

# Randomized algorithms for finding a majority element

Paweł Gawrychowski<sup>1</sup>, Jukka Suomela<sup>2</sup>, and Przemysław Uznański<sup>3</sup>

<sup>1</sup>Institute of Informatics, University of Warsaw, Poland

<sup>2</sup>Helsinki Institute for Information Technology HIIT, Department of Computer Science,  
Aalto University, Finland

<sup>3</sup>Department of Computer Science, ETH Zürich, Switzerland

## Abstract

Given  $n$  colored balls, we want to detect if more than  $\lfloor n/2 \rfloor$  of them have the same color, and if so find one ball with such majority color. We are only allowed to choose two balls and compare their colors, and the goal is to minimize the total number of such operations. A well-known exercise is to show how to find such a ball with only  $2n$  comparisons while using only a logarithmic number of bits for bookkeeping. The resulting algorithm is called the Boyer–Moore majority vote algorithm. It is known that any deterministic method needs  $\lceil 3n/2 \rceil - 2$  comparisons in the worst case, and this is tight. However, it is not clear what is the required number of comparisons if we allow randomization. We construct a randomized algorithm which always correctly finds a ball of the majority color (or detects that there is none) using, with high probability, only  $7n/6 + o(n)$  comparisons. We also prove that the expected number of comparisons used by any such randomized method is at least  $1.038n$ .

## 1 Introduction

A classic exercise in undergraduate algorithms courses is to construct a linear-time constant-space algorithm for finding the majority in a sequence of  $n$  numbers  $a_1, a_2, \dots, a_n$ , that is, a number  $x$  such that more than  $\lfloor n/2 \rfloor$  numbers  $a_i$  are equal to  $x$ , or detect that there is no such  $x$ . The solution is to sweep the sequence from left to right while maintaining a candidate and a counter. Whenever the next number is the same as the candidate, we increase the counter; otherwise we decrease the counter and, if it drops down to zero, set the candidate to be the next number. It is not difficult to see that if the majority exists, then it is equal to the candidate after the whole sweep, therefore we only need to count how many times the candidate occurs in the sequence. This simple yet beautiful solution was first discovered by Boyer and Moore in 1980; see [3] for the history of the problem. Even though this might seem like a toy problem, it has some serious motivation for voting in hardware systems.

The only operation on the input numbers used by the Boyer–Moore algorithm is testing two numbers for equality, and furthermore at most  $2n$  such checks are ever being made. This suggests that the natural way to think about the algorithm is that the input consists of  $n$  colored balls and the only possible operation is comparing the colors of any two balls. Now the obvious question is how many such comparisons are necessary and sufficient in the worst possible case. Fischer and Salzberg [10] proved that the answer is  $\lceil 3n/2 \rceil - 2$ . Their algorithm is a clever modification of the original Boyer–Moore algorithm that reuses the results of some previously made comparisons during the verification phase. They also show that no better solution exists by an adversary-based argument. However, this argument assumes that the strategy is deterministic, so the next step is to allow randomization.

Surprisingly, not much seems to be known about randomized algorithms for computing the majority in the general case. For the special case of only two colors, Christofides [4] gives a randomized algorithm that uses  $\frac{2}{3}(1 - \frac{\epsilon}{3})n$  comparisons in expectation and returns the correct answer with probability  $1 - \epsilon$ , and he

also proves that this is essentially tight; this improves on a previous lower bound of  $\Omega(n)$  by De Marco and Pelc [11]. Note that in the two-color case any deterministic algorithm needs precisely  $n - B(n)$  comparisons, where  $B(n)$  is the number of 1s in the binary expansion of  $n$ , and this is tight [1, 13, 14]. For a random input, with each ball declared to be red or blue uniformly at random, roughly  $2n/3$  comparisons are sufficient and necessary in expectation to find the majority color [2].

However, much less is known about randomized algorithms for the general case of multiple colors. Chung et al. [5] studies *oblivious* algorithms, that is, algorithms in which subsequent comparisons do not depend on the previous answers, but to our knowledge upper and lower bounds on the expected number of comparisons for *adaptive* algorithms without any restrictions on the number of colors have not been studied before.

While it might seem that minimizing the number of comparisons is a purely academic exercise, it is feasible that a single comparison is so expensive that their number is the bottleneck in the algorithm. Such line of thought motivated a large body of work studying the related questions of the smallest number of comparisons required to find the median element; see [7, 8, 12] and the references therein. Given the original motivation in hardware systems, it is natural to consider Las Vegas algorithms, that is, the number of comparisons depends on the random choices of the algorithm but the answer is always correct. This way the result will be correct even if the source of random bits is compromised; an adversary that is able to control the random number generator can only influence the running time.

**Model.** We identify balls with numbers  $1, 2, \dots, n$ . We write  $\text{cmp}(i, j)$  for the result of comparing the colors of balls  $i$  and  $j$  (true for equality, false for inequality). We consider randomized algorithm that, after performing a number of such comparisons, either finds a ball of the majority color or detects that there is no such color. A majority color is a color with the property that more than  $\lfloor n/2 \rfloor$  balls are of such color. The algorithm should always be correct, irrespectively of the random choices made during the execution. However, the colors of the balls are assumed to be fixed in advance, and therefore the number of comparisons is a random variable. We are interested in minimizing its expectation.

**Contributions.** We construct a randomized algorithm, which always correctly determines a ball of the majority color or detects that there is none, using  $7n/6 + o(n)$  comparisons with high probability (in particular, in expectation). We also show that the expected number of comparisons used by any such algorithm must be at least  $1.038n$ . Therefore, randomization allows us to circumvent the lower bound of Fischer and Salzberg and construct a substantially better algorithm.

## 2 Preliminaries

We denote the set of balls (items) by  $M = \{1, 2, \dots, n\}$ . We write  $\text{color}(x)$  for the color of ball  $x$ , and  $\text{cmp}(x, y)$  returns true if the colors of balls  $x$  and  $y$  are identical.

An event occurs with very high probability (w.v.h.p.) if it happens with probability at least  $1 - \exp(-\Omega(\log^2 n))$ . Observe that the intersection of polynomially many very high probability events also happens with very high probability.

**Lemma 2.1** (symmetric Chernoff bound). *The number of successes for  $n$  independent coin flips is w.v.h.p. at most  $\frac{n}{2} + \mathcal{O}(\sqrt{n} \log n)$ .*

**Lemma 2.2** (sampling). *Let  $X \subseteq M$  such that  $|X| = m$ . Let  $m'$  denote the number of hits on elements from  $X$  if we sample uniformly at random  $k \leq n$  elements from  $M$  without replacement. Then w.v.h.p.  $|m'/k - m/n| = \mathcal{O}(k^{-1/2} \log n)$ .*

*Proof.* Let  $\delta = \Theta(k^{-1/2} \log n)$ : From the tail bound for hypergeometric distribution (see Chvátal [6]), we have

$$\Pr\left(\left|\frac{m'}{k} - \frac{m}{n}\right| \geq \delta\right) \leq 2 \exp(-2\delta^2 k) = \exp(-\Theta(\log^2 n)).$$

Worth noting is that the error bound here would be essentially the same if we replace “without replacement” with “with replacement”. In that case, we would be invoking a tail bound for binomial distribution.  $\square$

Now we consider a process of pairing the items without replacement (choosing a random perfect matching on  $M$ ; if  $n$  is odd then one item remains unpaired). For any  $X \subseteq M$ , let  $u_{XX}$  be a random variable counting the pairs with both elements belonging to  $X$  when choosing uniformly at random  $\frac{n}{2}$  pairs of elements from  $M$  without replacement. Of course  $E[u_{XX}] = \frac{|X|(|X|-1)}{2(n-1)}$ .

**Lemma 2.3** (concentration for pairs). *For any  $X \subseteq M$  w.v.h.p.*

$$\left| u_{XX} - \frac{|X|^2}{2n} \right| = \mathcal{O}(\sqrt{|X|} \log n).$$

*Proof.* For simplicity, we can assume that  $n$  is even. Set  $\Delta = \Theta(\sqrt{|X|} \log n)$ . Instead of choosing a random perfect matching on  $M$ , we consider an equivalent two-step random process:

1. We choose uniformly at random a partition of  $M$  into  $M_1, M_2$  such that  $|M_1| = |M_2| = \frac{n}{2}$ . This partition induces a partition of  $X$  into  $X_1, X_2$ , such that  $X_1 = M_1 \cap X$  and  $X_2 = M_2 \cap X$ .

Observe that, by Lemma 2.2,  $|X_1|, |X_2| \in [\frac{1}{2}|X| - \Delta, \frac{1}{2}|X| + \Delta]$  w.v.h.p.

2. Now, instead of pairing uniformly at random the elements from  $M_1$  with the elements from  $M_2$ , for our purposes it is enough to choose uniformly at random a set  $M'_2 \subseteq M_2$  of elements that are paired with  $X_1$ , and count how many elements of  $X_2$  we have chosen. Thus  $u_{XX} = |M'_2 \cap X_2|$ .

By Lemma 2.2 we have

$$\left| \frac{|M'_2 \cap X_2|}{|X_2|} - \frac{|M'_2|}{|M_2|} \right| = \mathcal{O}(|X_2|^{-1/2} \log |M_2|) \quad \text{w.v.h.p.},$$

so the following holds (since  $|M'_2| = |X_1|$ ) w.v.h.p.:

$$\left| u_{XX} - \frac{|X_1||X_2|}{n/2} \right| = \mathcal{O}(\sqrt{|X_2|} \log |M_2|).$$

Hence

$$\left| u_{XX} - \frac{|X|^2}{2n} \right| = \mathcal{O}(\sqrt{|X_2|} \log |M_2|) + \mathcal{O}\left(\frac{\Delta^2}{n/2}\right) = \mathcal{O}(\Delta). \quad \square$$

**Lemma 2.4** (pairs in partition). *Let  $\mathcal{F} = \{X_1, \dots, X_m\}$  be a partition of  $M$ . Then w.v.h.p.*

$$\left| \sum_{X \in \mathcal{F}} u_{XX} - \sum_{X \in \mathcal{F}} \frac{|X|^2}{2n} \right| = \mathcal{O}(n^{2/3} \log n).$$

*Proof.* Let  $\mathcal{F}'$  be a partition  $\{X'_1, \dots, X'_{m'}\}$  obtained from  $\mathcal{F}$  by merging all sets  $X_i$  such that  $|X_i| < n^{2/3}$  into larger sets of size between  $n^{2/3}$  and  $2n^{2/3}$  (this can be done in a greedy fashion). Let  $\mathcal{A}$  be the family of original large sets,  $\mathcal{B}$  the family of original small sets, and  $\mathcal{C}$  the family of new larger sets.

Since  $\mathcal{B}$  is a finer partition than  $\mathcal{C}$ , we have that

$$\sum_{X \in \mathcal{B}} u_{XX} \leq \sum_{X \in \mathcal{C}} u_{XX}, \quad \sum_{X \in \mathcal{B}} \frac{|X|^2}{2n} \leq \sum_{X \in \mathcal{C}} \frac{|X|^2}{2n}.$$

Since all sets in  $\mathcal{C}$  are smaller than  $2n^{2/3}$ , we obtain

$$\sum_{X \in \mathcal{C}} \frac{|X|^2}{2n} \leq \left( \sum_{X \in \mathcal{C}} |X| \right) n^{-1/3} \leq n^{2/3}.$$

Then, because all sets in  $\mathcal{C}$  are large, by a direct application of Lemma 2.3, w.v.h.p.

$$\sum_{X \in \mathcal{C}} u_{XX} \leq \sum_{X \in \mathcal{C}} \left( \frac{|X|^2}{2n} + \mathcal{O}(\sqrt{|X|} \log n) \right) \leq n^{2/3} + \mathcal{O}(\log n) \sum_{X \in \mathcal{C}} \sqrt{|X|} \leq \mathcal{O}(n^{2/3} \log n).$$

Similarly, because all sets in  $\mathcal{F}'$  are large, by an application of Lemma 2.3, w.v.h.p.

$$\left| \sum_{X \in \mathcal{F}'} u_{XX} - \sum_{X \in \mathcal{F}'} \frac{|X|^2}{2n} \right| \leq \sum_{X \in \mathcal{F}'} \mathcal{O}(\sqrt{|X|} \log n) = \mathcal{O}(n^{2/3} \log n).$$

Now our goal is to bound

$$\left| \sum_{X \in \mathcal{F}} u_{XX} - \sum_{X \in \mathcal{F}} \frac{|X|^2}{2n} \right|$$

which, because  $|a + b| \leq |a| + |b|$ , is at most

$$\left| \sum_{X \in \mathcal{B}} u_{XX} - \sum_{X \in \mathcal{C}} u_{XX} \right| + \left| \sum_{X \in \mathcal{F}'} u_{XX} - \sum_{X \in \mathcal{F}'} \frac{|X|^2}{2n} \right| + \left| \sum_{X \in \mathcal{B}} \frac{|X|^2}{2n} - \sum_{X \in \mathcal{C}} \frac{|X|^2}{2n} \right|.$$

The first and the third addend can be bounded by  $\mathcal{O}(n^{2/3} \log n)$  because if  $0 \leq a \leq b$  then  $|b - a| \leq b$ , hence by plugging in the previously derived bounds we obtain

$$\left| \sum_{X \in \mathcal{F}} u_{XX} - \sum_{X \in \mathcal{F}} \frac{|X|^2}{2n} \right| \leq \mathcal{O}(n^{2/3} \log n) + \mathcal{O}(n^{2/3} \log n) + \mathcal{O}(n^{2/3}) = \mathcal{O}(n^{2/3} \log n). \quad \square$$

**Lemma 2.5.** *Let  $X \subseteq M$  such that  $|X| = m$ . Let  $k(m', m, n)$  denote the number of draws without replacement until we hit  $m'$  elements from  $X$ . Then w.v.h.p.*

$$k(m', m, n) \leq \frac{n}{m} m' + \mathcal{O}\left(\frac{n}{\sqrt{m}} \cdot \log n\right).$$

*Proof.* Denoting by  $S$  the number hits on  $X$  when drawing without replacement  $k$  items, then by Lemma 2.2 w.v.h.p.

$$S \geq \frac{m}{n} k - C \cdot m^{1/2} \log n$$

for some constant  $C > 0$ . Substituting  $k = \frac{n}{m} m' + C \cdot \frac{n}{\sqrt{m}} \log n$  we obtain

$$S \geq m' + C \cdot m^{1/2} \log n - C \cdot m^{1/2} \log n = m'$$

meaning that indeed w.v.h.p. after  $k$  draws we will have at least  $m'$  elements from  $X$ . □

### 3 Algorithm

In this section we describe a randomized algorithm for finding majority. Recall that the algorithm is required to always either correctly determine a ball of the majority color or decide that there is no such color, and the majority color is a color of more than  $\lfloor n/2 \rfloor$  balls. For simplicity we will assume for the time being that  $n$  is even, as the algorithm can be adjusted for odd  $n$  in a straightforward manner without any change to the asymptotic cost. Hence to prove that there is a majority color, it is sufficient to find  $n/2 + 1$  balls with the same color. In such case our algorithm will actually calculate the multiplicity of the majority color. To prove that there is no majority color, it is sufficient to partition the input into  $n/2$  pairs of balls with different colors.

The algorithm consists of three parts. Intuitively, by choosing a small random sample we can approximate the color frequencies and choose the right strategy: (i) There is one color with a large frequency. We will use algorithm HEAVY. In essence, we have only one candidate for the majority, and we compute the frequency of the candidate in a naive manner. If the frequency is too small, we need to form sufficiently many pairs of balls with different colors. This can be done by reusing the results of the previous comparisons. (ii) There are two colors with frequencies close to 0.5. Now we will use algorithm BALANCED. In essence, we can now reduce the size of the input by a pairing process, and then find majority recursively. (iii) All frequencies are small. In this case we will use LIGHT which, again, resorts to a recursive strategy.

We start with presenting the main procedure of the algorithm; see Algorithm 1. The parameters are chosen by setting  $\alpha = \frac{1}{3}$ ,  $\varepsilon = n^{-1/10}$  and  $\beta = 0.45$ . In fact we could chose any  $\beta \in (\beta_1, \beta_2)$ , where  $\beta_1 = 1 - \frac{1}{\sqrt{3}} \approx 0.4226$  and  $\beta_2 \approx 0.47580$  is a root to  $p^3 - 19p^2 - 8p + 8 = 0$ .

---

**Algorithm 1:** MAJORITY( $M$ )

---

```

1 if  $|M| = 1$  then return  $M[1]$  is the majority with multiplicity 1 in  $M$   sample  $M' \subseteq M$  such that
    $|M'| = n^\alpha$ 
2 let  $v_1, v_2, \dots, v_k$  be the representatives of the colors in  $M'$ 
3 let  $q_i |M'|$  be the frequency of color( $v_i$ ) in  $M'$ , where  $q_1 \geq q_2 \geq \dots \geq q_k$ 
4 if  $q_1, q_2 \in [\frac{1}{2} - 4\varepsilon, \frac{1}{2} + 4\varepsilon]$  then
5   return BALANCED( $M$ )
6 else if  $q_1 \geq \beta$  and  $q_1^2 \geq q_2^2 + \dots + q_k^2 + 2\varepsilon$  then
7   return HEAVY( $M, v_1$ )
8 else
9   return LIGHT( $M$ )

```

---

Before we proceed to describe the subprocedures, we elaborate on the sampling performed in line 3. Intuitively, we would like to compute the frequencies of all colors in  $M$ . This would be too expensive, so we select a small sample  $M'$  and claim that the frequencies of all colors in  $M'$  are not too far from the frequencies of all colors in  $M$ . Formally, let  $p_1, p_2, p_3, \dots, p_\ell$  be the frequencies of all colors in  $M$ , that is there are  $p_i \cdot n$  balls of color  $i$  in  $M$  and let  $q_i$  be the frequency of color  $i$  in the sample  $M'$ . By Lemma 2.2, w.v.h.p.  $|p_i - q_i| = \mathcal{O}(n^{-\alpha/2} \log n) = o(\varepsilon)$ . We argue that  $\sum_i q_i^2$  is a good estimation of  $\sum_i p_i^2$ .

**Lemma 3.1.** *Let  $p_i$  be the frequency of color  $i$  in  $M$  and  $q_i$  be its frequency in  $M'$ , where  $M' \subseteq M$  a random sample without replacement of size  $n^\alpha$ . Then w.v.h.p.*

$$\left| \sum_i p_i^2 - \sum_i q_i^2 \right| = \mathcal{O}(n^{-\alpha/3} \log n) = o(\varepsilon).$$

*Proof.* Let  $m = n^\alpha$ . We analyze the following two sampling methods.

1. Partition the elements of  $M$  into  $\frac{n}{2}$  disjoint pairs uniformly at random. Select  $\frac{m}{2}$  of these pairs uniformly at random. Denote by  $A_1$  and  $A_2$  the pairs with both elements of the same colors in the first and the second pairing, respectively. By Lemma 2.4, w.v.h.p.  $\left| |A_1| - \frac{n}{2} \sum_i p_i^2 \right| = \mathcal{O}(n^{2/3} \log n)$ . Observe that by Lemma 2.2 w.v.h.p.  $\left| |A_2| - \frac{m}{n} |A_1| \right| = \mathcal{O}(m^{1/2} \log n)$ . Thus, by the triangle inequality, w.v.h.p.

$$\left| \frac{|A_2|}{m/2} - \sum_i p_i^2 \right| = \mathcal{O}(n^{-1/3} \log n) + \mathcal{O}(m^{-1/2} \log n).$$

2. Partition the elements of  $M'$  into  $\frac{m}{2}$  disjoint pairs uniformly at random, and denote by  $B$  all pairs with both elements of the same color. By Lemma 2.4, w.v.h.p.  $\left| |B| - \frac{m}{2} \sum_i q_i^2 \right| = \mathcal{O}(m^{2/3} \log n)$ , or equivalently  $\left| \frac{|B|}{m/2} - \sum_i q_i^2 \right| = \mathcal{O}(m^{-1/3} \log n)$ .

Now, because  $A_2$  and  $B$  have identical distributions, by the triangle inequality we have

$$\left| \sum_i p_i^2 - \sum_i q_i^2 \right| = \mathcal{O}(n^{-1/3} \log n) + \mathcal{O}(m^{-1/2} \log n) + \mathcal{O}(m^{-1/3} \log n) = \mathcal{O}(m^{-1/3} \log n). \quad \square$$

Now we present the subprocedures; see Algorithms 2–4.

---

**Algorithm 2:** BALANCED( $M$ )

---

```

1 randomly shuffle  $M$ 
2  $X \leftarrow [], Y \leftarrow []$ 
3 for  $i = 1$  to  $|M|/2$  do
4   if  $\text{cmp}(M[2i-1], M[2i])$  then
5     append  $M[2i]$  to  $X$ 
6   else
7     append  $M[2i-1]$  and  $M[2i]$  to  $Y$ 
8 run MAJORITY( $X$ )
9 if there is no majority in  $X$  then return no majority in  $M$  let  $\text{color}(v)$  be the majority with
   multiplicity  $k$  in  $X$ 
10  $\text{cnt} \leftarrow 2k$ 
11 for  $i = 1$  to  $|Y|/2$  do
12   if  $\text{cmp}(v, Y[2i-1])$  then
13      $\text{cnt} \leftarrow \text{cnt} + 1$ 
14   else if  $\text{cmp}(v, Y[2i])$  then
15      $\text{cnt} \leftarrow \text{cnt} + 1$ 
16 if  $\text{cnt} \leq |M|/2$  then
17   return no majority in  $M$ 
18 else
19   return  $\text{color}(v)$  is the majority with multiplicity  $k$  in  $X$ 

```

---

**Lemma 3.2.** *Algorithm 1 always returns the correct answer.*

*Proof.* We analyze separately every subprocedure.

BALANCED( $M$ ). If the majority exists then removing two elements with different colors preserves it. Hence if the recursive call returns that there is no majority in  $X$  then indeed there is no majority in  $M$ , and otherwise  $\text{color}(v)$  is the only possible candidate for the majority in  $M$ . The remaining part of the subprocedure simply verifies it.

HEAVY( $M, v$ ). The subprocedure first checks if  $\text{color}(v)$  is the majority. Hence it is enough to analyze what happens if  $\text{color}(v)$  is not the majority. Then  $X$  contains all elements with other colors. We partition the elements in  $X$  into pairs and check which of these pairs consists of elements with different colors. If the number of elements in all the remaining pairs is smaller than the number of elements of color  $\text{color}(v)$ , then clearly we can partition all elements in  $M$  into disjoint pairs of elements with different colors, hence indeed there is no majority. Otherwise, we revert to the simple  $2n$  algorithm, which is always correct.

LIGHT( $M$ ). Again, if the majority exists then removing two elements with different color preserves it. Hence we can assume that  $\text{color}(v)$  is the only possible candidate for the majority. Then,  $Y$  consists of pairs of two elements with different colors. From the recursive call we also know what is the frequency of  $\text{color}(v)$  in  $M \setminus Y$ . We iterate through the elements of  $Y$  and check if their color is  $\text{color}(v)$ . However, if the color of the first element in a pair is  $\text{color}(v)$ , then the second element has a different color. So the subprocedure

---

**Algorithm 3:** HEAVY( $M, v$ )

---

```
1 cnt ← 0, X ← []
2 for i = 1 to |M| do
3   if cmp(v, M[i]) then
4     cnt ← cnt + 1
5   else
6     append M[i] to X
7 if cnt > |M|/2 then return color(v) is the majority with multiplicity k in M k ← |M|/2 - cnt
8 randomly shuffle X
9 for i = 1 to |X|/2 do
10  if ¬cmp(X[2i - 1], X[2i]) then
11    k ← k - 1
12  if k = 0 then return no majority in M
13 return BOYER-MOORE(M) ▷ fallback, 2n comparisons
```

---

---

**Algorithm 4:** LIGHT( $M$ )

---

```
1 randomly shuffle M
2 X ← [], Y ← []
3 for i = 1 to |M|/2 do
4   if cmp(M[2i - 1], M[2i]) then
5     append M[2i] to X
6   else
7     append M[2i - 1] and M[2i] to Y
8 run MAJORITY(X)
9 if there is no majority in X then return no majority in M let color(v) be the majority with
  multiplicity k in X
10 cnt ← 2k - |X|
11 for i = 1 to |Y| do
12  if ¬cmp(v, Y[2i - 1]) then
13    if ¬cmp(v, Y[2i]) then
14      cnt ← cnt - 1
15  if cnt = 0 then return no majority in M
16 return color(v) is the majority with multiplicity (|M|/2 + cnt) in M
```

---

either correctly determines the frequency of the majority  $\text{color}(v)$ , or find sufficiently many elements with different colors to conclude that  $\text{color}(v)$  is not the majority.  $\square$

**Theorem 3.3.** *Algorithm 1 w.v.h.p. uses at most  $\frac{7}{6}n + o(n)$  comparisons on an input of size  $n$ . The expected number of comparisons is also at most  $\frac{7}{6}n + o(n)$ .*

*Proof.* Let  $T(n)$  be a random variable counting the comparisons on the given input of size  $n$ . We will inductively prove that  $T(n) \leq \frac{7}{6}n + C \cdot n^{9/10}$  for a fixed constant  $C$  that is sufficiently large. In the analysis we will repeatedly invoke Lemmas 2.2, 2.3, 2.4, 2.5, 3.1 and Chernoff bound to bound different quantities. We will assume that each such the application succeeds. Since there will be a polynomial number of applications, each on a polynomial number of elements, this happens w.v.h.p. with respect to the size of the input. We also assume that  $n$  is large enough. Algorithm 1 uses at most  $\mathcal{O}(n^{2\alpha}) = \mathcal{O}(n^{2/3})$  comparisons in the sampling stage. We bound the number of subsequent comparisons used by each subprocedure as follows.

**BALANCED( $M$ ).** We have that  $p_1, p_2 = \frac{1}{2} \pm \mathcal{O}(\varepsilon)$ . Thus also  $\sum_i p_i^2 = \frac{1}{2} \pm \mathcal{O}(\varepsilon)$ . By Lemma 2.4,  $|X| = (\frac{n}{2} \sum_i p_i^2) \pm \mathcal{O}(n^{2/3} \log n)$ , thus  $|X| = (\frac{1}{4} \pm \mathcal{O}(\varepsilon))n$ . Also  $|Y| = n - 2|X| = (\frac{1}{2} \pm \mathcal{O}(\varepsilon))n$ .

List  $Y$  consists of pairs of elements with different colors. Because at most  $\mathcal{O}(\varepsilon n)$  of all elements are not of color 1 or 2, there are at most  $\mathcal{O}(\varepsilon n)$  pairs not of the form  $\{1, 2\}$ . Since the relative order of elements  $Y[2i-1]$  and  $Y[2i]$  is random, for each pair  $\{1, 2\}$  we pay 1 with probability  $1/2$  and pay 2 with probability  $1/2$ , and for any other pair we pay always 2. Thus the total cost incurred by the loop in line 11 is (by Chernoff bound) at most

$$\mathcal{O}(\varepsilon n) \cdot 2 + \frac{3}{2}|Y|/2 + \mathcal{O}(\sqrt{|Y|} \log n) \leq \frac{3}{8}n \pm \mathcal{O}(\varepsilon n).$$

Thus the total cost is

$$T(n) \leq T\left(\left(\frac{1}{4} + \varepsilon\right)n\right) + \frac{1}{2}n + \frac{3}{8}n + \mathcal{O}(\varepsilon n) \leq \frac{7}{6}n + \mathcal{O}(n^{9/10}) + C \cdot \left(\frac{1}{3}n\right)^{9/10}$$

and  $\frac{7}{6}n + \mathcal{O}(n^{9/10}) + C \cdot \left(\frac{1}{3}\right)^{9/10} \cdot n^{9/10} \leq \frac{7}{6}n + C \cdot n^{9/10}$  for a large enough  $C$ .

**HEAVY( $M, v$ ).** If  $p_1 > \frac{1}{2}$ , then we terminate in line 7 after  $n$  comparisons. Thus we can assume that  $p_1 \in [0.45 - \varepsilon, \frac{1}{2}]$ . Because by Lemmas 2.2 and 3.1 both  $p_1^2$  and  $\sum_i p_i^2$  are estimated within an absolute error of  $o(\varepsilon)$ , we have that  $p_1^2 - \sum_{i \geq 2} p_i^2 \geq 2\varepsilon - 2o(\varepsilon) \geq \varepsilon$ .

We argue that the loop in line 9 will eventually find sufficiently many pairs of elements with different colors, and thus return without falling back to the  $2n$  algorithm. By definition,  $|X| = (1 - p_1)n$  and initially  $k = (1/2 - p_1)n$ . By Lemma 2.4, after the random shuffle the number of pairs of elements  $(X[2i-1], X[2i])$  with different colors, denoted by  $D$ , can be bounded by

$$\begin{aligned} D &\geq \frac{|X|}{2} - \frac{\sum_{j \geq 2} (p_j n)^2}{2|X|} - \mathcal{O}(|X|^{2/3} \log |X|) \geq \\ &\geq \frac{1 - p_1}{2}n - \frac{p_1^2 - \varepsilon}{2(1 - p_1)}n - o(\varepsilon n) \geq \frac{1 - 2p_1}{2(1 - p_1)}n + \frac{\varepsilon}{2}n - o(\varepsilon n) \geq \left(\frac{1}{2} - p_1\right)n; \end{aligned}$$

thus indeed there are sufficiently many pairs. Hence, because the pairs are being considered in a random order, the total cost can be bounded using Lemma 2.5 by

$$\begin{aligned} T(n) &\leq n + \frac{|X|}{D} \left(\frac{1}{2} - p_1\right)n + \mathcal{O}\left(\frac{|X|}{\sqrt{D}} \log |X|\right) \leq \\ &\leq n + \frac{(1 - p_1)^2}{2}n + \mathcal{O}\left(n/\sqrt{\frac{\varepsilon}{3}n} \log n\right) \leq \\ &\leq (1 + 0.55^2/2)n + \mathcal{O}(\varepsilon n) + \mathcal{O}\left(\sqrt{\frac{n}{\varepsilon}} \log n\right) = 1.15125n + \mathcal{O}(n^{9/10}), \end{aligned}$$

where we used  $D \geq \frac{\varepsilon}{2} - o(\varepsilon n) \geq \frac{\varepsilon}{3}n$  for a large enough  $n$ .

**LIGHT( $M$ ).** We start with bounding the sizes of  $X$  and  $Y$ . By Lemma 2.4,  $|X| = \frac{n}{2} \sum_i p_i^2 \pm \mathcal{O}(n^{2/3} \log n)$ . Also, by Lemma 2.3 there are  $\frac{n}{2}p_1^2 \pm \mathcal{O}(n^{1/2} \log n)$  elements from  $A_1$  in  $X$ , thus there are  $n(p_1 - p_1^2) \pm \mathcal{O}(n^{1/2} \log n)$  of elements from  $A_1$  in  $Y$  (each paired with an element not from  $A_1$ ).

We know that either  $p_1 \leq 0.45 + \varepsilon$  or  $p_1^2 - \sum_{i \geq 2} (p_i^2) \leq \varepsilon$ . If there is no majority in  $X$ , then  $p_1 \leq \frac{1}{2}$  and the total cost is bounded by

$$T(n) \leq \frac{n}{2} + T(|X|) \leq \frac{n}{2} + \frac{7}{6}|X| + C \cdot |X|^{9/10},$$

which, because  $|X| \leq \frac{n}{4} + \mathcal{O}(n^{2/3} \log n)$ , is less than  $\frac{19}{24}n + o(n)$ . Hence we can assume that there is a majority in  $X$ . In such case,  $\text{cnt}$  is set to

$$c = \frac{n}{2} \left( p_1^2 - \sum_{i \geq 2} p_i^2 \right) \pm \mathcal{O}(n^{2/3} \log n).$$



We denote by  $I$  the total number of iterations of the loop in line 11. By Lemma 2.5

$$I \leq \frac{\frac{1}{2}|Y|}{\frac{1}{2}|Y| - |A_1 \cap Y|} \cdot c + \mathcal{O}(E),$$

where  $E = |Y|/\sqrt{\frac{1}{2}|Y| - |A_1 \cap Y|}$ . Substituting  $S = \sum_{i \geq 2} p_i^2$ , by Lemma 2.4 we have

$$\begin{aligned} |Y| &= (1 - p_1^2 - S \pm \mathcal{O}(n^{-1/3} \log n))n, \\ |Y| - 2|A_1 \cap Y| &= ((1 - p_1)^2 - S \pm \mathcal{O}(n^{-1/3} \log n))n, \\ c &= \frac{1}{2}(p_1^2 - S \pm \mathcal{O}(n^{-1/3} \log n))n. \end{aligned}$$

Since  $p_1 \leq \frac{1}{2}$  and  $p_2 \leq \frac{1}{2} - 3\varepsilon$  (as for a larger  $p_2$  the sampled  $q_2$  would be sufficiently large for other subprocedure to be used), we have

$$(1 - p_1)^2 - S - o(\varepsilon) \geq \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2} - 3\varepsilon\right)^2 - (3\varepsilon)^2 - o(\varepsilon) = 3\varepsilon - 18\varepsilon^2 - o(\varepsilon) \geq 2\varepsilon.$$

Thus  $E \leq \sqrt{\frac{n}{2\varepsilon}}$ . Now, since  $|Y| = \Theta(n)$  we can bound  $I$  from above by

$$\begin{aligned} I &\leq \frac{|Y|}{|Y| - 2|A_1 \cap Y|} \cdot \frac{1}{2}(p_1^2 - S)n + \mathcal{O}(1/\varepsilon) \cdot \mathcal{O}(n^{2/3} \log n) + \mathcal{O}(E) \leq \\ &\leq \frac{1}{2} \frac{1 - p_1^2 - S + \mathcal{O}(n^{-1/3} \log n)}{(1 - p_1)^2 - S - \mathcal{O}(n^{-1/3} \log n)} (p_1^2 - S)n + \mathcal{O}\left(\frac{n^{2/3} \log n}{\varepsilon}\right) + \mathcal{O}\left(\sqrt{\frac{n}{4\varepsilon}}\right) \leq \\ &\leq \frac{1}{2} \frac{1 - p_1^2 - S}{(1 - p_1)^2 - S - \mathcal{O}(n^{-1/3} \log n)} (p_1^2 - S)n + \frac{\mathcal{O}(n^{2/3} \log n)}{2\varepsilon} + \mathcal{O}(n^{23/30} \log n), \end{aligned}$$

which, because  $(1 - p_1)^2 - S$  is sufficiently large, can be bounded by

$$\begin{aligned} I &\leq \frac{1}{2} \frac{1 - p_1^2 - S}{(1 - p_1)^2 - S} (p_1^2 - S)n \cdot \left(1 + \frac{\mathcal{O}(n^{-1/3} \log n)}{(1 - p_1)^2 - S}\right) + \mathcal{O}(n^{23/30} \log n) \leq \\ &\leq \frac{1}{2} \frac{1 - p_1^2 - S}{(1 - p_1)^2 - S} (p_1^2 - S)n \cdot \left(1 + \frac{\mathcal{O}(n^{-1/3} \log n)}{2\varepsilon}\right) + \mathcal{O}(n^{23/30} \log n) \leq \\ &\leq \frac{1}{2} \frac{1 - p_1^2 - S}{(1 - p_1)^2 - S} (p_1^2 - S)n + \frac{\mathcal{O}(n^{2/3} \log n)}{8\varepsilon^2} + \mathcal{O}(n^{23/30} \log n) \leq \\ &= \frac{1}{2} (1 - p_1^2 - S) \frac{p_1^2 - S}{(1 - p_1)^2 - S} n + \mathcal{O}(n^{26/30} \log n). \end{aligned}$$

For each of  $c$  iterations we pay 2, and for each of the remaining  $I - c$  iterations we pay only  $\frac{3}{2}$  in expectation (for each iteration independently). Thus, by Chernoff bound the total cost is

$$\begin{aligned} T(n) &\leq \frac{1}{2}n + T(|X|) + \frac{3}{2}(I - c) + \mathcal{O}(\sqrt{I - c} \log(I - c)) + 2c \leq \\ &\leq \frac{1}{2}n + \frac{7}{12}n(p_1^2 + S) + \frac{3}{4}(1 - p_1^2 - S) \frac{p_1^2 - S}{(1 - p_1)^2 - S} n + \frac{1}{4}(p_1^2 - S)n + \mathcal{O}(n^{26/30} \log n) + C \cdot |X|^{9/10} = \\ &= \frac{n}{2} \left(1 + \frac{7}{6}(p_1^2 + S) + \frac{3}{2}(p_1^2 - S) \frac{1 - p_1^2 - S - ((1 - p_1)^2 - S)}{(1 - p_1)^2 - S} + \frac{3}{2}(p_1^2 - S) + \frac{1}{2}(p_1^2 - S)\right) + \mathcal{O}(n^{9/10}) = \\ &= \frac{n}{2} \left(1 + \frac{19}{6}p_1^2 - \frac{5}{6}S + 3(p_1^2 - S) \frac{p_1 - p_1^2}{(1 - p_1)^2 - S}\right) + \mathcal{O}(n^{9/10}). \end{aligned}$$

We reason that, for a fixed  $p_1$ , the quantity

$$1 + \frac{19}{6}p_1^2 - \frac{5}{6}S + 3(p_1^2 - S)\frac{p_1(1-p_1)}{(1-p_1)^2 - S}$$

is a decreasing function of  $S$ , since  $p_1^2 \leq (1-p_1)^2$ . Now we consider two cases. If  $p_1^2 - S \leq \varepsilon$  then simplifying above estimation either with  $p_1^2 - S \leq 0$ , or, since  $(1-p_1)^2 - S \geq 2\varepsilon$ , with  $0 \leq \frac{p_1^2 - S}{(1-p_1)^2 - S} \leq \frac{1}{2}$ , we get either

$$T(n) \leq \frac{n}{2} \left( 1 + \frac{7}{3}p_1^2 \right) + \mathcal{O}(n^{9/10}),$$

which, since  $p_1 \leq \frac{1}{2}$ , is bounded from above by  $\frac{19}{24}n + o(n)$ , or

$$T(n) \leq \frac{n}{2} \left( 1 + \frac{7}{3}p_1^2 + \frac{3}{2}p_1 - \frac{3}{2}p_1^2 \right) + \mathcal{O}(n^{9/10}),$$

which is bounded from above by  $\frac{47}{48}n + o(n)$ .

Otherwise,  $p_1 \leq 0.45 + \varepsilon$  and substituting  $S = 0$  (since the cost is decreasing in  $S$ ) we obtain

$$T(n) \leq \frac{n}{2} \left( 1 + \frac{19}{6}p_1^2 + 3\frac{p_1^3}{1-p_1} \right) + \mathcal{O}(n^{9/10}) = 1.06915n + \mathcal{O}(n^{9/10}).$$

**Wrapping up.** We see that in each subprocedure, the number of comparisons is bounded by  $\frac{7}{6}n + C \cdot n^{9/10}$ . Each subprocedure makes at most one recursive call, where the size of the input is reduced by at least a factor of 2. Thus the worst-case number of comparison is always bounded by  $\mathcal{O}(n)$ . Recall that the bound on the number of comparisons used by every recursive call holds w.v.h.p. with respect to the size of the input to the call. Eventually, the size of the input might become very small, and then w.v.h.p. with respect to the size of the input is no longer w.v.h.p. with respect to the original  $n$ . However, as soon as this size decreases to, say,  $n^{0.1}$ , the number of comparisons is  $\mathcal{O}(n)$  irrespectively of the random choices made by the algorithm. Thus w.v.h.p. the number of comparisons is at most  $\frac{7}{6}n + \mathcal{O}(n^{9/10})$ , and the expected number of comparisons is also bounded by  $\frac{7}{6}n + \mathcal{O}(n^{9/10})$ .  $\square$

## 4 Lower bound

We consider only Las Vegas algorithms. That is, the algorithm must always correctly determine whether a majority elements exists. We want to prove that the expected number of comparisons used by any such algorithm is at least  $c \cdot n - o(n)$ , for some constant  $c > 1$ . By Yao's principle, it is sufficient to construct a distribution on the inputs, such that the expected number of comparisons used by any deterministic algorithms run on an input chosen from the distribution is at least  $c \cdot n - o(n)$ . The distribution is that with probability  $\frac{1}{n}$  every ball has a color chosen uniformly at random from a set of  $n$  possible colors. With probability  $1 - \frac{1}{n}$  every ball is either black or white, with both possibilities equally probable. We fix a correct deterministic algorithm  $\mathcal{A}$  and analyze its behavior on an input chosen from the distribution. As a warm-up, we will prove first that  $\mathcal{A}$  needs  $n - o(n)$  comparisons in expectation on such input.

### 4.1 A lower bound of $n - o(n)$

In every step  $\mathcal{A}$  compares two balls, thus we can describe its current knowledge by defining an appropriate graph as follows. Every node corresponds to a ball. Two nodes are connected with a *negative edge* if the corresponding balls have been compared and found out to have different colors. Two nodes are connected with a *positive edge* if the corresponding balls are known to have the same colors under the assumption that every ball is either black or white (either because they have been directly compared and found to have the same color, or because such knowledge has been indirectly inferred from the assumption). After every step of

the algorithm the graph consists of a number of components  $C_1, C_2, \dots$ . Every component is partitioned into two parts  $C_i = A_i \cup B_i$ , such that both  $A_i$  and  $B_i$  are connected components in the graph containing only positive edges and there is at least one (possibly more than one) negative edge between  $A_i$  and  $B_i$ . There are no other edges in the graph. Now we describe how the graph changes after  $\mathcal{A}$  compares two balls  $x \in C_i$  and  $y \in C_j$  assuming that every ball is either black or white. If  $i = j$  then the result of the comparison is already determined by the previous comparisons and the graph does not change. Otherwise,  $i \neq j$  and assume by symmetry that  $x \in A_i, y \in A_j$ . The following two possibilities are equally probable:

1.  $\text{color}(x) = \text{color}(y)$ , then we merge both components into a new component  $C = A \cup B$ , where  $A = A_i \cup A_j$  and  $B = B_i \cup B_j$  by creating new positive edges  $(x, y)$  and  $(x', y')$  for some  $x' \in B_i, y' \in B_j$  (if  $B_i, B_j \neq \emptyset$ ).
2.  $\text{color}(x) \neq \text{color}(y)$ , then we merge both components into a new component  $C = A \cup B$ , where  $A = A_i \cup B_j$  and  $B = B_i \cup A_j$  by creating new positive edges  $(x, y')$  for some  $y' \in B_j$  (if  $B_j \neq \emptyset$ ) and  $(x', y)$  for some  $x' \in B_i$  (if  $B_i \neq \emptyset$ ). We also create a new negative edge  $(x, y)$ . Observe that here we crucially use the assumption that every ball is either black or white.

The graph exactly captures the knowledge of  $\mathcal{A}$  about a binary input.

Any binary input contains a majority and  $\mathcal{A}$  must report so. However, because with very small probability the input is arbitrary, this requires some work due to the following lemma.

**Lemma 4.1.** *If  $\mathcal{A}$  reports that a binary input contains a majority element, then the graph contains a component  $C = A \cup B$  such that  $|A| > \frac{n}{2}$  or  $|B| > \frac{n}{2}$ .*

*Proof.* Assume otherwise, that is,  $\mathcal{A}$  reports that a binary input contains a majority element even though both parts of every component are of size less than  $\frac{n}{2}$ . Construct another input by choosing, for every component  $C = A \cup B$ , two fresh colors  $c_A$  and  $c_B$  and setting  $\text{color}(x) = c_A$  for every  $x \in A$ ,  $\text{color}(y) = c_B$  for every  $y \in B$ . Every comparison performed by  $\mathcal{A}$  is an edge of the graph, so its behavior on the new input is exactly the same as on the original binary input. Hence  $\mathcal{A}$  reports that there is a majority element, while the frequency of every color in the new input is less than  $\frac{n}{2}$ , which is a contradiction.  $\square$

From now on we consider only binary inputs. If we can prove that the expected number of comparisons used by  $\mathcal{A}$  on such input is  $n - o(n)$ , then the expected number of comparisons on an input chosen from our distribution is also  $n - o(n)$ . Because every comparison decreases the number of components by one, it is sufficient to argue that the expected size of some component when  $\mathcal{A}$  reports that there is a majority is  $n - o(n)$ . We already know that there must exist a component  $C = A \cup B$  such that (by symmetry)  $|A| > \frac{n}{2}$ . We will argue that  $|B|$  must also be large. To this end, define *balance* of a component  $C_i = A_i \cup B_i$  as  $\text{balance}(C_i) = (|A_i| - |B_i|)^2$ , and the total balance as  $\sum_i \text{balance}(C_i)$ .

**Lemma 4.2.** *The expected total balance when  $\mathcal{A}$  at any given step is  $n$ .*

*Proof.* In the very beginning the total balance is  $n$  because every component is a singleton. Recall that when  $\mathcal{A}$  compares two balls  $x \in C_i$  and  $y \in C_j$  then  $C_i$  and  $C_j$  are merged into a new component  $C$ . It is easy to verify that if  $\text{balance}(C_i) = b_i^2$  and  $\text{balance}(C_j) = b_j^2$  then either  $\text{balance}(C) = (b_i + b_j)^2$  or  $\text{balance}(C) = (b_i - b_j)^2$ , with both possibilities equally probable. Hence the expected total balance after the comparison is equal to the previous total balance increased by  $-b_i^2 - b_j^2 + \frac{1}{2}((b_i - b_j)^2 + (b_i + b_j)^2) = 0$ . Consequently, the expected total balance after any comparison is equal to the original total balance  $n$ .  $\square$

Total balance when  $\mathcal{A}$  reports a majority is a random variable with expected value  $n$ . By Markov's inequality, with probability  $1 - 1/n^{1/3}$  its value is at most  $n^{4/3}$ , which implies that for any component  $C_i = A_i \cup B_i$ ,  $\text{balance}(C_i) \leq n^{4/3}$ , and there exists a component  $C = A \cup B$  such that  $|A| > \frac{n}{2}$ , so  $||A| - |B|| \leq n^{2/3}$  and  $|B| \geq \frac{n}{2} - n^{2/3}$ . We obtain that with probability  $1 - 1/n^{1/3}$  there is a component containing  $n - n^{2/3}$  nodes. Every comparison decreased the number of components by at most 1, and if there is such a large component then the total number of components cannot exceed  $1 + n - (n - n^{2/3}) = n^{2/3} + 1$ , which means that the algorithm must have performed  $n - n^{2/3} - 1$  comparisons. This makes the expected number of comparisons at least  $(1 - 1/n^{1/3})(n - n^{2/3} - 1) = n - o(n)$ .

## 4.2 A stronger lower bound

To obtain a stronger lower bound we extend the definition of the graph capturing the current knowledge of  $\mathcal{A}$ . Now a positive edge can be *verified* or *non-verified*. A verified positive edge  $(x, y)$  is created only after comparing two balls  $x$  and  $y$  such that  $\text{color}(x) = \text{color}(y)$ . All other positive edges are non-verified. The algorithm can also turn a non-verified positive edge  $(x, y)$  into a verified positive edge by comparing  $x$  and  $y$ . By the same reasoning as in Lemma 4.1 we obtain the following.

**Lemma 4.3.** *If  $\mathcal{A}$  reports that a binary input contains a majority element, then the graph consisting of all verified positive edges contains a connected component on at least  $\frac{n}{2}$  nodes.*

Now that the goal is to construct a large component in the graph consisting of all verified positive edges, it also make sense for  $\mathcal{A}$  to compare two balls from the same component. However, without loss of generality, such comparisons are executed after having identified a large component in the graph consisting of all positive edges. Then,  $\mathcal{A}$  asks sufficiently many queries of the form  $(x, y)$ , where  $(x, y)$  is a non-verified edge from the identified component. In other words,  $\mathcal{A}$  first isolates a candidate for a majority, and then makes sure that all inferred equalities really hold, which is necessary because with very small probability the input is not binary. This allows us to bound the total number of comparisons from below as follow.

**Lemma 4.4.** *The expected number of comparisons used by  $\mathcal{A}$  on a binary input is at least  $n - o(n)$  plus the expected number of non-verified edges between nodes of the majority color.*

*Proof.* Recall that with probability  $1 - 1/n^{1/3}$  when there exists a component  $C = A \cup B$  such that  $|A| > \frac{n}{2}$  then  $|B| \geq \frac{n}{2} - n^{2/3}$ . Set  $A$  consists of nodes of the majority color, although possibly not all nodes of the majority color are there. However, because  $B$  is large, there are at most  $n^{2/3}$  nodes of the majority color outside of  $A$ . Also, because we consider binary inputs chosen uniformly at random, by Chernoff bound  $|A| \leq \frac{n}{2} + \mathcal{O}(\sqrt{n \log n})$  with probability  $1 - \frac{1}{n}$ .

The expected number of comparisons used by  $\mathcal{A}$  to construct a component  $C = A \cup B$  such that  $|A| > \frac{n}{2}$  is  $n - n^{2/3} - 1$ . Then,  $\mathcal{A}$  needs to verify sufficiently many non-verified edges inside  $A$  to obtain a connected component of size  $\frac{n}{2}$  in the graph consisting of verified positive edges. By construction, there are no cycles in the graph consisting of positive edges, hence with probability  $1 - \frac{1}{n^{1/3}} - \frac{1}{n}$  there will be no more than  $n^{2/3} + \mathcal{O}(\sqrt{n \log n})$  non-verified positive edges between nodes outside of  $B$  when  $\mathcal{A}$  reports a majority. Consequently, the additional verification cost is the expected number of non-verified edges between nodes of the majority color minus  $n^{2/3} + \mathcal{O}(\sqrt{n \log n}) = o(n)$ .  $\square$

In the remaining part of this section we analyze the expected number of non-verified edges between nodes of the majority color constructed during the execution of the algorithm. The goal is to show that this is at least  $(c - 1)n - o(n)$ . Then, Lemma 4.4 implies the claimed lower bound.

A component  $C = A \cup B$  is called *monochromatic* when  $A = \emptyset$  or  $B = \emptyset$  (by symmetry, we will assume the latter) and *dichromatic* otherwise. With probability  $1 - 1/n^{1/3}$ , when  $\mathcal{A}$  reports a majority there is one large dichromatic component on at least  $n - n^{2/3}$  nodes, so the total number of components is at most  $n^{2/3} + 1$ . It is convenient to think about execution of  $\mathcal{A}$  as of a process of eliminating components by merging two components into one. Each such merge might create a new non-verified edge. The cost of such non-verified edge  $(x, y)$  is the probability that  $x$  and  $y$  are nodes of the majority color. We want to argue that because all but  $n^{2/3}$  components will be eventually eliminated, the total cost of all created non-verified edges is  $(c - 1)n - o(n)$ .

We analyze in more detail the merging process in terms of mono- and dichromatic components. Let  $\text{predict}_k$  be the random variable denoting the probability that, after  $k$  steps of  $\mathcal{A}$ , a node from the larger part of a component is of the majority color. It is rather difficult to calculate  $\text{predict}_k$  exactly, so we will actually use a rather crude upper bound instead. An important property of the upper bound will be that it is nondecreasing in  $k$ . When  $\mathcal{A}$  compares two balls  $x \in C_i$  and  $y \in C_j$  with  $i \neq j$  to obtain a new component  $C = A \cup B$  there are the following cases.

1.  $C_i$  and  $C_j$  are monochromatic. Then with probability  $\frac{1}{2}$  the new component  $C$  is also monochromatic, and with probability  $\frac{1}{2}$  it is dichromatic.

2.  $C_i$  is dichromatic and  $C_j$  is monochromatic. The new component is dichromatic and, with probability  $\frac{1}{2}(1 - \text{predict}_k)$ , contains a new non-verified edge.
3.  $C_i$  and  $C_j$  are dichromatic. Then with probability  $\frac{1}{2}$  we create a new non-verified edge inside both  $A$  and  $B$ .

We analyze the expected total cost of all non-verified edges when only one component remains. When  $\mathcal{A}$  reports a majority up to  $n^{2/3}$  components might remain, but this changes only the lower order terms of the bound.

**Lemma 4.5.** *The expected total cost of all non-verified edges when only one component remains is at least  $\sum_{k=1}^{2n/3} \mathbb{E}[\min(\frac{1}{6}, \frac{1}{2}(1 - \text{predict}_k))]$ .*

*Proof.* We start with  $n$  components and need to eliminate all but at most one of them. To each component we associate credit,  $\frac{1}{2}$  to each dichromatic and  $\frac{1}{6}$  to each monochromatic one. The algorithm can collect the credit from both of the components it merges, but it has to pay for credit of newly created one. Additionally algorithm has to pay for any non-verified edge created by merging.

In every step we have three possibilities:

1. Merge two monochromatic components into one. With probability  $\frac{1}{2}$  new component is dichromatic, and with probability  $\frac{1}{2}$  new component is monochromatic. Thus the expected amortized cost for this step is 0.
2. Merge a monochromatic components with a dichromatic component. Then the total number of monochromatic components decreases by 1 and we add with probability at least  $\frac{1}{2}(1 - \text{predict}_k)$  a non-verified edge. The expected amortized cost for this step is  $\frac{1}{2}(1 - \text{predict}_k) - \frac{1}{6}$ .
3. Merge two dichromatic components while adding with probability  $\frac{1}{2}$  a non-verified edge. The expected amortized cost for this step is 0.

In total algorithm has to pay for initial credits and for each step, making the total expected cost at least

$$\frac{n}{6} + \sum_{k=1}^{n-1} \mathbb{E}[\min(0, \frac{1}{2}(1 - \text{predict}_k) - \frac{1}{6})] \geq \sum_{k=1}^{2/3n} \mathbb{E}[\min(\frac{1}{6}, \frac{1}{2}(1 - \text{predict}_k))]. \quad \square$$

We note that by truncating the sum at  $\frac{2}{3}n$  we do not lose any cost estimation, as for  $k \geq \frac{2}{3}n$  our estimation for  $\text{predict}_k$  gives 1.

Now we focus on upper bounding the expression obtained in Lemma 4.5. To bound  $\text{predict}_k$  we use an approach due to Christofides [4]. At any given step  $k$  we will look at all the nonzero balances of the components. Specifically, we introduce two new random variables:  $M_k$  being the largest balance of a component, and  $N_k$  being the number of components with nonzero balance. Since at each step, number of nonzero balance components is decreased at most in expectation by  $\frac{3}{2}$ , we have  $\mathbb{E}[N_k] \geq n - \frac{3}{2}(k-1)$ , and w.v.h.p., by Chernoff bound  $N_k \geq n - \frac{3}{2}k - \mathcal{O}(\sqrt{k} \log n)$ .

Since by Lemma 4.2 expected sum of balances is  $n$ , and each nonzero component contributes at least 1 to the sum, we have  $\mathbb{E}[M_k] \leq n - \mathbb{E}[N_k - 1] = \frac{3}{2}k - \frac{1}{2}$ .

Now to proceed, for a component  $C_i = A_i \cup B_i$  we define  $\delta_i = ||A_i| - |B_i||$ , a positive value such that  $\delta_i^2 = \text{balance}(C_i)$ . Thus, at any given step  $k$ , algorithm observes nonzero values  $\delta_1, \delta_2, \dots, \delta_{N_k}$ . Without loss of generality we can narrow our focus on a component  $C_1$ . We are interested in bounding the probability

$$\Pr(A_1 \text{ in majority}) = \Pr(\delta_1 + \varepsilon_2 \delta_2 \dots + \varepsilon_{N_k} \delta_{N_k} \geq 0) = \frac{1}{2} + \frac{1}{2} \Pr(\varepsilon_2 \delta_2 \dots + \varepsilon_{N_k} \delta_{N_k} \in [-\delta_1, \delta_1]),$$

where  $\varepsilon_2, \varepsilon_3, \dots, \varepsilon_{N_k} \in \{-1, 1\}$  are drawn independently and uniformly at random. By a result of Erdős [9], the above probability, when  $\delta_2, \dots, \delta_{N_k} \geq 1$  is maximized when  $\delta_2, \dots, \delta_{N_k}$  are all set to 1.

We now approximate binomial distribution using the symmetric case of de Moivre–Laplace Theorem. Recall that

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt$$

is the *cumulative distribution function* of the *normal distribution*.

**Theorem 4.6** (De Moivre–Laplace). *Let  $S_n$  be the number of successes in  $n$  independent coin flips. Then*

$$\Pr\left(\frac{n}{2} + x_1\sqrt{n} \leq S_n \leq \frac{n}{2} + x_2\sqrt{n}\right) \sim \Phi(2x_2) - \Phi(2x_1).$$

In our case we are interested in  $N_k - 1$  coin flips and the number of successes in the range  $[(N_k - 1)/2 - \delta_1/2, (N_k - 1)/2 + \delta_1/2]$ . Thus probability that 1 is the majority can be bounded from above by

$$\Pr(A_1 \text{ is the majority}) \leq \Phi\left(\frac{\delta_1}{\sqrt{N_k - 1}}\right) - \Phi\left(-\frac{\delta_1}{\sqrt{N_k - 1}}\right) = 2\Phi\left(\frac{\delta_1}{\sqrt{N_k - 1}}\right) - 1.$$

Because  $M_k$  is the largest balance of a component,  $\delta_1, \delta_2, \dots, \delta_{N_k}$  are bounded from above by  $\sqrt{M_k}$ . Additionally, w.v.h.p.  $N_k \geq n - \frac{3}{2}k - \mathcal{O}(\sqrt{n} \log n)$ , thus

$$\text{predict}_k \leq 2\Phi\left(\sqrt{\frac{M_k}{n - \frac{3}{2}k - \mathcal{O}(\sqrt{n} \log n)}}\right) - 1.$$

Since  $\Phi(\sqrt{x}/\text{const})$  is a concave function, we can apply expected value, and get

$$\mathbb{E}[\text{predict}_k] \leq 2\Phi\left(\sqrt{\frac{\mathbb{E}[M_k]}{n - \frac{3}{2}k - \mathcal{O}(\sqrt{n} \log n)}}\right) - 1 \sim 2\Phi\left(\sqrt{\frac{\frac{3}{2}k}{n - \frac{3}{2}k}}\right) - 1.$$

Now we are ready to bound the sum from Lemma 4.5. Using the linearity of expectation and inequality  $\min(\frac{1}{6}, \frac{1}{2}x) \geq \frac{1}{6}x$  for  $x \geq 0$  we obtain:

$$\begin{aligned} \mathbb{E}\left[\sum_{k=1}^{2n/3} \min\left(\frac{1}{6}, \frac{1}{2}(1 - \text{predict}_k)\right)\right] &= \sum_{k=1}^{2n/3} \mathbb{E}\left[\min\left(\frac{1}{6}, \frac{1}{2}(1 - \text{predict}_k)\right)\right] \geq \\ &\geq \sum_{k=1}^{2n/3} \frac{1}{6}(1 - \mathbb{E}[\text{predict}_k]) \geq n \cdot \int_0^{2/3} \frac{1}{3} \left(1 - \Phi\left(\sqrt{\frac{\frac{3}{2}x}{1 - \frac{3}{2}x}}\right)\right) dx - o(n). \end{aligned}$$

Finally, we calculate

$$1 + \int_0^{2/3} \frac{1}{3} \left(1 - \Phi\left(\sqrt{\frac{\frac{3}{2}x}{1 - \frac{3}{2}x}}\right)\right) dx \approx 1.0382578.$$

**Theorem 4.7.** *Any algorithm that reports majority exactly requires in expectation at least 1.038n comparisons.*

## Acknowledgments

Most of the work has been done while PU was affiliated to Aalto University, Finland.

## References

- [1] L. Alonso, E. M. Reingold, and R. Schott. Determining the majority. *Inf. Process. Lett.*, 47(5):253–255, 1993.
- [2] L. Alonso, E. M. Reingold, and R. Schott. The average-case complexity of determining the majority. *SIAM J. Comput.*, 26(1):1–14, 1997.
- [3] R. S. Boyer and J. S. Moore. MJRTY: A fast majority vote algorithm. In *Automated Reasoning: Essays in Honor of Woody Bledsoe*, pages 105–118, 1991.
- [4] D. Christofides. On randomized algorithms for the majority problem. *Discrete Applied Mathematics*, 157(7):1481–1485, 2009.
- [5] F. R. K. Chung, R. L. Graham, J. Mao, and A. C. Yao. Oblivious and adaptive strategies for the majority and plurality problems. *Algorithmica*, 48(2):147–157, 2007.
- [6] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25(3):285–287, 1979.
- [7] D. Dor and U. Zwick. Selecting the median. *SIAM J. Comput.*, 28(5):1722–1758, 1999.
- [8] D. Dor and U. Zwick. Median selection requires  $(2 + \epsilon)n$  comparisons. *SIAM J. Discrete Math.*, 14(3):312–325, 2001.
- [9] P. Erdős. On a lemma of Littlewood and Offord. *Bull. Amer. Math. Soc.*, 51(12):898–902, 12 1945.
- [10] M. Fischer and S. Salzberg. Finding a majority among  $n$  votes: solution to problem 81-5. *Journal of Algorithms*, 1982.
- [11] G. D. Marco and A. Pelc. Randomized algorithms for determining the majority on graphs. *Combinatorics, Probability & Computing*, 15(6):823–834, 2006.
- [12] M. Paterson. Progress in selection. In *Algorithm Theory–SWAT’96*, pages 368–379. Springer, 1996.
- [13] M. E. Saks and M. Werman. On computing majority by comparisons. *Combinatorica*, 11(4):383–387, 1991.
- [14] G. Wiener. Search for a majority element. *Journal of Statistical Planning and Inference*, 100(2):313–318, 2002.