Order Statistics

In this lecture we look at order statistics.
- Select the $i$th smallest of $n$ elements (the element of rank $i$).
  - If $i = 1$ we have the minimum
  - If $i = n$, we have the maximum
  - If $i = \lfloor (n + 1)/2 \rfloor$ or $\lceil (n + 1)/2 \rceil$, then we have the median
- A simple solution is just to sort the input and choose the $i$th element of the sorted array.
- Worst case is $\Theta(n \log n)$ time using Merge Sort or Heap Sort for example.
- We can do better than that!

Partitioning revisited

For Quicksort we used a random partitioning that was able to take a pivot $x$ and rearrange the input array $A$ so that all values $\leq x$ are to the left of $x$ and those $\geq x$ are to right of $x$.

RAND-PARTITION$(A, p, q)$ partitions $A[p, \ldots, q]$ according to a randomly chosen pivot and returns $r$, the index of the pivot.

First we have to choose a random pivot and then call PARTITION with the pivot placed at end of the input array.

Input: $A, p, q$

\[
\begin{align*}
\text{Input: } & A, p, q \\
i & \leftarrow \text{RANDOM}(p, q); \\
\text{swap } & A[q] \text{ and } A[i]; \\
\text{return} & \text{PARTITION}(A, p, q);
\end{align*}
\]

At the start of every iteration, the following invariants are true
- For $p \leq k < i$, $A[k] \leq x$
- For $i + 1 \leq k < j$, $A[k] > x$
- $A[q] = x$
Randomised algorithm

We can use the partition function repeatedly to find the \(i\)th smallest element.

- Pick a pivot at random
- Partition according to the pivot
- Now we only need to look in one of the two “halves” after the partitioning
- Recurse until either we happen to choose the \(i\)th smallest element as a pivot or the half we need to look in only has one element in it

\[
\text{R AND -SELECT}(A, p, q, i) \text{ returns the } i\text{th smallest element of } A[p, \ldots, q]
\]

**Input**: \(A, p, q\) and \(i\)

**Output**: The \(i\)th smallest value in \(A[p, \ldots, q]\)

\[
\begin{align*}
\text{if } p &= q \text{ then } \text{return } A[p]; \\
\text{end} \\
r &\leftarrow \text{R AND -PARTITION}(A, p, q); \\
k &\leftarrow r - p + 1 \quad \triangleright \text{pivot index in } A[p, \ldots, q]; \\
\text{if } i &= k \text{ then } \text{return } A[r]; \\
\text{end} \\
\text{if } i &< k \text{ then } \text{return } \text{R AND -SELECT}(A, p, r - 1, i); \\
\text{else} \\
\text{return } \text{R AND -SELECT}(A, r + 1, q, i - k); \\
\text{end}
\end{align*}
\]

Order Statistics Example

In this example we want to find the 5th smallest number in the array 3, 8, 1, 7, 4, 6, 2.

Running Time - Intuition

In the worst case we might partition the array with one element on one side and all the others on the other side. (We assume all elements are distinct to make life easy for ourselves.)

- \(T(n) = \Theta(n) + T(n - 1)\)
- Therefore \(T(n) = \Theta(n^2)\). Very unlucky and worse than sorting!

If are very lucky we would partition the array exactly in half each time

- \(T(n) = T(n/2) + \Theta(n)\)
- Case 3 of the Master Theorem as \(cn \in \Omega(n^{\log_2 1}) = \Omega(n^0) = \Omega(1)\).
- Therefore \(T(n) = \Theta(n)\). This is very different from the worst case!
- We don’t need to be that lucky to get linear time
  - For example, if \(T(n) = T(99n/100) + \Theta(n)\) then we still have \(T(n) = \Theta(n)\)
  - In fact, the average case is also linear time and the method is very fast in practice.
Running Time - Better Intuition

We can be slightly more mathematical about our intuition.

- Call a *good* partition one where the largest side in the partition is of size $\leq 3n/4$.
- Fifty percent of the pivot choices will give us this.
- On average, we need only two pivot choices to find such a partition. This is the same as the expected number of times you need to toss a coin to get a head.
- Therefore the expected running time is $T(n) \leq T(3n/4) + O(n)$.

This gives us $T(n) \in \mathcal{O}(n)$ on average.

Worst case linear time analysis

1. $\Theta(n)$ time to break into groups
2. $\Theta(n)$ time find median of each group
3. $T(\lfloor n/5 \rfloor)$ time to find median of group medians
4. $\Theta(n)$ time to partition
5. Time to recursively call SELECT on one “half” of the input

- There are $\lfloor n/5 \rfloor$ group medians, at least half of which are $\leq x$, the pivot by definition. This is at least $\lfloor \lfloor n/5 \rfloor/2 \rfloor = \lfloor n/10 \rfloor$ group medians.
- Therefore at least $3 \lfloor n/10 \rfloor$ elements are $\leq x$ in total
- Similarly at least $3 \lfloor n/10 \rfloor$ elements are $\geq x$ in total

Worst case linear time order statistics

The problem with the randomised approach is that we might choose bad pivots. Blum, Floyd, Pratt, Rivest, and Tarjan [1973] found a way to guarantee a good pivot each time. $\textsc{Select}(A, p, q, i)$

1. Divide the $n = q - p + 1$ elements into groups of 5. Find the median of each 5-element group.
2. Recursively $\textsc{Select}$ the median $x$ of the $\lfloor n/5 \rfloor$ group medians to be the pivot.
3. Partition $A[p, \ldots, q]$ using the pivot $x$. $k = \text{rank}(x)$
4. If $i = k$ then return $x$
5. Else if $i < k$

   - then recursively $\textsc{Select}$ the $i$th smallest element in the lower part
   - else recursively $\textsc{Select}$ the $i - k$th smallest element in the higher part

We simplify by noticing that for $n \geq 20$, $3\lfloor n/10 \rfloor \geq n/4$. For $n < 20$ we can find the order statistics in constant time.

- Therefore the final recursive call is on an array of no more than $3n/4$ elements.
- This takes at most $T(3n/4)$ time
- Therefore $T(n) \leq T(3n/4) + T(n/5) + \Theta(n)$
- Guess $T(n) \in \Theta(n)$ and substitute $T(n) = cn$ giving $T(n) \leq 3cn/4 + cn/5 + \Theta(n)$
- $T(n) = 19/20cn + \Theta(n)$
- $T(n) = cn - (cn/20 - \Theta(n))$
- $T(n) \leq cn$, if we choose $c$ to be large enough to compensate for the $\Theta(n)$ term
Summary

- **RAND-SELECT** is linear time on average and simple to implement but harder to analyse.
- However, **RAND-SELECT** runs in $\Theta(n^2)$ in the worst case.
- **SELECT** runs in linear time in the worst case but is slower in practice unless you are very unlucky.
- **SELECT** is slightly harder to implement but simpler to analyse.

Further Reading

- **Introduction to Algorithms**
  - Chapter 9 – Order statistics
- **Algorithms**
  S. Dasgupta, C.H. Papadimitriou and U.V. Vazirani
  http://www.cse.ucsd.edu/users/dasgupta/mcgrawhill/
  - Chapter 2, Section 2.4 – Medians