Randomized algorithms for the majority problem

Demetres Christofides [1]

School of Mathematics
University of Birmingham
Birmingham, United Kingdom

Abstract

In the majority problem, we are given n balls coloured black or white and we are allowed to query whether two balls have the same colour or not. The goal is to find a ball of majority colour in the minimum number of queries. The answer is known to be $n - B(n)$, where $B(n)$ is the number of 1's in the binary representation of n. In [5], De Marco and Pelc proved that even if we use a randomized algorithm which is allowed to fail with probability at most ε, we still need linear expected time to determine a ball in majority colour. We prove that any such algorithm has expected running time at least $\left(\frac{2}{3} - \delta(\varepsilon)\right) n$, where $\delta(\varepsilon) \to 0$ as $\varepsilon \to 0$. Moreover, we provide a randomized algorithm showing that this result is best possible.

Keywords: Randomized algorithms, majority problem.

1 Introduction

In the 'majority problem', we are given n balls coloured black or white. At any stage we are allowed to select two balls and ask whether they have the

[1] Email: christod@maths.bham.ac.uk

same colour or not. Our task is to find a ball of majority colour, or decide that no such ball exists. How many questions do we need to ask, in the worst case? Clearly $n-1$ questions suffice. For example, we can compare the first ball with all the rest. The following recursive algorithm does slightly better: If n is odd, it is enough to determine majority in the first $n-1$ balls. Indeed, a ball which is in majority colour when restricted to the first $n-1$ balls, is also in majority in the totality of n balls. On the other hand if no majority exists in the first $n-1$ balls, then the n-th ball is in majority. If n is even, we can pair the balls arbitrarily, make $n/2$ comparisons, throw out all pairs for which the colours were different, and keep one ball from each pair for which the colours were the same. Then, clearly, it is enough to determine majority in the balls left. An easy inductive argument now shows that this algorithm determines majority in at most $n-B(n)$ questions, where $B(n)$ is the number of 1's in the binary representation of n. Saks and Werman [7] showed that in the worst case we do need that many questions. We refer the reader to [1,2] for surveys on the majority problem and some of its variants.

What happens if we allow some randomization in our algorithm for determining majority? To be more precise, at each step we are allowed to pick the two balls to be compared using some probability distribution which is allowed to depend on our current knowledge so far. We allow our algorithm to fail with probability at most ε. (I.e. given any input, the randomized algorithm must produce a correct answer with probability at least $1-\varepsilon$.) To the best of our knowledge, this was first studied by De Marco and Pelc [5]. They showed that randomization does not improve the running time by much, in the sense that there are inputs for which the expected running time of any randomized algorithm is linear. Although it does not appear explicitly in their work, their proof shows that for any randomized algorithm which fails with probability at most ε, there is an input on which the algorithm has expected running time is at least $n/40$. (Provided ε is sufficiently small.) We already know that one can solve the majority problem in n steps (even without randomization) so it is natural to ask what the right constant for the speed of the running time (in the worst case) is. This question is answered by the following two theorems from [4].

Theorem 1.1 *For every $\varepsilon > 0$ if n large enough, then there is a randomized algorithm for determining majority on n balls which fails with probability at most ε and has expected running time at most $\frac{2}{3}\left(1-\frac{\varepsilon}{3}\right)n$.*

Theorem 1.2 *For every $\delta > 0$, there exists an $\varepsilon = \varepsilon(\delta) > 0$ such that, for every large enough n, any randomized algorithm for determining majority on*

n balls with expected running time less that $\left(\frac{2}{3} - \delta\right) n$, fails with probability at least ε on some input.

2 Sketch proof of Theorem 1.1

As De Marco and Pelc observed, if we know that the difference between the number of white and black balls is large, then there is a randomized algorithm with small expected running time for determining majority (with small error probability). The proof of the next lemma is a simple application of the Chernoff bound.

Lemma 2.1 *Suppose that the difference between the number of black and white balls is at least $2\alpha n$. Then there is a randomized algorithm for determining majority which fails with probability at most ε and has expected running time at most $\frac{1}{2\alpha^2} \log\left(\frac{1}{\varepsilon}\right)$.*

On the other hand, the next lemma shows that if we know that the difference is small then we need expected time about $2n/3$. To see why such a result might be true, observe that if we have roughly equal number of white and black balls we can initially partition them into $n/2$ pairs. After making $n/2$ comparisons, about $n/4$ of them will result in the the balls having the same colour which we can discard. So after $n/2$ comparisons we are left with about $n/4$ balls in which we need to determine majority and a recursive argument gives that we need a total of about $2n/3$ comparisons.

Lemma 2.2 *Suppose that the difference, in absolute value, between the number of white and black balls is at most $d = 2\alpha n$. Then there is a randomized algorithm which determines majority with no error whose expected running time is at most $\frac{2n+d}{3}$.*

The final ingredient in the proof is that after a small sampling we can determine with very small error probability whether the difference in the number of white and black balls is large or small.

3 Sketch proof of Theorem 1.2

Observe that a randomized algorithm is nothing else than a probability distribution on the set of all deterministic algorithms. Thus, it seems reasonable to expect that a good understanding of the behaviour of every deterministic algorithm, will yield a good understanding on the behaviour of randomized algorithms as well.

Let A be a deterministic algorithm and suppose we run this algorithm when the balls are coloured uniformly and independently at random. Its running time T becomes a random variable. We will show that if A fails with probability at most ε then the expected running time $\mathbb{E}(T)$ is large.

Theorem 3.1 *Let A be any deterministic algorithm which fails with probability at most ε when the balls are coloured independently and uniformly at random. Then, provided n is large enough, we have that $\mathbb{E}(T) \geqslant \frac{2}{3}(1-\gamma)n$, where $\gamma = \gamma(\varepsilon) \to 0$ as $\varepsilon \to 0$.*

Having proved this, a simple averaging argument will yield the required result. (The observation that in order to find lower bounds on the running time of randomized algorithms it is enough to find lower bounds on the running time of each deterministic algorithm is due to Yao [8].)

Our knowledge after each step of a deterministic algorithm can be described by a graph G, on vertex set $[n]$, where i is joined to j if and only if we have already compared ball i to ball j. The edges of G are labelled with a YES or a NO, depending on the answer we have obtained. Within each component of G, we have enough information to determine whether two balls have the same colour or not. Let M_i be the difference, in absolute value, between the number of black and white balls in component i. We can ignore the components where the difference is 0, and order the other components so that $M_1 \geqslant M_2 \geqslant \ldots \geqslant M_C$. So, regarding the majority problem, the vector $(M_1, \ldots, M_C)$ contains all the information that we are interested in.

The next result from [3] shows that instead of proving that any such algorithm has large expected running time, it is enough to prove that the expected number of components $\mathbb{E}(C)$ in which the difference is not 0, is not too large. In [4] we gave a short inductive proof of this lemma but here we omit it due to lack of space.

Lemma 3.2 *Let A be as above and suppose that the balls are coloured independently and uniformly at random. Then $\frac{3}{2}\mathbb{E}(T) + \mathbb{E}(C) \geqslant n$.*

Suppose that when A announces a ball of majority colour, there are $C \geqslant \sqrt{\varepsilon}n$ components left, of sizes $M = M_1 \geqslant M_2 \geqslant \ldots \geqslant M_C \geqslant 1$, with $M \leqslant \alpha\sqrt{C}$, for some α to be determined later. One can show that the probability that the announced ball is not in the majority is at least

$$\Pr(M < \varepsilon_2 M_2 + \ldots + \varepsilon_C M_C) = \frac{1}{2}\Pr(\varepsilon_2 M_2 + \ldots + \varepsilon_C M_C \notin [-M, M]),$$

where the ε_i take the values ± 1 uniformly and independently at random. It seems impossible to determine this probability, but we are only interested in

a lower bound. So when is this probability minimized? Intuitively, this probability is as small as possible when $M_i = 1$ for each $i \geqslant 2$. This is indeed the case and it follows easily from the solution of Erdős to the Littlewood-Offord problem [6]. But by the normal approximation to the binomial distribution we can deduce that if n is large enough then this probability is at least $\Phi(-2\alpha) - \sqrt{\varepsilon}$, where Φ is the distribution function of the standard normal distribution. Thus, provided α is small enough (but independent of n), the probability of error is at least $\sqrt{\varepsilon}$.

Since A fails with probability at most ε, it follows that with probability at least $1 - \sqrt{\varepsilon}$, either $C \leqslant \sqrt{\varepsilon}n$, or $C \leqslant M^2/\alpha$.

To complete the proof that $\mathbb{E}(C)$ is not too large, it remains to show that $\mathbb{E}(M^2)$ is not too large. However, it is not too difficult to show that $\mathbb{E}(M^2) \leqslant n$. (The crucial observation here is that $\mathbb{E}(M^2)$ increases after each step of the algorithm.) This completes the proof of the theorem.

References

[1] M. Aigner, Variants of the majority problem, Discrete Appl. Math. **137** (2004), 3–25.

[2] M. Aigner, Two colors and more, in Entropy, Search, Complexity, 9–26, Springer, Berlin 2007.

[3] L. Alonso, E. M. Reingold and R. Schott, The average-case complexity of determining the majority, SIAM J. Comput. **26** (1997), 1–14.

[4] D. Christofides, On randomized algorithms for the majority problem, Discrete Appl. Math., to appear.

[5] G. De Marco and A. Pelc, Randomized algorithms for determining the majority on graphs, Combin. Probab. Comput. **15** (2006), 823–834.

[6] P. Erdős, On a lemma of Littlewood and Offord, Bull. Amer. Math. Soc. **51** (1945), 898–902.

[7] M. E. Saks and M. Werman, On computing majority by comparisons, Combinatorica **11** (1991), 383–387.

[8] A. C. C. Yao, Probabilistic computations: toward a unified measure of complexity (extended abstract), in *18th Annual Symposium on Foundations of Computer Science (Providence, R.I., 1977)*, 222–227, IEEE Comput. Sci., Long Beach, Calif.