ex8 for a random A and a specific k. Since $E[B_{i,j,k}] = E[B_{j,i,k}]$, we can assume $i < j$ without loss of generality.

- When $k \leq i$ holds:

$$\begin{aligned}
\sum_{k \leq i < j} E[B_{i,j,k}] &= \sum_{k<j} \frac{\binom{1}{j-k}}{j-k+1} \\
&= \sum_{\Delta=1}^{n-k} \frac{\Delta}{\Delta+1} \\
&= \sum_{\Delta=1}^{n-k} (1 - \frac{1}{\Delta+1}) \\
&= n - k - H_{n-k+1} + 1
\end{aligned}$$

- When $j \leq k$ holds:

$$\begin{aligned}\sum_{i<j\leq k} E[B_{i,j,k}] &= \sum_{i<k} \frac{\binom{1}{k-i}}{k-i+1} \\ &= \sum_{\Delta=1}^{k-1} \frac{\Delta}{\Delta+1} \\ &= \sum_{\Delta=1}^{k-1} (1 - \frac{1}{\Delta+1}) \\ &= k - H_k\end{aligned}$$

- When $i < k$ and $k < j$ holds:

$$\begin{aligned}\sum_{i<k<j} E[B_{i,j,k}] &= \sum_{i=1}^{k-1} \sum_{j=k+1}^{n} \frac{1}{j-i+1} \\ &= \sum_{i=1}^{k-1} \sum_{t=k-i+2}^{n-i+1} \frac{1}{t} \\ &= \sum_{i=1}^{k-1} (\sum_{t=3}^{n} \frac{1}{t} - \sum_{t=3}^{k-i+1} \frac{1}{t} - \sum_{t=n-i+2}^{n} \frac{1}{t}) \\ &= \sum_{i=1}^{k-1} (H_n - H_2) - \sum_{i=1}^{k-1} \sum_{t=3}^{k-i+1} \frac{1}{t} - \sum_{i=1}^{k-1} \sum_{t=n-i+2}^{n} \frac{1}{t} \\ &= (k-1)(H_n - H_2) - \sum_{t=3}^{k} \sum_{i=1}^{k-t+1} \frac{1}{t} - \sum_{t=n-k+3}^{n} \sum_{i=n-t+2}^{k-1} \frac{1}{t} \\ &= (k-1)(H_n - H_2) - \sum_{t=3}^{k} \frac{k-t+1}{t} - \sum_{t=n-k+3}^{n} \frac{t+k-n-2}{t} \\ &= (k-1)(H_n - H_2) - \sum_{t=3}^{k} (\frac{k+1}{t} - 1) - \sum_{t=n-k+3}^{n} (\frac{k-n-2}{t} + 1) \\ &= (k-1)(H_n - H_2) - (k+1) \sum_{t=3}^{k} \frac{1}{t} + (k-2) - (k-n-2) \sum_{t=n-k+3}^{n} \frac{1}{t} - (k-2) \\ &= (k-1)(H_n - H_2) - (k+1)(H_k - H_2) - (k-n-2)(H_n - H_{n-k+2}) \\ &= (n+1)H_n - (k+1)H_k - (n-k+2)H_{n-k+2} + 3\end{aligned}$$

Then we have:

$$\begin{aligned}E[C] &= 2(n - k - H_{n-k+1} + 1 + k - H_k + (n+1)H_n - (k+1)H_k - (n-k+2)H_{n-k+2} + 3) \\ &= n - (n-k+3)H_{n-k+1} + (n+1)H_n - (k+2)H_k + 3\end{aligned}$$