Using Methodology we cross the boundary Clever/Miraculous Algorithms
CSE 202 Lecture 5

Lecture given by Russell Impagliazzo, scribed by Sashka Davis

October 6, 2005

Abstract

In this lecture we see another example of using the methodology to design clever algorithms. We use the same framework as before: 1. Discover an algorithm; 2. Abstract away the specific details; 3. Find an optimal algorithm among the class of algorithms defined above.

The particular problem we consider is the **Rank selection problem**. The difference is that our specific algorithm is a *probabilistic* algorithm for rank selection running in worst case expected time[1] $O(n)$. The miracle is that applying the methodology above delivers a deterministic algorithm running in time $O(n)$!

1 Probabilistic Algorithm for Rank Selection Problem

The rank selection problem is defined as follows. An instance is an array of numbers $A[1..n]$ and an integer $I \in [n]$. The problem is to find the I-th largest element of A. We begin with a randomized rank selection algorithm. The idea of the probabilistic algorithm is:

Algorithm $PRank(A, I)$

1. Pick $J \in_u [n]$, Let j be the rank of $A[J]$.
2. Separate the elements of A onto two sets B and S. B are all elements of A bigger than $A[J]$, and $S = A - B$.
3. If $I < |B|$ then $PRank(B, I)$ else $PRank(A, I - |B|)$

Analysis of PRank:
Call j good if $j \in [\frac{n}{4}, \frac{3n}{4}]$. If j is good then $|B|, |S| \leq \frac{3n}{4}$. With probability 1/2 j is good and then the size of the instance in the recursive call is smaller than $\frac{3n}{4}$. When j is not good, then the size of the instance is at most $n - 1$, but the main point is that it cannot possibly get bigger. This gives rise to the following recurrence expressing the expected running time of the probabilistic algorithm:

$$ET(n) \leq \frac{1}{2}(ET(\frac{3n}{4}) + ET(n)) + O(n)$$

Now we find closed form, to keep ourselves honest in the recurrence relation we write cn instead $O(n)$.

$$ET(n) \leq \frac{1}{2}(ET(\frac{3n}{4}) + ET(n)) + cn$$

[1]Worst-case expected time is defined as $\max_{A,I}\ \mathrm{avg}_{\mathrm{r}}\{T_r(A, I)\}$, where $T_r(A, I)$ is the running time of the algorithm on input A, I, using random bits r.

$$\frac{1}{2}ET(n) \leq \frac{1}{2}ET(\frac{3n}{4}) + cn$$

$$ET(n) \leq ET(\frac{3n}{4}) + 2cn$$

This is a top heavy case with solution $O(n)$.

2 Deterministic Median Selection Algorithm[1]

The worst case of Rank selection problem is when we want to find the median, i.e. $I = n/2$. This is the problem we consider here. We want to use the idea of the probabilistic algorithm above: choose a (good) pivot point, divide and conquer. The important part of the randomized algorithm was that, on average, over the random choices of the algorithm, we reduced the size of the instance by 1/4. We want to do the same: divide so that we reduce the size of the instance by 1/4. But how do we select a good pivot point deterministically? We want to be clever and will examine a class of algorithms parameterized by the number of partitions (as before the parameter we use is the number of partitions in the divide part).

2.1 Algorithm

1. Divide the instance: Partition the instance into b parts each of size $\frac{n}{b}$.
2. Find the medians of each of the partition recursively. Say the medians of these b partitions are $M_1, M_2, ..., M_b$ respectively.
3. Find the median M of the medians $M_1, ..., M_b$ recursively.

2.2 Analysis

How good of a median is the median M?

- M is smaller than half of the medians, which in terms are smaller than half of the elements in their respective groups. So $M \leq \lceil \frac{b}{2} \rceil$of the medians $\leq \approx \lceil \frac{k}{2} \rceil \times \lceil \frac{n}{2b} \rceil = \lceil \frac{n}{4} \rceil$ elements.
- Similarly $M \geq \lceil \frac{b}{2} \rceil$of the medians $\geq \lceil \frac{n}{2b} \rceil \approx \lceil \frac{n}{4} \rceil$ elements.

This leads to the following recurrence:

$$T(n) \leq bT(\frac{n}{b}) + T(b) + T(\frac{3b}{4} + O(n)$$

2.3 Optimize

Note that the possible values of b gives rise to a class of algorithms, with different running times. Within this class [1] found the best value of b to be $\frac{n}{5}$. Although any odd integer bigger than 3 in the place of 5 would also deliver a linear time algorithm, the hidden constant in the $O(n)$ is best when 5 is used and the actual recurrence is

$$T(n) \leq \frac{n}{5}T(5) + T(\frac{n}{5}) + T(\frac{3n}{4}) + O(n).$$

Finding medians in groups of constant size is done in constant time by sorting the group, hence

$$T(n) \leq T(\frac{n}{5}) + T(\frac{3n}{4}) + O(n).$$

The solution is $O(n)$, because the combined size of the partition is smaller than 1 (You could unwind the recurrence, and solve it. Pictorially - the tree you get is not balanced, but the depth is logarithmic in n and the amount of work at each consequitive level is smaller than the previous, hence the linear time).

3 Conclusion

Why is this algorithm miraculous? The reasons are many.

First, note that the specific algorithm that they started with is of different kind. They started with a randomized algorithm and the use of randomenss was essential to achieve the expected worste case linear time. The fact that a probabilistic algorithm can be derandomized without loss of efficiency is a non-trivial matter.

Second, the algorithm they designed is optimal. You cannot do better than than. It is easy to show a lower bound of $O(n)$ of any deterministic algorithm: the intuition is that you can't find the median of n number, within the comparison model, unless you read the whole input.

Third, is somehow related to the first reason, but has a philosophical implication: BPP=P? The authors of [1] were trying to show that randomized polynomial time algorithms are privileged, and the use of randomness, makes them more powerful than deterministic polynomial time algorithms. But they failed and instead, they found a deterministic algorithm just as good as the probabilistic one. So their algorithm is one of the many algorithms for problems which were found to have algorithms in the classes BPP or RP and later fall in P. Strengthening the evidence/belief that BPP=P (one after another the natural problems with algorithms in RP or BPP fall in P).

References

[1] Manuel Blum, Robert Floyd, Vaghan Pratt, Ronald Rivest, Robert Tarjan *Linear Time Bounds for Median Computations*, Journal of ACM, 1971.