Homework 2

Obliviate

March 2021

1 Solution:

Let S be the set of all candidates for maximum, which contains n items at the beginning.

After each comparison, we will remove the smaller item from S if necessary and then $|S|$ decreased by at most 1.

Suppose that there is an algorithm which does $n-2$ or fewer comparisons, then $|S|$ will be at least 2 after all comparisons. This implies that we cannot find the exact maximum.

2 Solution:

First, we compare $A[2k-1]$ with $A[2k]$ for $1 \leq k \leq \frac{n}{2}$. Let's put these smaller ones into set P and bigger ones into set Q. Obviously, the biggest number of A will be in set P and the smallest will be in set Q. That is to say, the biggest number of P is also the biggest number of A and the smallest number of Q is also the smallest number of A.

Building two sets costs $\frac{n}{2}$ comparisons. Finding the biggest(smallest) number in set $P(Q)$ costs $\frac{n}{2}-1$ comparisons. In all, we used $\frac{n}{2} + 2 \times (\frac{n}{2} - 1) = \frac{3n}{2} - 2$ comparisons.

3 Solution:

It can be done by a recursive algorithm. For $A[1, 2k]$, we divide it into $A[1, k]$ and $A[k+1, 2k]$. Firstly we find the maximal element recursively and then compare the maximal element of each part. This procedure will cost $n-1$ comparisons.

Consider our method to find the maximum. Every comparison means the smaller element can't be the biggest. Obviously, the second biggest element means that it's not the biggest but it's bigger than any other elements. Therefore, the second biggest element must have compared with the biggest element, which is the only way to tell us it's not the biggest.

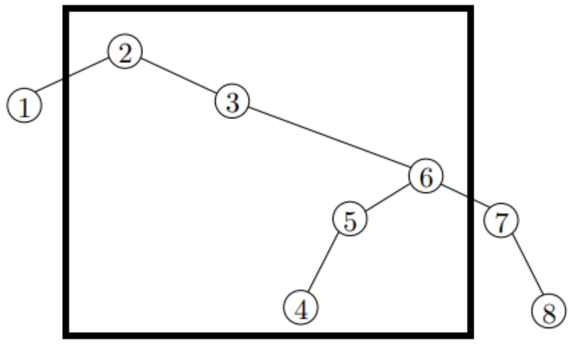
Thus we can simply find the maximal among all all $\log_2(n)$ numbers which had been compared with the maximal element in the recursive algorithm. This step will cost us $\log_2(n) - 1$ comparisons.

The total algorithm finds the second largest element with $n + \log_2(n) - 2$ comparisons.

4 Solution:

QuickSort algorithm always do both the left and the right part of the sub-QuickSort, while QuickSelect algorithm only does one part. So QuickSelect can be viewed as a partial execution of QuickSort.

Sample QuickSort tree is shown by the picture below. The part in the rectangular is the part visited by QuickSelect(4).



quicksort tree of $[2, 3, 6, 1, 7, 5, 8, 4]$

5 Solution:

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

Since i is an ancestor of j iff i is the first one appears in all numbers between i and j, i is an ancestor of k iff i is the first one appears in all numbers between i and k; $B_{i,j,k} = 1$ means i must be the first one appears in all numbers between $\min(i, j, k)$ and $\max(i, j, k)$, otherwise $B_{i,j,k} = 0$.

So $\mathbb{E}[B_{i,j,k}] = 1 \times \frac{1}{\max(i,j,k)-\min(i,j,k)+1} + 0 = \frac{1}{\max(i,j,k)-\min(i,j,k)+1}$.

6 Solution:

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

For each ancestor of k(including itself), we call it i, then all elements in its subtree will compare with i (except itself). So we need to count the number of pair (i, j), which satisfies that i is an ancestor of k and j.

7 Solution:

$$\begin{aligned}
Ans = \mathbb{E}[C(\pi,1)] &= \sum_{i\neq j} \mathbb{E}[B_{i,j,1}] \\
&= \sum_{i\neq j} \frac{1}{\max(i,j)-1+1} = 2\cdot\sum_{i>j}\frac{1}{i} \\
&= 2\cdot\sum_{i=2}^{n}\sum_{j=1}^{i-1}\frac{1}{i} = 2\cdot\sum_{i=2}^{n}\frac{i-1}{i} \\
&= 2\cdot\sum_{i=2}^{n}\left(1-\frac{1}{i}\right) \\
&= 2\cdot\left(n-1-\sum_{i=2}^{n}\frac{1}{i}\right) \\
&= 2\cdot\left(n-\sum_{i=1}^{n}\frac{1}{i}\right) \\
&= 2\cdot(n-\ln(n)+O(1))
\end{aligned}$$

8 Solution:

$$\begin{aligned}
\mathbb{E}[C(\pi,k)] = \sum_{i\neq j}\mathbb{E}[B_{i,j,k}] \\
&= \sum_{i\neq j}\frac{1}{\max(i,j,k)-\min(i,j,k)+1} \\
&= 2\cdot\left(\sum_{i<j<k}\frac{1}{k-i+1}+\sum_{i\leq k\leq j, i\neq j}\frac{1}{j-i+1}+\sum_{k<i<j}\frac{1}{j-k+1}\right)
\end{aligned}$$

Since

$$\begin{aligned}
\sum_{i<j<k}\frac{1}{k-i+1} &= \sum_{i\leq k-2}\frac{k-i-1}{k-i+1} = \sum_{i\leq k-2}\left(1-\frac{2}{k-i+1}\right) \\
&= k-2-2\sum_{d=3}^{k}\frac{1}{d} = k-2-2\left(H_k-\frac{3}{2}\right) \\
&= k+1-2H_k \\
\sum_{k<i<j}\frac{1}{j-k+1} &= \sum_{j\geq k+2}\frac{j-k-1}{j-k+1} = \sum_{j\geq k+2}(1-2\frac{1}{j-k+1}) \\
&= n-k-1-2\sum_{d=3}^{n-k+1}\frac{1}{d} = n-k-1-2\left(H_{n-k+1}-\frac{3}{2}\right) \\
&= n-k+2-2H_{n-k+1}
\end{aligned}$$

$$\begin{aligned}\sum_{i\leq k\leq j, i\neq j}\frac{1}{j-i+1} &= \sum_{l=2}^{\min(k,n-k+1)}\frac{l}{l} + \sum_{\min(k,n-k+1)}^{k}\frac{\min(k,n-l+1)-\max(1,k-l+1)+1}{l} \\ &= k(H_{n-k+1}-H_k)+(n+1)(H_n-H_{n-k+1})\end{aligned}$$

we have

$$\begin{aligned}\mathbb{E}[C(\pi,k)] &= 2\big(n+3-(k+2)H_k+(n+1)H_n-kH_k-(n-k+1)H_{n-k+1}\big) \\ &= 2\left(n+3+(k+2)\ln\frac{n}{k}+(n-k+3)\ln\frac{n}{n-k+1}-5H_n\right) \\ &= 2\left(n\left(1+\kappa\ln\frac{1}{\kappa}+\left(1-\kappa+\frac{1}{n}\right)\ln\frac{1}{1-\kappa+\frac{1}{n}}\right)+2\ln\frac{n}{k}+2\ln\frac{n}{n-k+1}-5H_n+3\right) \\ &= 2\Big(n\big(1+O(1)\big)+o(n)\Big)\end{aligned}$$