CS231 Algorithms
Prof. Lyn Turbak
Wellesley College

Handout # 16
October 23, 2001

Order Statistics

Reading: *CLRS* Ch. 9, *CLR* Ch. 10

Terminology

Let S be a set of n elements (necessarily distinct).

- The ith **order statistic** of S is the ith smallest element (i.e., the element that is larger than exactly $i-1$ other elements in the set). Such an element is said to have **rank** i.
- The **minimum** of S is the first order statistic (element with rank 1).
- The **maximum** of S is the nth order statistic (element with rank n).
- The **median(s)** of S is (are) the element(s) with rank $\lfloor (n+1)/2 \rfloor$ and $\lceil (n+1)/2 \rceil$.

Example

$$B = \{43, 5, 17, 91, 2, 42, 19, 72, 37, 3\}$$

- minimum of $B =$
- maximum of $B =$
- medians of $B =$
- 3rd order statistic of $B =$
- rank of 17 in $B =$

The Selection Problem

List Selection

`select i xs`: Return the rank i element from a list `xs` of n distinct elements.

Array Selection (CLRS/CLR)

`Select(A,i)`: Return the rank i element from an array `A` of n distinct elements.

In this handout, we focus on the list selection problem. Array selection is similar; see *CLRS*/*CLR* for details.

Trivial List Selection Algorithm

```
select i xs = nth i (sort xs)

-- Returns the nth element (1-indexed) of the given list
nth 0 xs = error "nth: index out of range"
nth n [] = error "nth: index out of range"
nth 1 (x:xs) = x
nth n (x:xs) = nth (n-1) xs
```

What is the best worst-case running time for this algorithm?

The burning question: *Can we do better?*

Important Special Cases

- Can find mininum or maximum with $n - 1$ comparisons.

- Can find minimum *and* maximum with $3\lceil n/2 \rceil$ comparisons.

- Can find two smallest elements with $n + \lceil \lg n \rceil - 2$ comparisons. (This is a problem on Problem Set 5).
- For any fixed $k \geq 1$, can select kth or $(n + 1 - k)$th element in $\Theta(n)$ time by k applications of minimum/maximum + deletion.

Partition-Based Selection

Key idea: Use quicksort-like partitioning, but only explore one partition.

```
partitionSelect i [] = error "empty list"
partitionSelect 1 [x] = x
partitionSelect i (x:xs)
  | i <= length(ls) = partitionSelect  i ls
  | i > length(ls) = partitionSelect (i - length(ls)) gs
  where (ls',gs') = partition x xs
        (ls,gs) = if null ls' then ([x],gs') else (ls',x:gs')
```

- What are the best-case inputs for `partitionSelect`? What is the running time for these inputs?

- What are the worst-case inputs for `partitionSelect`? What is the running time for these inputs?

- Sorted or nearly sorted lists are common in practice. What are some practical way to avoid the worst-case running times for these inputs?

- What is average-case running time `partitionSelect`?

Analysis: Partition Selection is Expected Linear Time

What are the possible lengths of the `ls'`?

Assuming that the list argument of `paritionSelect` is a random permutation of distinct elements, what is the probability that `ls'` has any one of the possible lengths?

What are the possible lengths of the `ls`?

What is the probability that `ls` has any one of the possible lengths k, where $k > 1$?

What is the probability that `ls` has length 1? where $k > 1$?

Deriving a recurrence equation:

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

Using substitution method to show $T(n) \leq cn$ is a solution to the above recurrence (for simplicity, assume n is even, but also works for odd case):

$$\begin{aligned} T(n) &\leq \frac{2}{n}\left(\sum_{k=n/2}^{n-1} ck\right) + O(n) \\ &= \frac{2c}{n}\left(\sum_{k=n/2}^{n-1} k\right) + O(n) \\ &= \frac{2c}{n} \cdot \frac{n}{2} \cdot \frac{n/2+n-1}{2} + O(n) \\ &= c \cdot \frac{3n-2}{4} + O(n) \\ &= c \cdot \frac{3n}{4} + O(n) \\ &\leq cn, \text{ for sufficiently large } c. \end{aligned}$$

Note: $T(n) \leq cn$ is *not* a solution to the average-case quicksort recurrence:

$$T(n) = \frac{2}{n}\left(\sum_{k=1}^{n-1} T(k)\right) + O(n).$$

(Try it and see!)

Selection in Worst-Case Linear Time

I find the presentation of this algorithm in *CLRS/CLR* confusing. I prefer the following explanation.

We will consider the **median-of-medians-of-r** algorithm, where r is an integer constant ≥ 1. The following is a sketch of the function `medianOfMedians r i xs`:

1. For simplicity, assume `r` evenly divides $n = \mathit{length}(\texttt{xs})$. If `r` does not evenly divide n, can always create an extended list `xs'` that does by padding `xs` at the end with large elements that won't affect results. Let $c = n/r$, after any such padding.

2. View padded list `xs'` = `[xs'`$_1$`, xs'`$_2$`, ..., xs'`$_n$`]` as a two-dimensional array `B[1..r, 1..c]` with `r` rows and `c` columns:

$$\begin{array}{llcl} \texttt{B[1,1] = xs'}_1 & \texttt{B[1,2] = xs'}_{r+1} & \cdots & \texttt{B[1,c] = xs'}_{n-r+1} \\ \texttt{B[2,1] = xs'}_2 & \texttt{B[2,2] = xs'}_{r+2} & \cdots & \texttt{B[2,c] = xs'}_{n-r+2} \\ \vdots & \vdots & \ddots & \vdots \\ \texttt{B[r,1] = xs'}_r & \texttt{B[r,2] = xs'}_{2r} & \vdots & \texttt{B[r,c] = xs'}_n \end{array}$$

   Sort each column using any method (even a $\Theta(n^2)$ one like insertion sort). Since each column contains `r` elements, this step is linear, not quadratic. After this step, the row `B[`$\lfloor (r+1)/2 \rfloor$`, 1..c]` contains the (lower) medians of the `c` columns. Let `meds` be the list of elements in this row.

3. Use `medianOfMedians r, meds,` $\lfloor (c+1)/2 \rfloor$ to find median of medians `mm`.

4. Partition `xs'` around `mm` to yield the pair of lists `(ls,gs)`, guaranteeing that `mm` ends up in `ls`. Let `k` be the number of elements in `ls` and `(n - k)` the number of elements in `gs`.

5. If `i` $\leq$ `k`, return `medianOfMedians r i ls`; if `i` $>$ `k`, return `medianOfMedians r (i - k) gs`.

Analysis:

- _____ $\leq$ $\boxed{\text{number of elements} \leq \texttt{mm}}$ $<$ _____
- _____ $<$ $\boxed{\text{number of elements} > \texttt{mm}}$ $\leq$ _____
- Recurrence for worst-case running time $T(n)$:

- Solution to recurrence: