9.Medians and order statistics

Hsu, Lih-Hsing

- The *i*th **order statistic** of a set of n element is the *i*th smallest.
- The **selection problem** can be specified formally as follows:
  - **Input:** A set of n (distinct) numbers and a number i, with $1 \le i \le n$.
  - **Output:** the element $x \in$ A that is larger than exactly i - 1 other elements of A.

9.1 Minimum and Maximum

MINIMUM(A)

1 $min \leftarrow A[1]$

2 **for** $i \leftarrow 2$ **to** length[A]

3 **do if** $min > A[i]$

4 **then** $min \leftarrow A[i]$

5 **return** min

Analysis: $^{*}O(n)$

- The expected number of times that line 4 is executed is $O(\log n)$.

- The second smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.
- $\left\lceil \frac{3n}{2} \right\rceil - 2$ comparisons are necessary in the worst case to find both the maximum and the minimum of n elements.

9.2 Selection in expected linear time

```
RANDOMIZED_SELECT(A,p,r,i)
if p = r
    then return A[p]
q ← RANDOMIZED_PARTITION(A, p, r)
k ← q − p + 1
if i = k          ► the pivot value is the answer
    then return A[q]
elseif i < k
    then return RANDOMIZED_SELECT(A,p,q-1,i)
else return RANDOMIZED_SELECT(A,q+1,r,i-k)
```


Analysis

For $k = 1, 2, \ldots, n$, we define indicator random variables X_k where $X_k = \mathrm{I}$ {the subarray A[$p..q$] has exactly k elements} ,
and so we have

$$E[X_k] = 1/n .$$

$$T(n) \le \sum_{k=1}^{n} X_k \cdot (T(\max(k-1, n-k)) + O(n))$$

$$= \sum_{k=1}^{n} (X_k \cdot T(\max(k-1, n-k)) + O(n))$$

Taking expected values, we have

$$
\begin{aligned}
&E[T(n)] \\
&\le E[\sum_{k=1}^{n} X_k \cdot T(\max(k-1)) + O(n)] \\
&= \sum_{k=1}^{n} E[X_k \cdot T(\max(k-1, n-k))] + O(n) \\
&= \sum_{k=1}^{n} E[X_k] \cdot E[T(\max(k-1, n-k))] + O(n) \\
&= \sum_{k=1}^{n} \frac{1}{n} \cdot E[T(\max(k-1, n-k))] + O(n)
\end{aligned}
$$

$$
\max(k-1, n-k) = \begin{cases} k-1 & if \quad k > \lceil n/2 \rceil \\ n-k & if \quad k \le \lceil n/2 \rceil \end{cases}
$$

$$
E[T(n)] \le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)
$$

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor} k \right) + an \\
&= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2 \rfloor - 1)\lfloor n/2 \rfloor}{2} \right) + an \\
&\le \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(n/2-2)(n/2-1)}{2} \right) + an \\
&= \frac{2c}{n}\left(\frac{n^2-n}{2} - \frac{n^2/4 - 3n/2 + 2}{2} \right) + an \\
&= \frac{c}{n}\left(\frac{3n^2}{4} + \frac{n}{2} - 2 \right) + an \\
&= c\left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n} \right) + an \\
&\le \frac{3cn}{4} + \frac{c}{2} + an \\
&= cn - \left(\frac{cn}{4} - \frac{c}{2} - an \right).
\end{aligned}
$$

We choose the constant c so that c/4 - a > 0, i.d., c>4a, we can Divide both sides by c/4 – a, giving

$$n \geq \frac{c/2}{c/4-a} = \frac{2c}{c-4a}.$$

Thus, if we assume that T(n)=O(1) for $n < 2c/(c\text{-}4a)$, we have T(n) = O(n).

Another analysis

$$\begin{aligned} T(n) &\leq \frac{1}{n}[T(\max(1,n-1)) + \sum_{k=1}^{n-1} T(\max(k,n-k))] + O(n) \\ &\leq \frac{1}{n}[T(n-1) + 2\sum_{k=1}^{n-1} T(k)] + O(n) \\ &= \frac{2}{n}\sum_{k=\lceil \frac{n}{2} \rceil}^{n} T(k) + O(n) \end{aligned}$$

GUESS $T(n) \leq cn \Rightarrow T(n) = O(n)$

$$T(n) \le \frac{2}{n}\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} ck + O(n) \le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil \frac{n}{2} \rceil - 1} k\right) + O(n)$$

$$\le \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lceil \frac{n}{2} \rceil - 1)\lceil \frac{n}{2} \rceil}{2}\right) + O(n)$$

$$\le c(n-1) - \frac{c}{n}\left(\frac{n}{2} - 1\right)\left(\frac{n}{2}\right) + O(n)$$

$$= c\left(\frac{3}{4}n - \frac{1}{2}\right) + O(n) \le cn$$

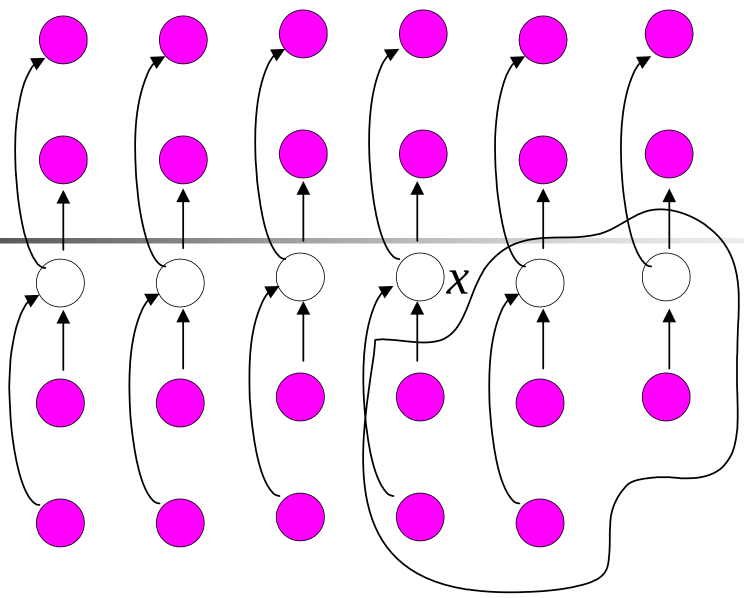
Pick c large enough so that $c(\frac{n}{4} + \frac{1}{2})$ dominates the O(n) term.

9.3 Selection in worst-case linear time

Algorithm

1. Divide the n elements of the input array into $\lfloor n/5 \rfloor$ groups of 5 elements each and at most one group made up of the remaining n **mod** 5 elements.
2. Find the median of each of the $\lceil n/5 \rceil$ groups by first insertion sorting the element of each group and picking its median.
3. Use SELECT recursively to find the median x of the $\lceil n/5 \rceil$ medians found in Step 2.

4. Partition the input array around the ***median-of-medians*** x using a modified version of partition. Let k be the number of elements on the lower side of the partition, so that x is the kth smallest element and there are n - k elements on the high side of the partition.
5. If $i = k$, then return x. Otherwise, use SELECT recursively to find the ith smallest element on the low side if $i < k$, or $(i\text{-}k)$th smallest element on the high side if $i > k$.



at least

$$3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$$

nodes

$$T(n) \leq \begin{cases} \Theta(1) & \text{if } n \leq 140 \\ T(\frac{n}{5}) + T(\frac{7n}{10} + 6) + O(n) & \text{if } n > 140 \end{cases}$$

Assume that $T(n) \le cn$ for some c and small $n \le 140$.

$$T(n) \le c\lceil n/5 \rceil + c(7n/10+6) + an$$

$$\le cn/5 + c + 7cn/10 + 6c + an$$

$$\le 9cn/10 + 7c + an$$

$$= cn + (-cn/10 + 7c + an),$$

which is at most cn if

$$-cn/10 + 7c + an \le 0$$

Pick c large enough so that $c(\frac{n}{10} - 7)$ is larger that the function described by O(n) term for all $n>140$.

$\Rightarrow T(n) = O(n)$

- QUICKSORT can be improved into $O(n \log n)$.