CS217 - Algorithm Design and Analysis
Homework Assignment 2

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

24th Oct, 2016

2 Sorting Algorithms

Exercise 2.1.

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

- Second, find the maximum in B ($n/2-1$ comparisons) and the mimimum in C ($n/2-1$ comparisons).

  these two elements are the maximum and minimum in A.

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

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 similar 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 $\dfrac{n}{2^i}$ comparions to generate $B_i (1 \le i \le \log_2 n)$

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

The maximum element 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:

$$\mathbb{E}[comparisons] = \sum_{i \neq j} \frac{1}{|i-j|+1} \tag{1}$$

$$= 2 \times \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{j-i+1} \tag{2}$$

$$= 2 \times \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{1}{k} \tag{3}$$

$$= 2 \times \sum_{k=2}^{n} \frac{n-k+1}{k} \tag{4}$$

$$= 2 \times [(n+1) \times \sum_{k=2}^{n} \frac{1}{k} - (n-1)] \tag{5}$$

$$= 2 \times [(n+1) \times (H_n - 1) - (n-1)] \tag{6}$$

$$= 2nH_n + 2H_n - 4n \tag{7}$$

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 blue are considered to be visited and the node colored gray is the root).

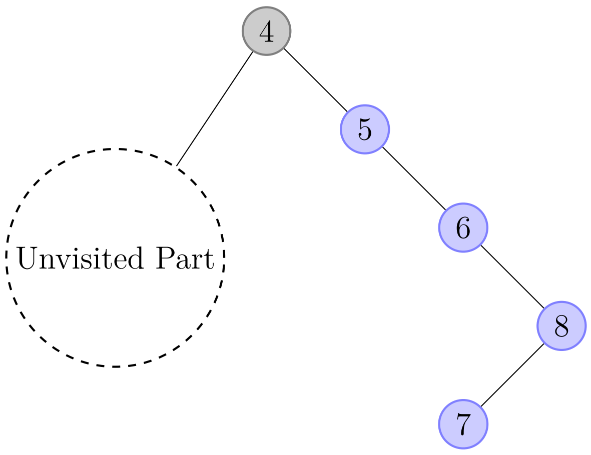

Figure 1: The example figure

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. Considering we get an exact sequence and wanting to select the k-th element, we want to calculate the exact number of comparisons. An obvirus notice is that the number of comparisons equals to the total number of comparisons of every node. Another notice is that every node is only compared by its **vaild** ancestor.
Here x is a **vaild** ancestor of y iff:

- x is an ancestor of y;
- x is visited in the procedure QUICKSELECT.

Hence, the number of comparisons of node i is just the number of its **vaild** ancestors. And j is a **vaild** ancestor of i iff j is the common ancestor of i and k. Hence, the comparsions of node i is

$$\sum_{j=1}^{n} B_{j,i,k}$$

And the total number of comparisons is that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} B_{j,i,k}$$

However, in the implementation code in the problem, if A has only one element, we don't have to make any comparsion. Hence, if k is a leaf, we should fix our answer and actually if k is a leaf, we should divide the comparisons by one. And thus the expected comparisons is that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} B_{j,i,k} - \mathbb{E}[k \text{ is a leaf}]$$

Exercise 2.9. We need to calculate the fomula in Ex8 for a random A and a specific k.

$$\begin{aligned}
\mathbb{E}[C] &= \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbb{E}[B_{i,j,k}] - \mathbb{E}[k \text{ is a leaf}] \\
&= \sum_{i \neq j} \mathbb{E}[B_{i,j,k}] + \sum_{i=1}^{n} \mathbb{E}[B_{i,i,k}] - \mathbb{E}[k \text{ is a leaf}]
\end{aligned}$$

$$= 2\sum_{1\le i<j\le n}\mathbb{E}[B_{i,j,k}] + \sum_{i=1}^{n}\mathbb{E}[A_{i,k}] - \mathbb{E}[k \text{ is a leaf}]$$

Firstly, we calculate $\sum_{1\le i<j\le n}\mathbb{E}[B_{i,j,k}]$

- When $k \le i$ holds:

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

- When $j \le k$ holds:

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

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

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

$$
\begin{aligned}
&= (k-1)(H_n - H_2) - \sum_{t=3}^{k}(\frac{k+1}{t} - 1) - \sum_{t=n-k+3}^{n}(\frac{k-n-2}{t} + 1) \\
&= (k-1)(H_n - H_2) - (k+1)\sum_{t=3}^{k}\frac{1}{t} + (k-2) \\
&\quad - (k-n-2)\sum_{t=n-k+3}^{n}\frac{1}{t} - (k-2) \\
&= (k-1)(H_n - H_2) - (k+1)(H_k - H_2) - (k-n-2)(H_n - H_{n-k+2}) \\
&= (n+1)H_n - (k+1)H_k - (n-k+2)H_{n-k+2} + 3
\end{aligned}
$$

Then we have:

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

Secondly, we calculate $\sum_{i=1}^{n}\mathbb{E}[A_{i,k}]$

$$
\begin{aligned}
\sum_{i=1}^{n}\mathbb{E}[A_{i,k}] &= \sum_{i=1}^{n}\frac{1}{|i-k|+1} \\
&= \sum_{i=1}^{k}\frac{1}{k-i+1} + \sum_{i=k+1}^{n}\frac{1}{i-k+1} \\
&= \sum_{t=1}^{k}\frac{1}{t} + \sum_{t=2}^{n-k+1}\frac{1}{t} \\
&= H_k + H_{n-k+1} - 1
\end{aligned}
$$

Thirdly, we calculate $\mathbb{E}[k \text{ is a leaf}]$ k is a leaf if and only if $k-1$ and $k+1$(if they exist) occur before k. So

$$
\mathbb{E}[k \text{ is a leaf}] = \begin{cases} 1 & n = 1, \\ \frac{1}{2} & n \neq 1 \text{ and } (k = 1 \text{ or } k = n), \\ \frac{1}{3} & n \neq 1 \text{ and } 1 < k < n. \end{cases}
$$

Totally, we get

$$
\begin{aligned}
\mathbb{E}[C] &= 2\sum_{1\le i<j\le n}\mathbb{E}[B_{i,j,k}] + \sum_{i=1}^{n}\mathbb{E}[A_{i,k}] - \mathbb{E}[k \text{ is a leaf}] \\
&= 2(n - (n-k+3)H_{n-k+1} + (n+1)H_n - (k+2)H_k + 3) + H_k + H_{n-k+1} - 1 + \mathbb{E}[k \text{ is a leaf}] \\
&= \begin{cases} 0 & n = 1, \\ 2n + 2(n+1)H_n - (2n-2k+5)H_{n-k+1} - (2k+3)H_k + 9/2 & n \neq 1 \text{ and } (k = 1 \text{ or } k = n), \\ 2n + 2(n+1)H_n - (2n-2k+5)H_{n-k+1} - (2k+3)H_k + 14/3 & n \neq 1 \text{ and } 1 < k < n. \end{cases}
\end{aligned}
$$

We have verified the answer using the program below:

```
void prepare() {
    for(int i = 1; i < N; i++) // Calculate the harmonic sequence
        H[i] = H[i - 1] + fraction(1, i);

    dp[1][1] = fraction(0, 1); // Calculate dp[i][j]
    for(int i = 2; i < N; i++)
        for(int j = 1; j <= i; j++){
            fraction sum(0, 1);
            for(int k = 1; k <= i; k++){
                if (k > j) sum = sum + dp[k - 1][j];
                if (k < j) sum = sum + dp[i - k][j - k];
            }
            sum = sum / fraction(i, 1);
            dp[i][j] = sum + fraction(i, 1);
        }
}
fraction get_ex8_ans(int n, int k) { // Calculate answer using the formular in ex8
    if (n == 1) return fraction(0, 1);
    fraction ans(0, 1);
    if (k == 1 || k == n) ans = ans - fraction(1, 2);
    else ans = ans - fraction(1, 3);
    for(int i = 1; i <= n; i++)
        for(int j = 1; j <= n; j++)
            ans = ans + fraction(1, max(i,j,k) - min(i,j,k) + 1);
    return ans;
}
fraction get_ex9_ans(int n, int k) { // Calculate answer using the formular in ex9
    if(n == 1) return fraction(0, 1);
    fraction ans(2 * n + 5, 1);
    ans = ans + fraction(2 * n + 2, 1) * H[n];
    ans = ans - fraction(2 * n - 2 * k + 5) * H[n - k + 1];
    ans = ans - fraction(2 * k + 3) * H[k];
    if(k == 1 || k == n) ans = ans - fraction(1, 2);
    else ans = ans - fraction(1, 3);
    return ans;
}
int main() {
    prepare();
    int n, k;
    cin >> n >> k;
    fraction standed_ans= dp[n][k];
    fraction ex8_ans = get_ex8_ans(n, k);
    fraction ex9_ans = get_ex9_ans(n, k);
    //these three ans are the same
    cout << "E = " << ex9_ans.x << " / " << ex9_ans.y << endl;
}
```

The program output the answer for $1 \le n \le 5$ and $1 \le k \le n$:

$n \backslash k$	1	2	3	4	5
1	0/1				
2	2/1	2/1			
3	11/3	13/3	11/3		
4	65/12	13/2	13/2	65/12	
5	433/60	173/20	136/15	173/20	433/60

Table 1: The table of answer for $1 \leq n \leq 5, 1 \leq k \leq n$