WIA2005: Algorithm Design and Analysis

Semester 2, Session 2016/17

Lecture 5: Order Statistics

Learning Objectives

- Order statistics
  - minimum, maximum
  - Randomized-Select Algorithm
  - Select Algorithm

Order statistics

- The i^{th} ***order statistic*** of a set of n elements is the i^{th} smallest element.
- For example, the ***minimum*** of a set of elements is the first order statistic (i = 1), and the ***maximum*** is the nth order statistic (i = n).
- A ***median***, informally, is the "halfway point" of the set.
  - n is odd (unique) $i = (n+1)/2.$
  - n is even, median occur at 2 places, i = n/2 and i = n/(2+1)
    - lower median $i = \lfloor (n+1)/2 \rfloor$
    - Upper median $i = \lceil (n+1)/2 \rceil$

Example

- Given n elements in an array, find the i th smallest number (element of rank i)

- Naïve algorithm
  - Sort A
  - return A[i]

Minimum and maximum

- How many comparisons are necessary to determine the minimum of a set of n elements?

```
MINIMUM(A)
min = A[1]
for i = 2 to A.length
    if min > A[i]
        min = A[i]
return min
```

Selection in expected linear time

- Randomized divide-and-conquer.
- The algorithm RANDOMIZED-SELECT is modeled after the quicksort algorithm.
- As in quicksort, the input array is partitioned recursively.
- But unlike quicksort, which recursively processes both sides of the partition, RANDOMIZED-SELECT works on only one side of the partition.

RANDOMIZED-SELECT

```
RANDOMIZED-SELECT(A, p, r, i)
if p == r
    return A[p]
q = RANDOMIZED-PARTITION(A, p, r)
k = q − p + 1
if i == k            // the pivot value is the answer
    return A[q]
elseif i < k
    return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

Intuition for analysis

- Assume that all elements are distinct.
- Lucky case: 1/10 : 9/10 partition.

  $T(n) = T(9/10) + \Theta(n)$

  $= \Theta(n)$

Master Methods: Case 3

- Unlucky case: 0 : n-1

  $T(n) = T(n-1) + \Theta(n)$

  $= \Theta(n^2)$ (arithmatic)

Analysis of RANDOMIZED-SELECT

- The procedure RANDOMIZED-PARTITION is equally likely to return any element as the pivot.
- Therefore, for each k such that 1≤ k≤ n, the subarray A[p..q] has k elements (all less than or equal to the pivot) with probability 1/n. For k = 1, 2,..,n, we define indicator random variables X_k where

$$X_k = \mathrm{I}\{\text{the subarray } A[p \mathinner{..} q] \text{ has exactly } k \text{ elements}\}$$

- and so, assuming that the elements are distinct,

$$\mathrm{E}[X_k] = 1/n$$

- Let T(n) be the random variable of Randomize-Select on n input size assuming they are independent

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

Taking expected values

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

- In order to apply equation (C.24), we rely on X_k and T(max(k-1, n-k)) being independent random variables.
- Let us consider the expression max T(max(k-1, n-k))

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

If n is even, each term from $T(\lceil n/2 \rceil)$ up to $T(n-1)$ appears exactly twice in the summation, and if n is odd, all these terms appear twice and $T(\lfloor n/2 \rfloor)$ appears once. Thus, we have

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

- Show E[T(n)] = O(n) by substitution
- Assume E[T(n)] ≤ cn for some constant c that satisfies the initial conditions of the recurrence.
- Using inductive hypothesis

$$
\begin{aligned}
T(n) &\le \frac{2}{n}\sum_{k=\lceil n/2\rceil}^{n-1} ck + O(n) \\
&\le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2\rceil-1} k\right) + O(n) \\
&= \frac{2c}{n}\left(\frac{1}{2}(n-1)n - \frac{1}{2}\left(\left\lceil\frac{n}{2}\right\rceil - 1\right)\left\lceil\frac{n}{2}\right\rceil\right) + O(n) \\
&\le c(n-1) - \frac{c}{n}\left(\frac{n}{2}-1\right)\left(\frac{n}{2}\right) + O(n) \\
&= c\left(\frac{3}{4}n - \frac{1}{2}\right) + O(n) \\
&\le cn ,
\end{aligned}
$$

Since we can pick *c* large enough so that *c*(*n*/4 + 1/2) dominates the *O*(*n*) term.

Selection in worst-case linear time

- Selection algorithm whose running time is O(n) in the worst case.
- Like RANDOMIZED-SELECT, the algorithm SELECT finds the desired ele- ment by recursively partitioning the input array.
- However, a good split is guaranteed upon partitioning the array.
- SELECT uses the deterministic partitioning algorithm PARTITION from quicksort, but modified to take the element to partition around as an input parameter.

SELECT algorithm

- SELECT(i,n)

1. Divide the n elements of the input array into floor(n/5) 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 ceiling(n/5) groups by first insertion-sorting the ele- ments of each group (of which there are at most 5) and then picking the median from the sorted list of group elements.
3. Use SELECT recursively to find the median x of the ceiling(n/5) medians found in step 2. (If there are an even number of medians, then by our convention, x is the lower median.)
4. Partition the input array around the median-of-medians x using the modified version of PARTITION. Let k be one more than the number of elements on the low side of the partition, so that x is the kth smallest element and there are nk 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 the (i-k)th smallest element on the high side if i > k.

Analysis of SELECT algorithm

- Step 1: determine a lower bound on the number of elements that are greater than the partitioning element x
- At least half of the medians found in step 2 are greater than or equal to the median-of-medians x.
- Thus, at least half of the ceiling(n/5) groups contribute at least 3 elements that are greater than x, except for the one group that has fewer than 5 elements if 5 does not divide n exactly, and the one group containing x itself.
- Discounting these two groups, it follows that the number of elements greater than x is at least

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

Thus, in the worst case, step 5 calls SELECT recursively on at most 7n/10 +6 elements.

SELECT algorithm in picture

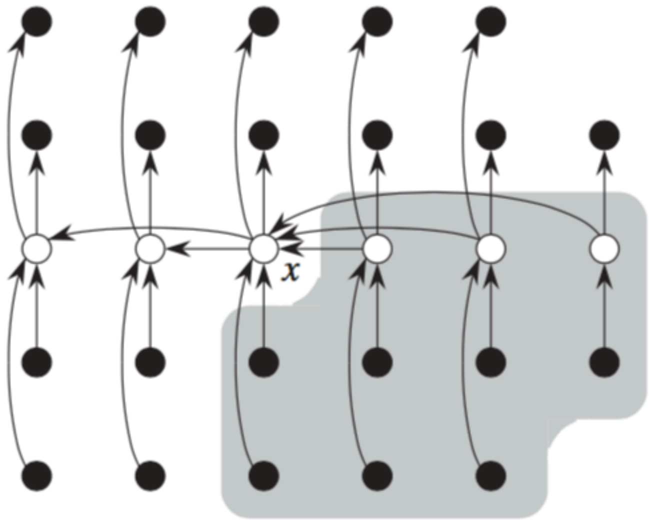


Figure 9.1 Analysis of the algorithm SELECT. The n elements are represented by small circles, and each group of 5 elements occupies a column. The medians of the groups are whitened, and the median-of-medians x is labeled. (When finding the median of an even number of elements, we use the lower median.) Arrows go from larger elements to smaller, from which we can see that 3 out of every full group of 5 elements to the right of x are greater than x, and 3 out of every group of 5 elements to the left of x are less than x. The elements known to be greater than x appear on a shaded background.

Worst-case running time for SELECT

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

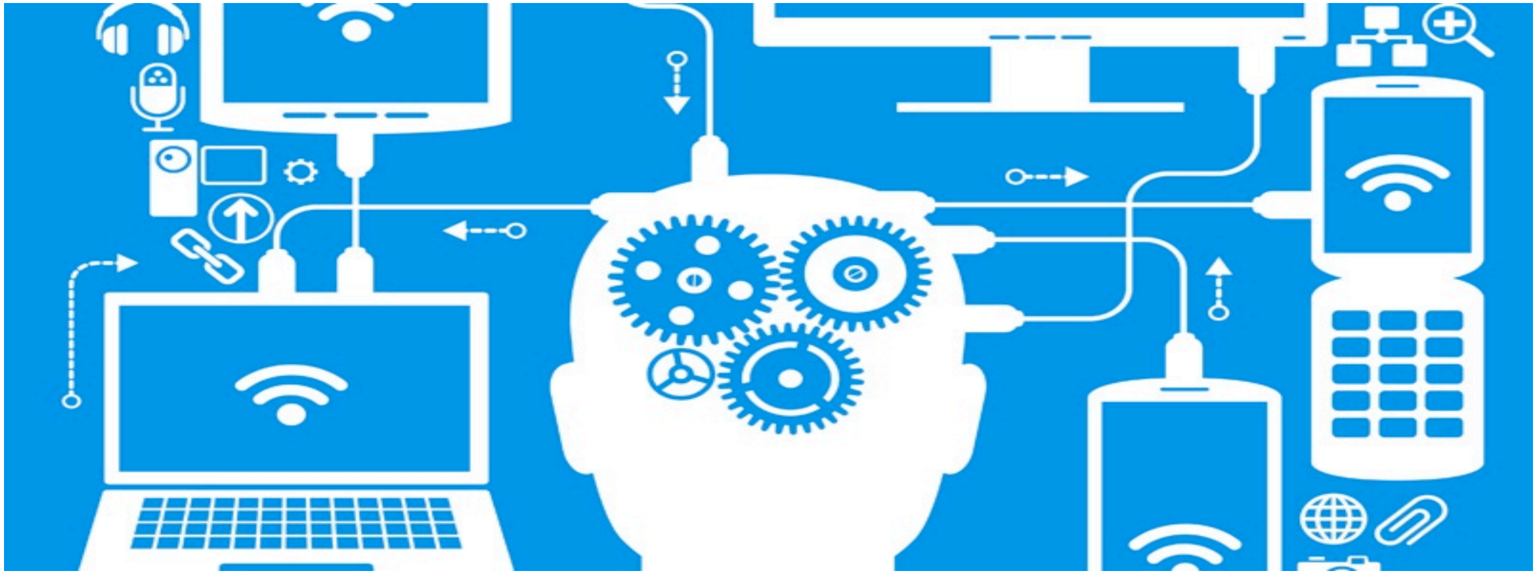
Show the running time is linear by substitution: T(n) ≤ cn for some suitably large constant c and all n > 0

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

which is at most cn if

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

In the next lecture..



Lecture 6: Heap and Heapsort