CS 217 – Algorithm Design and Analysis

Shanghai Jiaotong University, Fall 2019

NoCode

April 23, 2020

2 Sorting Algorithms

Exercise 1. Given an array A of n items (numbers), we can find the maximum with $n-1$ comparisons (this is simple). Show that this is optimal: that is, any algorithm that does $n-2$ or fewer comparisons will fail to find the maximum on some inputs.

Solution A number is known as the maximum only if everyone else is knwon smaller than that number directly or indirectly. If only $n-2$ or fewer comparisons taken, there must be at least two numbers that are never compared directly or indirectly, which means that is impossible to determine the maximum.

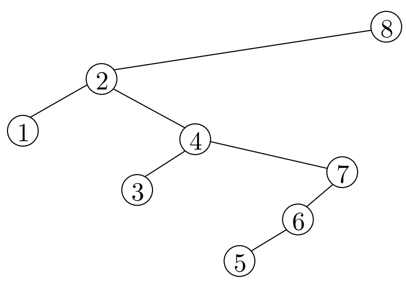
Exercise 2. Let A be an array of size n, where n is even. Describe how to find both the minimum and the maximum with at most $\frac{3}{2}n-2$ comparisons. Make sure your solution is *simple*, in describe it in a clear and succinct way!

Solution Suppose the $A = \{a_1, a_2, \cdots, a_n\}$, first we divide A in $\frac{n}{2}$ group $\{(a_1, a_2), (a_3, a_4), \cdots, (a_{n-1}, a_n)\}$, then we compare each group, put each larger element in B, and put each smaller element in C, this take $\frac{n}{2}$ comparisons. Then we compare every two adjacent elements in B and C,then we can get the maximum element in B and the minimum element in C, which takes $(\frac{n}{2}-1)*2$ comparisons. Because the minimum element in A is the minimum element in C, the maximum element in A, so we find the maximumand the minimum with at most $\frac{3n}{2}-2$ comparisons

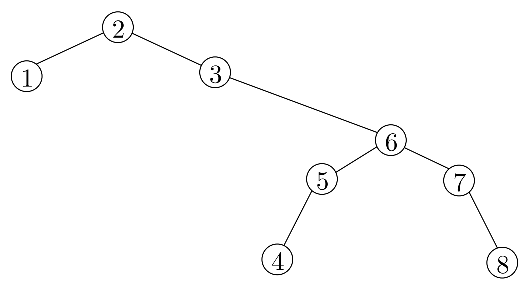
Exercise 3. Given an array A of size $n = 2^k$, find the second largest element element with at most $n + \log_2(n)$ comparisons. Again, your solution should be *simple*, and you should explain it in a clear and succinct way!

Solution First we divide A in 2^{k-1} group $\{(a_1, a_2), (a_3, a_4), \cdots, (a_{n-1}, a_n)\}$, we compare each group, put each larger element in A_1. Then we continuely divide A_1 in 2^{k-2} group, compare each group and put each larger element in A_2. We repeatedly do this until we find the maximum element M, which totally takes n comparisons. In this process, we have $log_2 n = k$ elements compared with M in some time. We only need to find the maximum in these k elements, which is the second largest element in A. This take $k - 1$ comparisons. So we totally take at most $n + log_2 n$ comparisons to find the second largest element.

Recall the quicksort tree defined in the lecture.



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



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

We denote a specific list (ordering) by π and the tree by $T(\pi)$. $A_{i,j}$ is an indicator variable which is 1 if i is an ancestor of j in the tree $T(\pi)$, and 0 otherwise. In the lecture, we have derived:

$$\begin{aligned} \mathbb{E}[A_{i,j}] &= \frac{1}{|i-j|+1} \\ \text{total number of comparisons } &= \sum_{i \neq j} A_{i,j} \ . \end{aligned}$$

Exercise 4. Determine the expected number of comparisons made by quicksort. Your final formula must be *closed*, meaning it must not contain $\mathbb{E}$, $\prod$, or $\sum$. It may, however, contain $H_n := 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$, the n^{th} Harmonic number. **Remark.** This gets a bit tricky, and you will need some summation wizardry towards the end.

2.1 Quickselect

Remember the recursive algorithm QuickSelect from the lecture. I write it below in pseudocode. In analogy to quicksort we define QuickSelect deterministically and assume that the input array is random, or has been randomly shuffled before QuickSelect is called. We assume that A consists of distinct elements and $1 \leq k \leq |A|$.

Algorithm 1 Select the k^{th} smallest element from a list A

```
procedure QuickSelect(X, k)
    if |X| = 1 then
        return X[1]
    else:
        p := X[1]
        Y := [x ∈ X | x < p]
        Z := [x ∈ X | x > p]
        if |B| = k − 1 then
            return p
        else if |Y| ≥ k then
            return QuickSelect(Y, k)
        else
            Return QuickSelect(Z, k − |Y| − 1)
        end if
    end if
end procedure
```

Let C be the number of comparison made by QuickSelect. In the lecture we proved that $\mathbb{E}[C] \leq O(n)$ when we run it on a random input.

Exercise 5. Explain how QuickSelect can be viewed as a "partial execution" of quicksort with the random pivot selection rule. Draw an example quicksort tree and show which part of this tree is visited by QuickSelect.

Let $B_{i,j,k}$ be an indicator variable which is 1 if i is a common ancestor of j and k in the quicksort tree. That is, if both j and k appear in the subtree of $T(\pi)$ rooted at i.

Exercise 6. What is $\mathbb{E}[B_{i,j,k}]$? Give a succinct formula for this.

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

Exercise 7. Let $C(\pi, k)$ be the number of comparisons made by QuickSelect when given π as input. Design a formula for $C(\pi, k)$ with the help of the indicator variables $A_{i,j}$ and $B_{i,j,k}$ (analogous to the formula $\sum_{i \neq j} A_{i,j}$ for the number of comparisons made by quicksort).

Solution the number of comparisons made by QuickSelect is

$$\sum_{i \neq j} B_{i,j,k}$$

In QuickSort, i will compare with j if i is the ancester of j in quicksort tree. But it is not true in QuickSelect, because if k and j not in the same subtree rooted by i. The QuickSelect will not enter this subtree because he know k in other subtree rooted by i's parent.It means $B_{i,j,k} = 0$.

And if k and j in the same subtree rooted by i, then QuickSelect will must choose the subtree. It means $B_{i,j,k} = 1$.

So the answer is the sum of $B_{i,j,k}$

Exercise 8. Suppose we use QuickSelect to find the minimum of the array. On expectation, how many comparisons will it make? Give an answer that is exact up to additive terms of order $o(n)$. You can use the fact that $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln(n) + o(1)$.

Solution Accroding to Ex7,

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

So it makes $2n - \ln n + o(1)$ comparisons.

Exercise 9. Derive a formula for $\mathbb{E}_\pi[C(\pi, k)]$, up to additive terms of order $o(n)$. You might want to introduce $\kappa := k/n$.