Divide and Conquer: Order Statistics

Lecture 6

Akshar Varma

13th July, 2023

CS3000 Algorithms and Data

Order Statistics

Good and Bad Pivots

Median-of-medians: MoMs

Quicksort: straightforward, MoMs, [Optional: randomized]

Two Data Structures: Priority Queues and Binary Search Trees

1. Order Statistics

"Chaos is a ladder."
–G. R. R. M.

Order Statistics

- Order statistics, as the name implies are statistical values that characterize order.
- You've heard of minimum and maximum; special cases of order statistics.
- The k^{th} order statistic of a collection is the k^{th} smallest value in it.
- Minimum is $k = 1$, maximum is $k = n$ (where n is the size of the collection).
- You may also have heard of other special order statistics:
    - Quartile: $0.25n, 0.5n, 0.75n$
    - Decile: $0.1n, 0.2n, 0.3n, \ldots, 0.8n, 0.9n$
    - Percentile: $0.1n, 0.33n, 0.68n, 0.95n, 0.99n$, etc.
    - Median: $\lfloor \frac{n}{2} \rfloor$
- This has extensive applications in statistics and inference.
- We'll see the $1^{\text{st}}/n^{\text{th}}$ order statistic (minimum/maximum) show up a lot in the future.

Finding order statistics: k-Selection

- $k = 1$ (minimum) and $k = n$ (maximum) can both be found in $\Theta(n)$ time.
- Note: k-selection is $\Omega(n)$ since we must look at all the data.

- Using $\Theta(k)$ space, we can do k-selection in $\Theta(n)$ time.
- Without space usage, we can find minimums repeatedly and finish in $\Theta(kn)$ time.

- All of the above assumed unordered collection/list/array.
- If sorted, then k-selection is doable in $\Theta(1)$ for any value of k (just index into array).

- Spending $O(n \log n)$ on sorting will allow k-selection in $\Theta(1)$.
- Can we do selection without sorting and in $\Theta(n)$ for all k?

Pivots in k-selection

- Pick a pivot element p. This can be any element, for example, the first one.
- Split array into everything $< p$ (*Left*), everything $= p$ (*Middle*), everything $> p$ (*Right*).
- This is $\Theta(n)$.
- Let ℓ, m, r be the sizes of *Left*, *Middle* and *Right*.
- If $k \leq \ell$ recurse in *Left*.
- If $\ell < k \leq \ell + m$, elements in the *Middle*, that is, p is the k^{th} order statistic.
- Otherwise, recurse on *Right*, to find k'^{th} order statistic where $k' = k - \ell - m$.

- Runtime: $T(n) = T(\max(\ell, r)) + \Theta(n)$.

2. Good and Bad Pivots

“Pivoting isn’t Plan B; it is part of the process.”

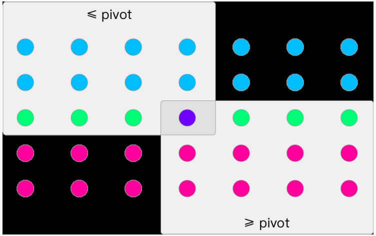
Good and Bad Pivots

- ℓ and r depend on how large p vs. other elements, that is, its order.
- If it is too small or too large, then $\max(\ell, r)$ may be $\Omega(n)$.
- In that case, the recursion does not give a $\Theta(n)$ runtime.
- So we need to ensure that p is close to the median.
- Pivoting needs $\Omega(n)$ time irrespective of p.
- So we only have $O(n)$ time to find a pivot p close to the median.

3. Median-of-medians: MoMs

"There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors." – Leon Bambrick

Median of Medians Algorithm

- We don't need the exact median, only something close.
- Sort every group of 5 elements (sorting is constant time), $\Theta(n)$ such sortings.
- Recurse with these $\lfloor \frac{n}{5} \rfloor$ medians. When ≤ 5 elements return true median.
- Once a median-ish pivot is found, continue with your k-selection.
- 

Median of Medians guarantee

- k-selection recurrence becomes $T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + \Theta(n) \implies T(n) = \Theta(n)$.

4. Quicksort: straightforward, MoMs, [Optional: randomized]

"There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors." – Leon Bambrick

Quicksort

- Pivoting also suggests a sorting algorithm.
- Perform pivoting, recurse on left and right halves.
- Same problems as before wrt "good pivot". Worst-case complexity is $\Theta(n^2)$ (exercise).
- Resort to MoM to get a good pivot. Show that in this case runtime is $\Theta(n \log n)$.
- Alternative: Randomness!

  Pick a random element as the pivot. Chance that it is bad is low.
- We won't do a proof, but this also gives a runtime of $\Theta(n \log n)$.
- In practice, due to large constants in MoM, randomized Quicksort is commonly used.

5. Two Data Structures: Priority Queues and Binary Search Trees

Two Data Structures: Priority Queues and Binary Search Trees

1. Priority Queues:
   - Keeps track of things in order of "priority".
   - $\Theta(1)$ access to the min or max priority element.
   - Creation of data structure: $O(n)$
   - Deletion $O(\log n)$
2. (Balanced) Binary Search Trees:
   - Keeps track of things in a binary tree with the following property: Left subtree has smaller values and right subtree has larger values.
   - Insertion, Deletion, Search, etc. are all $O(\log n)$ operations.
   - Many other things can be done.
   - Inorder, preorder, postorder traversals.
   - Depth and height of nodes.
   - Nodes can track subtree size to allow indexing and order statistics.