2.2. k-Select, Apx Median, QuickSort

Given an array of n elements $A[1..n]$ and a parameter k ($k \in \{1, \dots, n\}$), return the element with rank k. ⇒ k-Select problem

$A = \{7, 0, 4, 12, 9 \mid 3, 8, 15, 26, 100 \mid 2, 11, 6, 870, 25\}$

$A' = \{7, 15, 13\}$

exact median

$A_1 = \{7, 1, 4, 9, 3, 8, 2, 6\}$ $|A_1| = 8$

$A_2 = \{12, 15, 26, 100, 870, 25\}$ $|A_2| = 6$

Def. Algorithm

Def Select($A[n], k$)

1. groups the elements in A into $\frac{n}{5}$ groups of size 5 $O(n)$
2. compute the median for each group (e.g. by sorting) $O(\frac{n}{5} \cdot c) \in O(n)$
3. recursively apply 1. and 2. until $T(\frac{n}{5})$ the median of medians, call this p
4. use p to split A in
   $A_1 := \{a \in A \mid a < p\}$ and
   $A_2 := \{a \in A \mid a > p\}$ $O(n)$
5. if $|A_1| = k - 1$
   return p; $O(n)$
6. if $|A_1| > k$

Det Select (A_1, k) $T(?)$

else

Det Select $(A_2, k - |A_1| - 1)$

Lemma. Det Select runs in $O(n)$.

$T(n, k)$

$T(n) = \max_{k=1,\dots,n} T(n,k)$

What is worst-case in step 6?

$$T(n) = c \cdot n + T\left(\frac{n}{5}\right) + T(?)$$

p is the median of medians

$m = \frac{n}{5} \Rightarrow$ in at least $\lceil \frac{m}{2} \rceil$ of the groups at least 3 of 5 elements are $\le p$. Hence a lower bound for the number of elems in A_1 is $3\lceil \frac{m}{2} \rceil \ge \frac{3n}{10}$

And the same bound holds for the # of elems $\ge p$.

Therefore at most $\frac{7n}{10}$ elements can be in the subarray we recurse on.

So we get

$$T(n) \le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$$

Claim: $T(n) \le 10cn$

Proof by induction.

- for $n = 5$ trivially true.
- induction step

$$T(n+1) \le c(n+1) + T\left(\frac{n+1}{5}\right) + T\left(\frac{7(n+1)}{10}\right)$$

ind. hypothesis

$$\le c(n+1) + 10c \cdot \left(\frac{n+1}{5}\right) + 10c\,\frac{7\cdot(n+1)}{10}$$

$$= c(n+1) + 2c(n+1) + 7c(n+1)$$

$$= 10c(n+1) \quad //$$

So $T(n) \in O(n)$ □

Approximative Median

Given an array of n elements $A[n]$, and some tolerance δ, find an element which in the sorted sequence would be at most δ positions away from the median.

MC Apx Median

pick r u.a.r. in $\{1, \dots, n\}$
$a \leftarrow A[r]$
$rank \leftarrow 1$
for $i = 1$ **to** n **do**
  if $A[i] < a$ **then**
    $rank++$
if $rank \in \left[\frac{n}{2} - d, \frac{n}{2} + d\right]$ **then**
  return a_i
else
  return error.

Runtime: $O(n)$

Success probability

$$P(\text{success}) = \frac{2d}{n}$$

could be increased by running the alg. c times $\Rightarrow$ runtime of $O(cn)$

$$\Rightarrow P(\text{fail}) = \left(1 - \frac{2d}{n}\right)^c$$

Can be turned easily into LV alg. by running MC until an element is returned.

Expected number of rounds until first success: $O\left(\frac{n}{\delta}\right)$

total exp. runtime: $O\left(\frac{n^2}{\delta}\right)$

Rand. 4-Select

instead of choosing p as the median of medians, choose p.u.a.r.

Lemma RandSelect runs expectedly in $O(n)$.

Proof: We always assume to have to recurse on the larger of the two subarrays A_1 and A_2.

↳ there are two possible pivots (min, max), which lead to $\max(|A_1|, |A_2|) = n-1$

↳ there are two possible pivots such that $\max(|A_1|, |A_2|) = n-2$

...

$\Rightarrow$

$\cdots$ $\max(|A_1|, |A_2|) \leq \frac{n}{2}$

Total runtime:

$$T(n) = \frac{2}{n} \sum_{i=\frac{n}{2}}^{n-1} T(i) + cn$$

$T(n) \in O(n)$ proof by induction.

For $n=1$ correctness is obvious.

Now consider n

$$T(n+n) = \frac{2}{n+n} \sum_{i=\frac{n+n}{2}}^{n-n} T(i) + c(n+n)$$

$$= \underbrace{\frac{2}{n+n} \sum_{i=\frac{n+n}{2}}^{n-n} T(i) + cn + \frac{2}{n+n} T(n) + c}_{< T(n)}$$

$$T(n+n) < T(n) + \frac{2}{n+n} T(n) + c$$

$$T(n+n) \in O(n) \quad //$$

A very similar principle can be used for

Sorting.

```
QuickSort (A)
begin
    if |A| ≤ 1 then
        return A;
    select p u.a.r.
    A1 ← {a ∈ A | a ≤ p}
    A2 ← {a ∈ A | a > p}
    return QuickSort(A1) + p + QuickSort(A2);
```

if p would always be the median the runtime would be

$$T(n) = T(n/2) + T(n/2) + O(n)$$

$$\Rightarrow T(n) \in O(n \log n)$$

Wrong analysis: Expected rank of u.a.r chosen element is $n/2$, and so best case formula from above applies

WRONG because same analysis would hold for choosing min or max each with a prob. of $1/2$. → But then runtime $O(n^2)$

Lemma: QuickSort runs expectedly in $O(n \log n)$.

Proof. If you choose p. u.a.r. we have the same chance for splitting A into $|A_1|/|A_2|$ being $0, n-1$, as $1, n-2$, ... , and so on.

In general for p with rank i we get $i \,/\, n-i-1$.

Expected runtime:

$$T(n) = \frac{1}{n} \sum_{i=0}^{n-1} \left(T(i) + T(n-i-1) \right) + cn$$

$$\Leftrightarrow \quad T(n) = \frac{2}{n} \sum_{i=1}^{n-1} T(i) + cn$$

We will prove that $T(n) \leq 2n \log n + O(n)$.

$n \geq 1$ clear.

$$T(n) \leq \frac{2}{n} \sum_{i=1}^{n-1} \left(2i \log i + O(i) \right) + cn \qquad \text{(ind. hyp.)}$$

$$\leq \frac{2}{n} \sum_{i=1}^{n-1} 2i \log n + \frac{2}{n} \sum_{i=1}^{n-1} O(i) + cn$$

$$= \frac{4 \log n}{n} \sum_{i=1}^{n-1} i + \frac{2}{n} \sum_{i=1}^{n-1} O(i) + cn$$

$$\leq \frac{4\log n}{n} \cdot \frac{n^2}{2} + \frac{2}{n}\frac{O(n^2)}{2} + cn$$

$$\leq 2\log n \cdot n + O(n)$$

$$\Rightarrow T(n) \simeq 2n\log n + O(n) \quad //$$

Alternative proof.

Let $a_1, \dots, a_n$ be the sorted sequence of elements A.

We want to count the number of comparisons.

X_{ij} denote the rand. variable which is 1 if a_i and a_j are compared and 0 otherwise

$$T(n) = \sum_{j=1}^{n} \sum_{i<j} E(X_{ij})$$

$$E(X_{ij}) = P(X_{ij} = 1)$$

$$P(X_{ij} = 1) = \frac{2}{j - i + 1}$$

3, 3, 11, 20, 6, 1

(1, 3, 6, 7, 11, 20)

$$T(n) = \sum_{j=1}^{n} \sum_{i<j} E(X_{ij})$$

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

$$= \sum_{j=1}^{n} \sum_{i<j} \frac{2}{i} \leq \sum_{j=1}^{n} 2H(j)$$

$$\leq n \cdot 2H(n) \approx 2n \log n + 2n$$ //

Sorting Lower Bounds

Model: Operations are only "$a < b$"? and based on that decision rearrangements of the elements

"comparison-based"

Every det. comparison-based alg. needs in worst-case $\Omega(n \log n)$ steps.

Idea: $n!$ input sequences for an instance

$\log(n!)$ many steps to do to distinguish between them all,

because for any question "$a_i < a_j$?" only $\frac{k}{2}$ actions/sets of valid perm.

can be ruled out.

$$\Rightarrow \log_2(n!) = \log_2(n) + \log_2(n-1) + \dots + \log_2(1)$$

$$= \Omega(n \log n)$$

Example $n=3$ $\rightsquigarrow$ six possible input permutations

$\{123\}, \{132\}, \{213\}, \{231\}, \{312\}, \{321\}$

"$A[0] < A[1]$?"

$\{123\}$ $\{132\}$ $\{231\}$

"$A[1] > A[2]$?"

$\{132\}, \{231\}$

"$A[0] > A[2]$?"

$\{231\}$