Acta Math. Univ. Comenianae
Vol. LXXXVIII, 3 (2019), pp. 601–609

ADAPTIVE MAJORITY PROBLEMS FOR RESTRICTED QUERY GRAPHS AND FOR WEIGHTED SETS

G. DAMÁSDI, D. GERBNER, G. O. H. KATONA, A. METHUKU, B. KESZEGH, D. LENGER, D. T. NAGY, D. PÁLVÖLGYI, B. PATKÓS, M. VIZER AND G. WIENER

ABSTRACT. Suppose that the vertices of a graph G are colored with two colors in an unknown way. The color that occurs on more than half of the vertices is called the *majority color* (if it exists), and any vertex of this color is called a *majority vertex*. We study the problem of finding a majority vertex (or show that none exists), if we can query edges to learn whether their endpoints have the same or different colors. Denote the least number of queries needed in the worst case by $m(G)$. It was shown by Saks and Werman that $m(K_n) = n - b(n)$ where $b(n)$ is the number of 1's in the binary representation of n. In this paper we initiate the study of the problem for general graphs. The obvious bounds for a connected graph G on n vertices are $n-b(n) \leq m(G) \leq n-1$. We show that for any tree T on an even number of vertices we have $m(T) = n - 1$, and that for any tree T on an odd number of vertices, we have $n - 65 \leq m(T) \leq n - 2$. Our proof uses results about the weighted version of the problem for K_n, which may be of independent interest. We also exhibit a sequence G_n of graphs with $m(G_n) = n - b(n)$ such that the number of edges in G_n is $O(nb(n))$.

1. INTRODUCTION

Given a set X of n balls and an unknown coloring of X with a fixed set of colors, we say that a ball $x \in X$ is a *majority ball* if its color class contains more than $|X|/2$ balls. The *majority problem* is to find a majority ball (or show that none exists). In the basic model of majority problems, one is allowed to ask queries of pairs (x, y) of balls in X to which the answer tells whether the color of x and y is the same or not, which we denote by SAME and DIFF, respectively. The answers are given by an *Adversary* whose goal is to force us to use as many questions as possible. It is an easy exercise to see that if the number of colors is two, then in a non-adaptive search (all queries must be asked at once) the minimum number of queries to solve the majority problem is $n - 1$, unless n is odd, in which case $n - 2$ queries suffice.

Received June 6, 2019.
2010 *Mathematics Subject Classification.* Primary 90B40, 05C05.
Research supported by the Lendület program of the Hungarian Academy of Sciences (MTA), under grant number LP2017-19/2017, the János Bolyai Research Fellowship of the Hungarian Academy of Sciences, the National Research, Development and Innovation Office – NKFIH under the grants K 116769, K 124171, SNN 129364 and KH 130371 and by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC).

On the other hand, Fisher and Salzberg [**8**] proved that if we do not have any restriction on the number of colors, $\lceil 3n/2 \rceil - 2$ queries are necessary and sufficient to solve the majority problem adaptively (queries are asked sequentially). If the number of colors is two, then Saks and Werman [**15**] proved that the minimum number of queries needed in an adaptive search is $n-b(n)$, where $b(n)$ is the number of 1's in the binary form of n (we note that there are simpler proofs of this result, see [**1, 13, 16**]). There are several other generalizations of the problem, including more colors [**2, 4, 9, 11**], larger queries [**3, 4, 6, 7, 10, 11, 12**], non-adaptive [**1, 5, 9**], weighted versions [**9**].

In the present paper we study the adaptive majority problem for two colors when we restrict the set of pairs that can be queried to the edges of some graph G on n vertices. The original majority problem, where we can ask any pair, corresponds to $G = K_n$. To distinguish between the version when we are restricted to the edges of a graph, and the original, unrestricted version, we call the colored objects *vertices* and *balls*, respectively. Notice that it is possible to solve the majority problem (with any number of queries) if and only if G is connected when n is even, and if and only if G has at most two components when n is odd. For any such graph, denote the minimum number of queries needed to solve the majority problem in the worst case by $m(G)$. Obviously we have $n - b(n) = m(K_n) \leq m(G) \leq n - 1$ (moreover, $m(G) \leq n - 2$ when n is odd). Our main results are the following.

Theorem 1.1. *For every tree T on an even number n of vertices $m(T) = n-1$ and for every tree T on an odd number n of vertices $m(T) \geq n - 65$.*

The constant 65 is probably far from optimal, it is possible that $m(T) \geq n - 3$ holds for every tree. We have a better lower bound, $n - 6$ for paths.

We also study the least number of edges a graph must have if we can solve the majority problem as fast as in the unrestricted case, i.e., when $m(G) = n - b(n)$.

Theorem 1.2. *For every n, there is a graph G with n vertices and $O(nb(n))$ edges with $m(G) = n - b(n)$.*

It would be interesting to determine whether this bound can be improved to $O(n)$, or show a superlinear lower bound.

The proof of Theorem 1.1 uses a *weighted* version of the original (i.e., $G = K_n$ case of the) majority problem, which is defined in the next section. We think these results are interesting on their own.

In the following, we always suppose that only two colors are used, which we call red and blue. When both colors contain the same number of balls, then we call the coloring *balanced*.

2. Weighted majority problems

We define a variant of the majority problem, where the balls are given different weights. Given k balls with non-negative integer weights $w_1, \ldots, w_k$, a ball i is a (weighted) *majority ball* if the weight of its color class is more than $\sum_{i=1}^{k} w_i/2$. The (weighted) *majority problem* is to find a majority ball (or show that none

exists). Note that during the running of an adaptive algorithm for the majority problem, at any point the information obtained so-far can be represented by a weighted vector $\underline{w}$.[1]

A set of k balls with given weights $w_1, \ldots, w_k$ can be represented by a vector $\underline{w} = (w_1, \ldots, w_k)$. We denote the number of queries needed to solve the weighted majority problem in the worst case by $m(\underline{w})$. With this notation, the result of Saks and Werman for the non-weighted becomes $m(1, \ldots, 1) = k - b(k)$. Note that $m(\underline{w}) \leq k - 1$ and if $\sum_{i=1}^{k} w_i$ is odd, then $m(\underline{w}) \leq k - 2$ (if $k \geq 2$).

The weighted problem was first studied in [**9**], where the following proposition (which also implies the result of Saks and Werman) was proved about the non-weighted variant, generalizing a result of [**13**] (which built on [**14**]). Let $\mu(k)$ denote the largest l such that 2^l divides k (and define $\mu(0) = \infty$). For $\underline{w}$ denote by p the number of balanced colorings, and by p_i the number of (non-balanced) colorings such that w_i is in the majority class.[2]

Proposition 2.1.
(i) $m(\underline{w}) \geq k - \mu(p)$, *and*
(ii) $m(\underline{w}) \geq k - 1 - \mu(p_i)$ *for every* $i \leq k$.

Our main results about the weighted majority problem are exact bounds for some special $\underline{w}$.

Lemma 2.2. *Let* $\underline{w} = (w_1, \ldots, w_k)$ *and* $k > 2^n + 1$.
(i) *If* $w_1 = \cdots = w_{2^n} = 1$ *and* $\sum_{i=1}^{k} w_i = 2^{n+1}$, *then* $m(\underline{w}) = k - 1$.
(ii) *If* $w_1 = \cdots = w_{2^n} = 1$, $\sum_{i=1}^{k} w_i = 2^{n+1} + 1$, *then* $m(\underline{w}) = k - 2$.

Note that $w_k \neq 2^n$ implies $k > 2^n + 1$, as $w_i \neq 0$.
We can also generalize this to the following.

Lemma 2.3. *Let* $\underline{w} = (w_1, \ldots, w_k)$ *and* $k > 2^n + 1$.
(i) *If* $w_1 = \cdots = w_{2^n}$ *and there are an odd number of partitions* $R \cup B = \{2^n + 1, \ldots, k\}$ *such that* $\sum_{i \in B} w_i - \sum_{i \in R} w_i = w_1 2^n$, *then* $m(\underline{w}) = k - 1$.
(ii) *If* $w_1 = \cdots = w_{2^n}$ *and* $\sum_{i=1}^{k} w_i \leq w_1 2^{n+1} + 2w_j$ *for any* $2^n < j \leq k$, *with the inequality being strict for* $j = k$, *then* $m(\underline{w}) \geq k - 2$.

These imply, for example, that $m(3, 3, 7, 8, 9) = 4$ and $m(3, 3, 5, 5, 5) \geq 3$.

Call a vector $\underline{w} = (w_1, \ldots, w_k)$ *hard* if $m(\underline{w}) = k - 1$ and $\sum_{i=1}^{n} w_i$ is even, or $\sum_{i=1}^{n} w_i$ is odd and $m(\underline{w}) = k - 2$. Thus Lemma 2.2 states that the vectors satisfying its conditions are hard.

Observation 2.4. If $\underline{w} = (w_1, \ldots, w_k)$ is hard with $w_1 = w_2$ and $\underline{w}' = (2w_1, w_3, \ldots, w_k)$, then $\underline{w}'$ is also hard.

Combining Observation 2.4 with Lemma 2.2, we obtain the following statement.

[1] This is explained in more details in Section 3.

[2] Beware that in [**9**] a slightly different notation was used, where p denoted the number of balanced 2-partitions, which is half of the number of balanced colorings, and part (ii) of Proposition 2.1 was not explicitly stated.

Lemma 2.5. *If $w_1, \dots, w_j$ are each powers of two, $k > 2^n + 1$, and*

(i) $\sum_{i=1}^{j} w_i = 2^n$ *and* $\sum_{i=1}^{k} w_i = 2^{n+1}$, *then* $\underline{w}$ *is hard, i.e.,* $m(\underline{w}) = k - 1$.

(ii) $\sum_{i=1}^{j} w_i = 2^n$, $\sum_{i=1}^{k} w_i = 2^{n+1} + 1$, *then* $\underline{w}$ *is hard, i.e.,* $m(\underline{w}) = k - 2$.

For larger weights, we state the following weaker statement.

Corollary 2.6. *If $w_1, \dots, w_j$ are each powers of two and*

$\sum_{i=1}^{j} w_i = 2^n$, $w_{j+1} = 1$, $\sum_{i=1}^{k} w_i = 2^{n+1}+3$ *and* $k > 2^n+2$, *then* $m(\underline{w}) \geq k-3$.

Combining Lemma 2.3 and Corollary 2.6 we obtain the following.

Proposition 2.7. *If $1 \leq w_1, \dots, w_j \leq 2$ and*

$\sum_{i=1}^{j} w_i = 2^n$, $\sum_{i=1}^{k} w_i = 2^{n+1} + 3$ *and* $k > 2^n + 2$, *then* $m(\underline{w}) \geq k - 3$.

Proof. If there is some $i > j$ such that $w_i = 1$, we are done using Corollary 2.6. Otherwise, for all $j < i \leq k$ we have $w_i \geq 2$, and we can apply Lemma 2.3. □

3. Graphs

When we query the edges of a graph G, we call a maximal subset of vertices connected by already asked queries a *q-component*. The weight w of a q-component X is the number of its vertices. For a ball $v \in X$, let $w(v) := w(X)$. If a q-component X has weight zero, we say that X is *balanced*. A graph G on n vertices is *hard* if $m(G) = n - 1$ for even n and $m(G) = n - 2$ for odd n.

Proposition 3.1. *Every tree T on an even number n of vertices is hard, i.e., $m(T) = n - 1$.*

Surprisingly, it is much harder to give a lower bound for trees on an odd number of vertices. For paths, for example, we have $m(P_n) = n - b(n)$ for all odd $n \leq 13$, while $m(P_{15}) = 12 = n - b(n) + 1 = n - 3$. (This we have verified with a computer program.)

We start with a lemma that gives another proof for Proposition 3.1. First, we introduce a notation. In a graph G, for a subset of its vertices $X \subset V$ we denote by $\delta(X)$ the *parity* of the number of edges between X and $V \setminus X$. If G is a tree and X is a connected subset of vertices, then $\delta(X)$ equals the parity of the number of components of $V \setminus X$.

Lemma 3.2. *We can answer to queries in any graph G such that for the weight $w(X)$ of any q-component $X \subsetneq V$ we have*

(i) $w(X) = 1$ *if* $|X|$ *is odd, and*

(ii) $w(X) = 2\delta(X)$ *if* $|X|$ *is even.*

Proof of Lemma 3.2. Initially the conditions are satisfied. Suppose that the query is between two q-components, X and Y. If $|X| + |Y|$ is odd, then exactly one of $w(X)$ and $w(Y)$ equals 1, while the other equals 0 or 2, so we can achieve $w(X \cup Y) = 1$ to satisfy condition (i). If $|X|$ and $|Y|$ are both odd, then we can choose the weight of $X \cup Y$ to be 0 or 2; one of those is equal to $2\delta(X)$. If

$|X|$ and $|Y|$ are both even, then since $\delta(X \cup Y) = \delta(X) + \delta(Y) - 2|E(X,Y)| = \delta(X) + \delta(Y) \bmod 2$, we have $w(X \cup Y) = w(X) + w(Y) \bmod 4$. Observe that $w(X) + w(Y)$ is 0, 2 or 4. Thus we can answer so that $w(X \cup Y)$ becomes 0, 2 or 0, respectively, to satisfy condition (ii). □

For the lower bound of $n - 65$ for trees, we need another ingredient. Before the main proof, we prove a simpler result that contains this ingredient of the proof, and is of independent interest.

Theorem 3.3. *Let $n = 2^k + l$, where $l < 2^k$. If G has a set U of vertices such that $|U| \leq 2^{k-2}$ and the components of $G \setminus U$ are single vertices (i.e., every edge is incident to a vertex in U), then G is hard, i.e., $m(G) = n - 1$ if n is even and $m(G) = n - 2$ if n is odd.*

Proof. Denoting by $w(X)$ the weight of a q-component X, we initially have $\sum_X w(X) = n$. Adversary will maintain in the first part of the algorithm that $w(X) \neq 0$ for every q-component X.

Let us now describe a strategy of the Adversary for the first part of the algorithm. Whenever we compare some $v \in G \setminus U$ with a $u \in U$ such that $w(u) \geq 2$, the answer is such that the weight of the new q-component is $w(\{u, v\}) = w(u) - 1$, thus $\sum_X w(X)$ decreases by 2. In every other case the answer is such that the weights are added up, i.e., $\sum_X w(X)$ remains the same.

Introduce the potential function $\Psi = \sum_X w(X) + |\{X \mid X \cap U \neq \emptyset,\, w(X) = 1\}|$. The Adversary's strategy is such that every time we compare some $v \in G \setminus U$ with a $u \in U$, the function Ψ decreases by at least 1. Since initially $\Psi = n + |U|$, after $|U| + l$ queries involving some vertex of $V \setminus U$, we would have $2^k \geq \Psi \geq \sum_X w(X)$. But Adversary stops executing this algorithm the moment we have $\sum_X w(X) = 2^k$ or $\sum_X w(X) = 2^k + 1$; this surely happens, as $\sum_X w(X)$ can only decrease by 2.

Let us consider the vertices from $G \setminus U$ that were merged into some q-components (i.e. those that appeared in queries). Let x denote the number of those where the total weight did not decrease when they first appeared in a query, and y denote the number of those where the total weight did not decrease when they first appeared in a query. Then we have $x \leq y + |U|$. Indeed, consider a q-component containing a vertex $u \in U$, and observe that whenever the weight of this component increased by merging it with a vertex from $G \setminus U$, the next time its weight decreased.

This implies that at the point where Adversary stops executing the algorithm, the number of vertices in $G \setminus U$ that have not appeared in any query is at least $n - |U| - (|U| + l) \geq 2^{k-1}$. We are done by applying Lemma 2.2 to the current q-components as weighted balls. (So even if we could compare any two vertices from now, we still could not solve the majority problem with less queries.) □

With a similar method, we can obtain the following lower bound for odd paths.

Theorem 3.4. $m(P_n) \geq n - 6$.
Moreover, $m(P_n) \geq n - 5$ unless $n + 1$ or $n + 3$ is a power of two.

Proof. We have already seen that this holds if n is even, so it is enough to prove the theorem for n odd. First we prove the weaker claim $m(P_n) \geq n - 10$. The statement holds for $n < 1000$ as $m(P_n) \geq n - b(n)$. Let U include every 9^{th} vertex of P_n, starting with the first, and also the last vertex of P_n, so $\lceil \frac{n}{9} \rceil \leq |U| \leq \lceil \frac{n}{9} \rceil + 1$, and $P_n \setminus U$ consists of paths on 8 vertices (and possibly one shorter path at the end). We answer each query such that for any q-component X if $X \cap U = \emptyset$, then $w(X) \leq 1$, while if $X \cap U \neq \emptyset$, then $1 \leq w(X) \leq 2$. In each step the total weight decreases by 0 or 2, so after a while it becomes $2^k + 1$ for $k = \lfloor \log n \rfloor$. When this happens, we apply Lemma 2.2 to the current q-components as weighted balls. Indeed, $\sum_{X:X\cap U\neq\emptyset} w(X) \leq 2|U| = 2\lceil \frac{n}{9} \rceil + 2 \leq \frac{n}{4} \leq 2^{k-1}$ if $n > 1000$, so $\sum_{X:X\cap U=\emptyset} 1 \geq 2^{k-1}$. By Lemma 2.2 the number of queries needed to finish is at least the number of components minus 2, depending on the parity. Therefore, we need to connect all of U into at most two components. That means that there can be at most one path of length 9 (between two vertices from U) whose edges we have not queried. This proves $m(P_n) \geq n - 10$.

But we can do even better, because out of the 9 edges of the path at least 4 must be queried if the path contains no non-balanced components. This proves $m(P_n) \geq n - 6$ if n is large enough, but now we have to be more careful with the calculations. Because of this, we also change how we select U; instead of starting with the first vertex, we start with the second vertex of the path, then take every 9^{th} vertex, and finally the last but one vertex. We can afford to skip the endvertices, as a single vertex anyhow cannot form a balanced component, we can only compare it to its adjacent vertex from U. This gives $|U| = \lfloor \frac{n+14}{9} \rfloor$, and $\frac{n+14}{9} \leq \frac{n}{8}$ if $n \geq 112$, while for $n < 127$ the lower bound $n - b(n) \geq n - 6$ holds.

The proof of the moreover part is similar, except that after we start with the second vertex, we take every 8^{th} vertex, and finally the last but one vertex. This way only 4 edges can remain unqueried between two different components. This gives $|U| = \lfloor \frac{n+12}{8} \rfloor < 2^{\lfloor \log_2 n \rfloor - 2}$ unless $n + 1$ or $n + 3$ is a power of two. □

The lower bound for general trees, Theorem 3.6, is based on a similar idea as that of Theorem 3.3, but also combines ideas from Theorem 3.4 and uses Proposition 2.7. We also need the following version of the folklore generalization of the concept of centroid for trees, known as *centroid decomposition*.

Proposition 3.5. *In every tree on n vertices there is a subset of at most $2n/p$ vertices U such that every component of $G \setminus U$ has at most p edges (including the edges from the components to U).*

Theorem 3.6. *If G is a tree on n vertices, then $m(G) \geq n - 65$.*

Proof. Let $n = 2^k + l$, where $l < 2^k$ is odd. Apply Proposition 3.5 with $p = 32$ to obtain a set U of vertices such that $|U| \leq 2^{k-3} - 1$ and each component T has at most p edges. (We write p instead of 32 throughout the proof.)

We proceed as in the proof of Theorem 3.3. We denote by $w(X)$ the weight of a q-component X, and for a vertex u of G, $w(u)$ denotes the weight of the q-component containing u. We initially have $\sum_X w(X) = n$. The Adversary will

maintain in the first part of the algorithm that $w(X) \neq 0$ for every q-component X that intersects U.

We split each component T to a *connecting* part T' and some *hanging* parts $T_1, T_2, \ldots$ where any of these can be empty, as follows. If $v \in T$ separates some vertices of U from each other, then it goes to T'. Each connected component of $T \setminus T'$ forms a different T_i. Notice that each hanging part T_i has a unique vertex $r(T_i)$ that separates $T_i \setminus \{r(T_i)\}$ from $T \setminus T_i$; we call $r(T_i)$ the *root* of T_i.

We answer queries inside T_i according to Lemma 3.2 (applied only to T_i), while if the query $X \cap T' \neq \emptyset$, we answer such that $w(X) \leq 2$ (which is similar to Theorem 3.4). This way the weight of any $X \subset G \setminus U$ will be at most 2. The crucial property is that the balanced q-components of T will always separate either two U vertices, or some positive weight part of a T_i from a U vertex. This way they are "in the way" to compare these parts with the rest of the graph, so they cannot be simply ignored. The strategy of the Adversary will be to make sure that the game cannot end while there are many unbalanced q-components. After there are only few unbalanced q-components the game might end, but in this case the graph could be made into a single q-component by adding $O(p)$ further edges to it. This shows that at most these many queries can be saved.

Also, in case we merge all of some T_i into one q-component, Adversary would like to avoid $w(T_i) = 0$. This cannot happen if T_i has an odd number of vertices; if T_i has an even number of vertices, Adversary adds an (imaginary) extra degree one vertex $r'(T_i)$ to T_i that is adjacent only to $r(T_i)$, to obtain T_i^*, and applies Lemma 3.2 to T_i^* instead of T_i. Since $r'(T_i)$ is never compared with anything, merging all of T_i into a q-component cannot give $w(T_i) = 0$, because $T' = T_i^* \setminus T_i$ has only one component, $\{r'(T_i)\}$. Therefore, in case the whole tree T_i is merged, we get $w(T_i) = 2$.

Whenever we compare some $Y \subset G \setminus U$ with an X intersecting U such that $w(X) \geq 3$, Adversary answers such that the weight of the new q-component is $w(X) - w(Y)$, thus $\sum_X w(X)$ decreases by $2w(Y) \leq 4$. In every other case Adversary answers so that the weights are added up, i.e., $\sum_X w(X)$ remains the same. This way the weight of a q-component can never exceed 4, unless we merge two q-components that both intersect U. Because of this, we can conclude that $\sum_{X : X \cap U \neq \emptyset} w(X) \leq 4|U| \leq n/4 < 2^{k-1}$.

Adversary stops executing this algorithm the moment we have $\sum_X w(X) = 2^k + 1$ or $2^k + 3$; this surely happens, as $\sum_X w(X)$ is odd and can decrease by at most 4. As we have seen in the earlier proofs, if $\sum_X w(X) = 2^k + 1$, then we will have two non-balanced q-components when the algorithm is done. If $\sum_X w(X) = 2^k + 3$, then we can apply Proposition 2.7, whose conditions are shaped to work here, to conclude that we will have at most three non-balanced q-components when the algorithm is done.

Moreover, these few remaining non-balanced q-components need to cover U, as the weights of sets intersecting U stays positive throughout the algorithm. If at the end we have at most ℓ components, then adding $\ell - 1$ original tree component

T's, we can make the q-graph connected. As every tree has at most p vertices, and in our case $\ell \leq 3$, adding $2p$ edges can make the q-graph connected.

To summarize, instead of asking all $n-1$ edges, we might save $2p = 64$. □

Remark. We could get a better constant by considering the number of yet unqueried edges we need to add to connect the remaining non-balanced q-components.

References

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

2. Aigner M., De Marco G. and Montangero M., *The plurality problem with three colors and more*, Theoret. Comput. Sci. **337** (2005), 319–330.

3. Borzyszkowski A. M., *Computing majority via multiple queries*, Theoret. Comput. Sci. **539** (2014), 106–111.

4. Chang H., Gerbner D. and Patkós B., *Finding non-minority balls with majority and plurality queries*, preprint.

5. Chung F., Graham R., Mao J. and Yao A., *Oblivious and Adaptive Strategies for the Majority and Plurality Problems*, in: Computing and combinatorics, Lecture Notes in Comput. Sci. 3595, Springer, 2005, 329–338.

6. De Marco G., Kranakis E. and Wiener G., *Computing majority with triple queries*, Theoret. Comput. Sci. **461** (2012), 17–26.

7. Eppstein D. and Hirschberg D. S., *From discrepancy to majority*, Algorithmica **80** (2018), 1278–1297.

8. Fisher M. J.and Salzberg S. L., *Finding a majority among n votes*, J. Algorithms **3** (1982), 375–379.

9. Gerbner D., Katona G. O. H., Pálvölgyi D. and Patkós B., *Majority and plurality problems*, Discrete Appl. Math. **161** (2013), 813–818.

10. Gerbner D., Keszegh B., Pálvölgyi D., Patkós B., Vizer M. and Wiener G., *Finding a non-minority ball with majority answers*, Discrete Appl. Math. **219** (2017), 18–31.

11. Gerbner D., Lenger D., Vizer M., *A plurality problem with three colors and query size three*, preprint.

12. Gerbner D. and Vizer M., *Majority problems of large query size*, Discrete Appl. Math. **254** (2019), 124–134.

13. Hayes T. P., Kutin S. and van Melkebeek D., *On the quantum black-box Complexity of Majority*, Algorithmica **34** (2002), 480–501.

14. Rivest R. and Vuillemin J., *On recognizing graph properties from adjacency matrices*, Theoret. Comput. Sci. **3** (1976), 371–384.

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

16. Wiener G., *Search for a majority element*, J. Statist. Plann. Inference **100** (2002) 313–318.

G. Damásdi, MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University, Budapest, Hungary,
e-mail: damasdigabor@caesar.elte.hu

D. Gerbner, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary,
e-mail: gerbner@renyi.hu

G. O. H. Katona, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary,
e-mail: katona@renyi.hu

A. Methuku, Department of Mathematics, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland,
e-mail: abhishekmethuku@gmail.com

B. Keszegh, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences
MTA-ELTE Lendület Combinatorial Geometry Research Group, Budapest, Hungary,
e-mail: keszegh.balazs@renyi.mta.hu

D. Lenger, MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University, Budapest, Hungary,
e-mail: lengerd@cs.elte.hu

D. T. Nagy, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary,
e-mail: nagydani@renyi.hu

D. Pálvölgyi, MTA-ELTE Lendület Combinatorial Geometry Research Group, Institute of Mathematics, Eötvös Loránd University, Budapest, Hungary,
e-mail: dom@cs.elte.hu

B. Patkós, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary,
e-mail: patkos@renyi.hu

M. Vizer, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, Budapest, Hungary,
e-mail: vizermate@gmail.com

G. Wiener, Dept. of Computer Science and Information Theory, Budapest University of Technology and Economics, Budapest, Hungary,
e-mail: wiener@cs.bme.hu