CPSC 320 Notes: Deterministic Select

February 23, 2015

Last time we designed the `QuickSelect` algorithm to find the "k-th order statistic" (k-th smallest element) of a list of numbers in average-case linear time. This time, we'll explore whether we can do the same in **worst-case** linear time.

Reminder: For $0 < r < 1$, $\sum_{i=0}^{\infty} r^i = 1 + r + r^2 + r^3 + \ldots = \frac{1}{1-r}$ (i.e., a constant if r is a constant).

1 Using Recurrences to Explore What We Can Do

`QuickSelect` takes linear time at each node and "divides" the problem into a single subproblem of an unpredictable size, although analysing a $\frac{3}{4}$ size breakdown turns out to work for average case.

1. Draw the recursion tree for a recurrence like $T(n) = T(\frac{3}{4}n) + cn$ (with a reasonable base case) and sketch out the key pieces of its analysis: problem sizes, work per node, work per level, and number of levels. (For space: Use a separate sheet of paper or rotate this paper sideways!)

2. Alter your tree for a recurrence like $T(n) = T(a \cdot n) + cn$, where we don't know the breakdown factor a. What should be true about a so that our algorithm runs in linear time?

3. Now, draw a tree for the recurrence $T(n) = T(n - 1000) + cn$. (From now on, label all key pieces in your tree without our asking!) What performance does this get?

4. Let's see if we have a bit more freedom. Recall that `QuickSort` got $O(n \lg n)$ performance even though it had "unbalanced" branches. Draw a tree for a recurrence like $T(n) = T(\frac{2}{3}n) + T(\frac{1}{4}n) + cn$. (This one will take more space!)

5. Upper- and lower-bound the amount of work in this tree. (Remember how we analysed the "shortest" and "longest" branches for `QuickSort`?)

6. If we changed $\frac{2}{3}$ and $\frac{1}{4}$ to a and b, what would we need to know about a and b to ensure we got a good performance bound?

2 Designing Deterministic Select

OK; we can afford two recursive calls under certain conditions. One of them could be a `QuickSelect`-like recursive call on a problem perhaps $\frac{3}{4}$ as large. The other can help us find a quality pivot.

1. Consider the following completely unrelated diagram that is 49 characters wide, where an element labeled `<` is known to be less than the one labeled `*`, a `>` is known to be greater, and a `.` has an unknown relationship. If you were looking for the $103rd$ smallest element, circle the elements you **might** have to search through for that element.

```
<<<<<<<<<<<<<<<<<<<<<<<<<........................
<<<<<<<<<<<<<<<<<<<<<<<<<........................
<<<<<<<<<<<<<<<<<<<<<<<<*>>>>>>>>>>>>>>>>>>>>>>>>
........................>>>>>>>>>>>>>>>>>>>>>>>>>
........................>>>>>>>>>>>>>>>>>>>>>>>>>
```

2. Looking **only** at the middle row, describe the relationship of the element `*` to the other elements (in as few words as possible!).

3. Looking only at the middle column, describe the relationship of the `*` element to the other elements (in as few words as possible!).

4. Looking only at some column on the left side, and assuming we know that the middle element e is less than the `*`, what more do we need to know to know about the relationship between e and the top two elements in its column to know that those top two elements are also less than the `*`?

5. How long does it take to find the median of 5 elements?

6. Using $T(\cdot)$, how long does it take to find the median of one row of these elements?

7. Design the `DeterministicSelect` algorithm that finds the k-th order statistic of a list of n numbers in linear time.

8. Give a recurrence for `DeterministicSelect` and briefly justify based on the work we've already done why it takes linear time.

9. Pat yourself on the back for designing this strange algorithm. (In practice `DeterministicSelect`'s constant factors are too large for common use, but `QuickSelect` deeply rocks.)

3 Challenge

1. `RandomizedQuickSelect` works like `QuickSelect` except that it chooses a random pivot, not the first element. A **crucial** piece of the analysis of `RandomizedQuickSelect` is "linearity of expectations": the expected value of a sum of random variables is the sum of the expected values of those random variables. Basically, if you can break a quantity you want to know about (like the number of levels in `RandomizedQuickSelect`'s recursion tree) into a sum of variables (like the number of levels in various distinct "stages" of the tree), then you can find expectation values for the "smaller" variables and add them up.

   Use this to make our "$\frac{3}{4}$"-style analysis more rigorous for `RandomizedQuickSelect`'s expected-case running time.

2. Give a way to find the median of 5 elements using no more than 9 comparisons. Can you do it in 6?

3. Can we use a smaller or larger number than `5` for `DeterministicSelect`?

4. Find other meaningful ways to alter the recurrence $T(n) = T(a \cdot n) + T(b \cdot n) + cn$ while preserving its $O(n)$ bound.