Selection

1 Problem

Input: `A[...]` and an integer `i`$\epsilon\{1, ..., n\}$

Output: i^{th} order statistic of `A` (i^{th} smallest element)

2 QuickSelect

The algorithm for randomized selection is an adaptation of the partition subroutine used in quicksort. The high-level idea is to partition a random element from the list, and if it ends up in the proper index, return it, or otherwise recurse on the appropriate half of the array.

```
QuickSelect(A, i):
    if n==1: return A[0]
    p = rand(0, n)
    Partition(A, p)
    j = index_p //After pivot has been moved

    if j==i: return pivot
    if j > i: return RSelect(left, i)
    if j < i: return RSelect(right, i-j)
```

2.1 Analysis

If we let the pivot element be random on every partition but assume that the element is never found until the end of the algorithm *and* the algorithm always recurses on the larger half, we can get a reasonable upper bound for the average case of the algorithm. The recurrence is almost identical to the Quicksort recurrence (one less recursive call), so see the Quicksort notes for a more in-depth explanation.

For the constructive induction, we guess $T(n) \leq an$.

$$\begin{align}
T(n) &= \frac{1}{n}\sum_{q=1}^{n} T(\texttt{max(q-1, n-q)}) \quad + \quad n-1 \tag{1}\\
&= \frac{2}{n}\sum_{q=n/2}^{n-1} T(q) \quad + \quad n-1 \tag{2}\\
&\leq \frac{2}{n}\sum_{q=n/2}^{n-1} aq \quad + \quad n-1 \tag{3}\\
&= \frac{2a}{n}\left(\sum_{q=1}^{n-1} q - \sum_{q=1}^{\frac{n}{2}-1} q\right) + n - 1 \tag{4}\\
&= \left(\frac{3a}{4} + 1\right) n - \frac{a}{2} - 1 \tag{5}
\end{align}$$

For the constructive induction to hold,

$$\begin{align}
\frac{3a}{4} + 1 &\leq a \tag{6}\\
a &\geq 4 \tag{7}
\end{align}$$

This pessimistic analysis yields $T(n) \leq 4n \in \Theta(n)$.

3 Deterministic Selection

Deterministic selection is not as efficient as QuickSelect in practice, and is much trickier to implement. It is still $\Theta(n)$, and is guaranteed to be so for all cases, but in practice has much slower constants than QuickSelect.

```
DSelect(A, n, i):
    C = Choosepivot(A, n)
    p = DSelect(C, n/5, n/10)  //Recursively find median
    Partition(A, p)
    if j == i: return p
    if j < i: return DSelect(left, j-1, i)
    if j > i: return DSelect(right, n-j, i-j)

ChoosePivot(A, n):
    Break A into groups of size 5 each
    Sort each group
    Copy n/5 medians into new array C
    return C
```

3.1 Analysis

The key insight is that when partitioning in the final step, we are guaranteed to get a 30-70 split or better. To explain this, imagine the `ChoosePivot` subroutine as conceptually dividing `A` into a 5-by-$n/5$ matrix.

Each column is sorted in the subroutine so that when the median-of-medians is found (median of the center row of the matrix), we know for sure that the first three elements in all sorted columns with median lower than m are less than m, and that the last three elements in all sorted columns with median high than m are greater than m. This gives $3n/10$ elements that we are unsure about, so we get at worst a 30-70 split.

If we use a quadratic sort like Selection Sort to sort the columns, we can expect 10 comparisons per column. There are $n/5$ columns, so `ChoosePivot` uses $2n$ comparisons. The last partition takes again $n-1$ comparisons, and the recurrence is now fairly straightforward.

$$T(n) \leq T\left(\frac{7}{10}n\right) + T\left(\frac{n}{5}\right) + 3n - 1 \tag{8}$$

Use constructive induction to guess that $T(n) \leq an$:

$$T(n) \leq \frac{an}{5} + \frac{7an}{10} + 3n - 1 \tag{9}$$

$$= \left(\frac{9a}{10} + 3\right) n - 1 \tag{10}$$

For the induction to hold,

$$\frac{9a}{10} + 3 \leq a \tag{11}$$

$$a \geq 30 \tag{12}$$

So we get that the worst-case for deterministic selection by median-of-medians is $T(n) \leq 30n \in \Theta(n)$. This is a significantly worse comparison-based runtime than randomized selection, and worse even than sorting when $n \leq 2^{30}$. Practically, deterministic selection is used over randomized selection when the worst case must still be fast (QuickSelect is $O(n^2)$ in the absolute worst-case).

Note that because the algorithm can terminate early, this is still preferred over an $n \lg n$ sort.