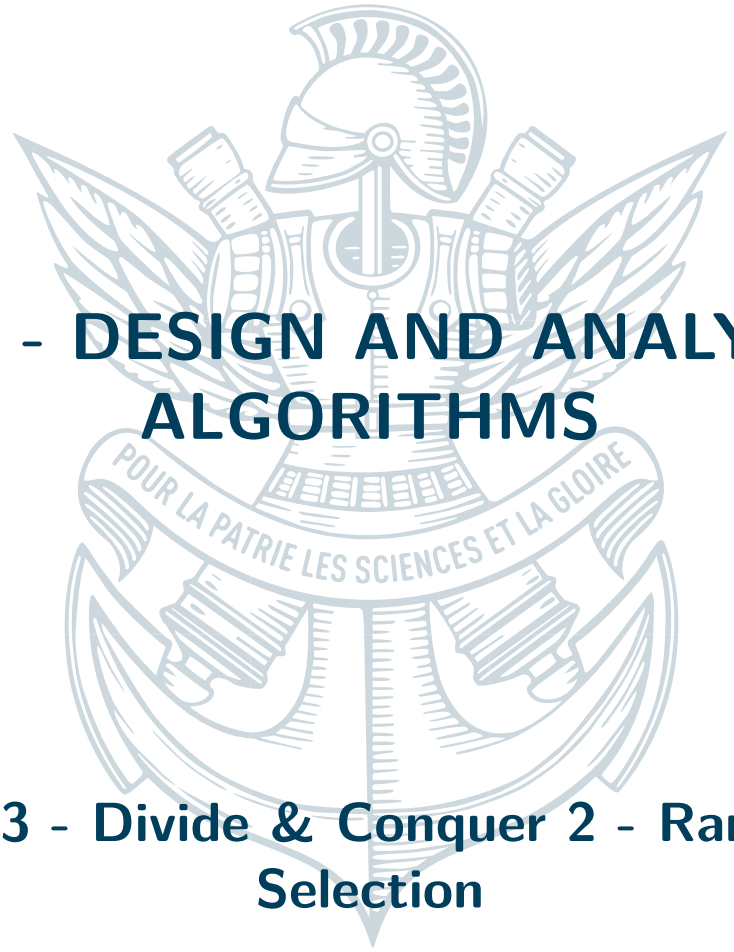

CSE202 - DESIGN AND ANALYSIS OF ALGORITHMS

Week 3 - Divide & Conquer 2 - Rankings, Selection

October 9, 2022

Yubo Cai

EXERCISE 1

We assume A and B be two databases with n numerical values and totally we have $2n$ elements. We are trying to find the nth smallest element out of all $2n$ elements. Let A_i and B_i be the $i-th$ smallest number of A and B.

let $k = \lfloor \frac{n}{2} \rfloor$, where we denote that m_1 and m_2 are the median for A and B where $m_1 = A_k$, $m_2 = B_k$. Since two databases are symmetry, therefore we assume $m_1 < m_2$. We get $(A_1...A_{k-1}) < m_1 < m_2$ and $m_1 < m_2 < (B_{k+1}...B_n)$. Therefore, the median of all elements is between m_1 and m_2. Then we denote $A' = \{A_{k+1}...A_n\}$ and $B' = \{B_1...B_{k-1}\}$. In this way, we can consider A' and B' as two small databases and apply the same algorithms to find the median in A' and B' which is also the median for A and B.

So we can have the pseudocode here

```
def findMedian(A, a_l, a_h, B, b_l, b_h):
    # here a_l, a_h represent the index of subarray
    if a_l == a_h:
        m1 = query(A ,1) # here we find the first element of A
        m2 = query(B ,1)
        return min(m1, m2)
    k1 = (a_l+a_h-1)//2
    k2 = (b_l+b_h-1)//2
    m1 = query(A, k1) # C(1)
    m2 = query(A, k2) # C(1)
    if m1 < m2:
        return findMedian(A, m1+1, a_h, B, b_l, m2) # C(n/2)
    else:
        return findMedian(A, a_l, m1, B, m2+1, b_h) # C(n/2)

# Initialization
findMedian(A, 1, n, B, 1, n)
```

Then we try to compute the complexity of the algorithms. We denote $C(n)$ is the number of queries that we need to use to compute the median of all numbers. Where we have the equation

$$C(n) = C(\lfloor \frac{n}{2} \rfloor) + \lambda$$

From the computation, we know that the complexity is $O(logn)$

EXERCISE 2

We first discuss the case $k = 3$. We apply the similar algorithms from the lecture that

$$\underbrace{\begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,q} \\ A_{2,1} & \cdots & \cdots & \\ A_{3,1} & A_{3,2} & \cdots & A_{3,q} \end{pmatrix}}_{q=\frac{n}{3}}$$

We select p as the median of medians and all the elements are split into $\frac{n}{3}$ groups with size 3 as the graph shown above.

Since p is the median of medians of all numbers, there are $\frac{n}{3}/2 = \frac{n}{6}$ medians of each group less than p. Also, for all those medians there is one number less than the medians in each group. We got at least $\frac{n}{6} + \frac{n}{6} = \frac{n}{3}$ elements less than p. For the same reason, we can deduce that there are $\frac{n}{6} + \frac{n}{6} = \frac{n}{3}$ elements greater than p.

We have the MOM selection algorithm recurrence that

1. select called recursively with $\frac{n}{3}$ elements to compute MOM p
2. at least $\frac{n}{3}$ elements less than p
3. at least $\frac{n}{3}$ elements greater than p
4. select called recursively with at most $n - (\frac{1}{3}n)$ elements

$$C(n) \leq \underbrace{C(\lfloor n/3 \rfloor)}_{\substack{\text{median of} \\ \text{medians}}} + \underbrace{C(n - \lfloor n/3 \rfloor)}_{\substack{\text{recursive} \\ \text{select}}} + \lambda n$$

We can deduce that

$$C(n) \leq C(\frac{n}{3}) + C(\frac{2n}{3}) + \lambda n$$

We have for every layer of the tree, the sum is still $C(n)$ with the height of $logn$. Therefore, the overall complexity is $O(n \cdot logn)$ which is even worse compared with the complexity of $k = 5$ of $O(n)$.

Then we discuss the case $k > 5$. We select p as the median of medians and all the elements are split into $\frac{n}{k}$ groups with size k.

Since p is the median of medians of all numbers, there are $\frac{n}{k}/2 = \frac{n}{2k}$ medians of each group less than p. Also, for all those medians there is $\frac{k-1}{2}$ number less than the medians in each group. Therefore, we have at least $\frac{n(k+1)}{4k}$ elements greater than p and at least $\frac{n(k+1)}{4k}$ elements less than p.

Therefore, we get the following recurrence formula that

$$C(n) \leq C(\lfloor n/k \rfloor) + C(n - \lfloor \frac{n(k+1)}{4k} \rfloor) + \lambda n$$

$$C(n) \leq C(\frac{n}{k}) + C(\frac{(3k-1)n}{4k}) + \lambda n$$

With super-additive property of $C(x) + C(y) \leq C(x+y)$ we got $C(n) \leq C(\frac{(3k+3)n}{4k}) + \lambda n$. As $n \to \infty$, we have $\frac{(3k+3)n}{4k} \to \frac{3}{4}n$. Therefore the complexity converge to $O(n)$.