Data Structures and Algorithms: Lecture 9

Barbara Morawska

September 25, 2018

Medians and order statistics

Definitions

- i'th order statistic of a set of n elements is the i'th smallest element of this set.
- minimum is the first order statistic ($i = 1$)
- maximum is the n'th order satistic
- median ("half-way point") is an i'th order statistic where:
  - if n is odd: $i = \dfrac{n+1}{2}$
  - if n is even: two medians
    - lower median $i = \lfloor n + 1/2 \rfloor$
    - upper median $i = \lceil n + 1/2 \rceil$

Problem

Select i'th order statistic from a set of n numbers.

Problem

Input:

A set A of n distinct numbers and an integer i, $1 \leq i \leq n$.

Output:

$x \in A$ such that, x is larger than exactly $i - 1$ elements in A (i's order statistic of A)

Obvious solution

Sort A and index i'th element in $O(n \lg n)$ with HEAPSORT or MERGE-SORT

We are looking for faster solutions.

Minimum and maximum

Assume we have a set of A with distinct numbers.
A is represented by an array.

To find minimum we need $n - 1$ comparisons

Procedure MINIMUM(A)

```
min = A[1]
for i = 2 to A.length do
    if min > A[i] then
        min = A[i]
return min
```

$n - 1$ comparisons is optimal.
MAXIMUM is computed in the same way.

Find MAXIMUM and MINIMUM at the same time

Each element compare with min and max variable

$(n-1)+(n-1) = 2n-2 = \Theta(n)$
Hence two comparisons for each element.

Smaller number of comparisons:

- Take two elements from the array: a, b
- Compare them: if $a < b$ then compare a with *min* and b with *max*
- Hence 3 comparisons for each 2 elements
- Together $3 \cdot \lfloor n/2 \rfloor$ comparisons
- How to initialize *min* and *max*?
  - if n is odd, choose $min = max = A[1]$
    $3 \cdot \lfloor n/2 \rfloor$ comparisons
  - if n is even, compare $A[1]$ with $A[2]$ and choose accordingly
    $1 + 3 \cdot (n-2)/2 = 3\frac{n}{2} - 2 \leq 3\lfloor n/2 \rfloor$ comparisons

Randomized selection in expected linear time

- Use divide and conquer approach
- Assume that the elements of input are distinct

Procedure RANDOMIZED-SELECT(A, p, r, i)

if $p == r$ **then**
    return $A[p]$
$q = \textsc{Randomized-Partition}(A, p, r)$
$k = q - p + 1$ `// k is the order of q`
if $i == k$ `// pivot is the answer`
then
    return $A[q]$
else if $i < k$ **then**
    return $\textsc{Randomized-Select}(A, p, q - 1, i)$
else $k < i$
    return $\textsc{Randomized-Select}(A, q + 1, r, i)$

Running time

Worst case: $\Theta(n^2)$

n times $\Theta(n)$ for unbalanced partition

Expected time:

- Let $T(n)$ be a random variable returning the running time.
- We have to compute $E[T(n)]$
- What is a recursion for $T(n)$?

Expected linear time

If $A[q]$ is chosen a pivot, the algorithm can:

- terminate ($A[q]$ is i'th smallest element)
- recurse on $A[p..q-1]$ – i'th smallest element is in this subarray
  the length is $q - 1 - p + 1 = q - p$
- recurse on $A[q+1..r]$ – i'th smallest element is in this subarray
  the length is $r - q$.

Assume that i'th element is always in the bigger subarray.

Hence if $A[q]$ is k'th smallest element:

the longer subarray has length $(k-1)$ or $(n-k)$
$\max(k-1, n-k)$

Expected linear time

Designing recursion for expected time

- If $A[q]$ (chosen as a pivot in the first step) is k'th smallest element, then $T(n) \leq T(\max(k-1, n-k)) + O(n)$
- But the recurrence is not correct, because k is different in every recursive call.
- We want to use an indicator variable for each possible k.
- Event: $A[q]$ is k'th smallest element in the array.
- Let X_k be an indicator variable associated with this event.
- Then the recurrence is:

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

Get rid of max

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

$k-1$ and $n-k$ have the same values for these cases.
Hence

$$\sum_{k=1}^{n} T(\max(k-1, n-k)) = 2 \sum_{k=\lceil n/2 \rceil}^{n} T(k-1) = 2 \sum_{k=\lfloor n/2 \rfloor}^{n-1} T(k)$$

Expected linear time

Hence we have to compute:

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

Since $E[X_k] = 1/n$, we have:

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

Solving recurrence by substitution

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

Guess $E[T(m)] = O(m)$ for $m < n$

induction step:

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

Expected linear time

$$E[T(n)] \leq cn - (\frac{cn}{4} - \frac{c}{2} - an)$$

We need to show when $\frac{cn}{4} - \frac{c}{2} - an \geq 0$.

- This is the case for $n \geq \frac{c/2}{c/4 - a} = \frac{2c}{c - 4a}$
- For $n < \frac{2c}{c - 4a}$ we assume $T(n) = O(1)$

Conclusion:

We can find any order statistic (e.g. median) in expected linear time. $E[T(n)] = O(n)$.

Selection in the worst case linear time

The procedure SELECT follows the following steps

The input is an array of numbers A, its starting index and its ending index, and a number i.

1. If the length of A is 1, then return $A[1]$
2. Divide the n elements into groups of 5 elements. $\lfloor n/5 \rfloor$ and one group of less than 5 elements.
3. Find the median of each of these groups
    - Sort these elements by Insertion-Sort
    - Choose the middle element as median
4. Find the median of medians by using Select recursively on the array of medians
5. Partition the unput array around the median of medians x using the deterministic modified version of Partition.

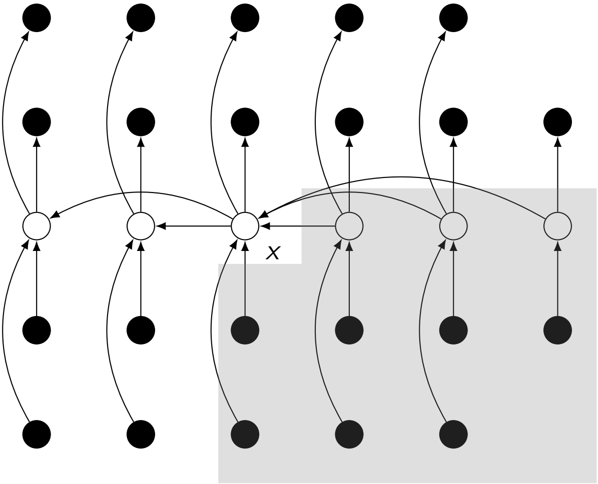
Selection in the worst case linear time

Select continued

Let k be such that x is the k'th smallest element in the array A. Hence there are $k-1$ smaller elements of A and $n-k$ bigger elements.

6. If $i = k$, return x. Otherwise,
   - if $i < k$ call Select on the lower subarray with the number i
   - if $i > k$ call Select on the upper subarray with the number $i - k$

Schema of SELECT



At least half - 1 of the medians are bigger than x.

How many elements are bigger/smaller than x?

- At least half of the medians are bigger or equal to x.
- Let us not count the group containing x and the group containing less than 5 elements
- Each of the other groups contribute at 3 elements to the elements bigger than x
- $3(\lceil \frac{1}{2} \cdot \lceil \frac{n}{5} \rceil \rceil \ - 2) \geq \frac{3n}{10} - 6$
- Also there is at least $\frac{3n}{10} - 6$ elements that are smaller than x.
- Worst case: the recursive call is called on the bigger part
- At most: $n - (\frac{3n}{10} - 6) = \frac{7n}{10} + 6$ elements
- Now we can define recurrence for the worst case for $\textsc{Select}$

Recurrence for SELECT

- Step 1,2,3,5 take $O(n)$ time (Step 3: $O(n)$ calls to INSERTION-SORT on 5 elements, hence each takes constant time)
- Step 4 takes time $T(\lceil n/5 \rceil)$
- Step 6 takes time at most $T(7n/10 + 6)$
- Assume that SELECT on an array with less than 140 elements takes constant time $O(1)$.

Recurrence:

$$T(n) \leq \begin{cases} O(1) & \text{if } n < 140 \\ T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n) & \text{if } n \geq 140 \end{cases}$$

Solving recurrence by subsitution

$T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$

Guess the solution $O(n)$

- Induction hypothesis: $T(m) \leq c \cdot m$ for any m such that, $140 \leq m < n$.
- To show: $T(n) \leq c \cdot n$

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c(7n/10 + 6) + an \\ &\leq cn/5 + c + 7cn/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an) \end{aligned}$$

This is smaller than cn for:

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

This is true for $n \geq 140$ and $c \geq 20a$.

Conclusion

- RANDOMIZED SELECTION and SELECTION run in linear time
- They are based on comparisons, but they are faster than comparison sort ($\Omega(n \lg n)$), because they select without sorting.