The query complexity of locating monochromatic matchings and trees

Joseph Briggs∗ Christopher Cox†

April 2, 2019

Abstract

We give new proofs of known Ramsey numbers relating to matchings and trees which yield efficient algorithms to locate these structures. Consider a two-player game between players Builder and Painter. Painter begins the game by picking a coloring of the edges of $K_n$ which is hidden from Builder. In each round, Builder points to an edge and Painter reveals its color. Builder’s goal is to locate a particular monochromatic structure in Painter’s coloring by revealing the color of as few edges as possible. In this paper, we consider the situation where this “particular monochromatic structure” is a large matching or a large tree. We show that in any $t$-coloring of $E(K_n)$, Builder can locate a monochromatic matching on at least $n - t + 1$ edges by revealing at most $O(n \log t)$ edges. We show also that in any 3-coloring of $E(K_n)$, Builder can locate a monochromatic tree on at least $n/2$ vertices by revealing at most $5n$ edges.

1 Introduction

In this note, we give new, algorithmic proofs of the following three Ramsey-type statements about matchings and trees. Let $K_n$ denote the complete graph on $n$ vertices and $rK_2$ denote the matching on $r$ edges.

Theorem 1 (Cockayne and Lorimer [2]). Suppose that $r_1 \geq \cdots \geq r_t$ are positive integers and that $n \geq r_1 + 1 + \sum_{i=1}^{t} (r_i - 1)$. In any $t$-coloring of $E(K_n)$, there is a copy of $r_i K_2$ in color $i$ for some $i \in [t]$.

Theorem 2 (Folklore). In any 2-coloring of $E(K_n)$, there is a monochromatic spanning tree.

Theorem 3 (Gerencsér and Gyárfás [4]). Define $k(n) := \frac{n}{2} + 1$ if $n \equiv 2 \pmod{4}$ and $k(n) := \left\lceil \frac{n}{2} \right\rceil$ otherwise. For $n \geq 3$, in any 3-coloring of $E(K_n)$, there is a monochromatic tree on at least $k(n)$ vertices.

There are colorings of $E(K_n)$ showing that the bounds in Theorems 1 and 3 are tight. We will recall and use these special colorings in the proofs of Theorems 1″ and 3″.

We will reprove the above theorems in a way that not only shows the existence of these structures, but yields efficient algorithms to locate them. We measure the efficiency of our algorithms in terms of query complexity, which can be viewed as a Builder–Painter game. This is a two-player game between players Builder and Painter. Painter begins the game by picking a coloring of the edges of $K_n$ which is hidden from Builder. In each round, Builder points to an edge of $K_n$ and

∗Department of Mathematical Sciences, Technion Institute of Israel, Haifa, Israel. briggs@campus.technion.ac.il.
†Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA. cocox@andrew.cmu.edu.
Painter reveals the color of that edge. Builder’s goal is to either locate a particular monochromatic structure in Painter’s coloring or determine that such a structure does not exist by revealing the colors of as few edges as possible, while Painter’s goal is to hide this information from Builder for as long as possible.

The Builder–Painter game described above is similar to the Builder–Painter game for online Ramsey numbers, which were independently introduced by Beck [1] and Kurek and Ruciński [6]. The main difference between the two is that the Builder–Painter game for online Ramsey numbers does not have a bound the total number of vertices in play.

Builder’s questions are called queries, and edges that Builder has queried previously in the game are called exposed.

The results in this paper give bounds on the query complexity of locating the structures in Theorems 1, 2 and 3. Here and throughout this paper, we use \( \lg = \log_2 \) and employ the convention that \( \lg 0 = 0 \) so that our results can be stated uniformly.

Firstly, for matchings, in Section 2, we show:

**Theorem 1’.** Let \( t \geq 2 \), let \( r_1 \geq r_2 \geq \cdots \geq r_t \) be positive integers and let \( n \geq r_1 + 1 + \sum_{i=1}^t (r_i - 1) \). Given any coloring \( \chi : E(K_n) \to [t] \), Builder can locate a copy of \( r_i K_2 \) in color \( i \) for some \( i \in [t] \) using at most \( \frac{1}{t+1} (2t - 1 + (t - 3) \lg(t - 2)) n \) queries.

In particular, \( n \) queries suffice for two colors, \( \frac{2}{3} n \) queries suffice for three colors, and in general, \( (2 + \lg(t - 2)) n \) queries suffice for \( t \) colors.

The above theorem is tight in the following sense.

**Theorem 1”.** For \( t, r \geq 2 \) and \( n = (t + 1)r - t \), Painter has a strategy for \( t \)-coloring \( E(K_n) \) that requires Builder to query all \( \binom{n}{2} \) edges in order to either find a monochromatic matching of size \( r \) or determine that no such matching exists.

Furthermore, even if Painter’s coloring is required to contain a monochromatic \( r K_2 \), Painter has a strategy for \( t \)-coloring \( E(K_n) \) which requires Builder to query \( \Omega(n^2) \) edges to determine which color has this property.

Note that in the latter part of the above theorem, Builder must only determine which of the \( t \) colors contains a matching of size \( r \) and does not actually have to locate said matching. In other words, Builder may determine that, say, color \( t \) contains this matching by somehow demonstrating that colors \( 1, \ldots, t - 1 \) cannot have a matching of the required size.

Turning now to trees, the following result was communicated to us by Micek and Pegden.

**Theorem 2’** (Micek and Pegden [7]). For \( n \geq 2 \), in any 2-coloring of \( E(K_n) \), Builder can locate a monochromatic spanning tree using at most \( 2n - 3 \) queries. This is tight in that Painter has a strategy for 2-coloring \( E(K_n) \) which requires Builder to use at least \( 2n - 3 \) queries to determine which color contains a spanning tree.

Since the above result is unpublished, we present a proof in Section 3.1 for completeness. We extend this result to 3-colorings in Section 3.2 by showing:

**Theorem 3’.** For \( n \geq 3 \), in any 3-coloring of \( E(K_n) \), Builder can locate a monochromatic tree on \( k(n) \) vertices using at most \( 5(n - 1) \) queries.

Again, this theorem is tight in the following sense.

**Theorem 3”.** For \( n \geq 3 \), even if Painter’s coloring is required to contain a monochromatic tree on at least \( k(n) + 1 \) vertices, Painter has a strategy for 3-coloring \( E(K_n) \) that requires Builder to query \( \Omega(n^2) \) edges to determine which color contains this tree.
The proofs of Theorems 1′, 2′ and 3′ do not rely on Theorems 1, 2 and 3, and thus provide self-contained proofs.

2 Monochromatic matchings

Builder’s strategy. We begin with the key lemma which motivates Builder’s strategy. Throughout this paper, a forest is assumed to have no isolated vertices, i.e. every connected component is a tree with at least one edge. For a forest \( F \), we denote the set of connected components of \( F \) by \( \text{comp}(F) \).

**Definition 4.** Let \( F \) be a forest and let \( \chi: E(F) \to [t] \). \( F \) is said to be a good forest (with respect to \( \chi \)) if
- \( \chi \) is a proper edge-coloring of \( F \), and
- there is some color \( c \in [t] \) such that every component of \( F \) contains an edge of color \( c \).

**Lemma 5.** Fix \( t \geq 2 \) and let \( r_1, \ldots, r_t \) be positive integers. Suppose that \( F \) is a forest and \( \chi: E(F) \to [t] \) is a \( t \)-coloring. If \( F \) is a good forest with respect to \( \chi \) and \( |V(F)| \geq \max_i r_i + \sum_i (r_i - 1) \), then, for some \( i \in [t] \), \( F \) contains a matching of size \( r_i \) in color \( i \).

**Proof.** Denote by \( m_i \) the number of edges of color \( i \) in \( F \), so \( \sum_i m_i = e(F) \). Since \( F \) is a good forest, \( \chi \) is a proper coloring of \( E(F) \), so the largest matching in color \( i \) in \( F \) has size precisely \( m_i \). Hence, we need only show that \( m_i \geq r_i \) for some \( i \in [t] \).

By assumption, there is some color \( c \in [t] \) which appears in each component of \( F \), so \( m_c \geq |\text{comp}(F)| \). Therefore,

\[
m_c + \sum_i m_i \geq |\text{comp}(F)| + e(F) = |V(F)| \geq \max_i r_i + \sum_i (r_i - 1),
\]

so the claim follows from the pigeonhole principle. \( \square \)

With this in mind, Builder’s strategy is to locate a good forest in Painter’s coloring which covers all but at most one vertex. The following lemma presents the main tool employed by Builder to accomplish this.

**Lemma 6.** Let \( \chi: E(K_n) \to [t] \), and suppose that Builder has exposed all edges of some tree \( T \subseteq E(K_n) \) on \( m \geq 1 \) edges. Suppose \( T \) is properly edge-colored under \( \chi \) and that \( xy \in E(K_n) \) is an exposed edge completely disjoint from \( T \) with \( \chi(xy) \notin \chi(T) := \{\chi(f) : f \in E(T)\} \).

There exists a procedure \( \text{TreeExtend}(\chi, T, xy) \) that, by querying at most \( 1 + \lceil \lg (\text{diam}(T) - 1) \rceil \) extra edges, returns a tree \( T^* \) with the following properties:

1. \( V(T^*) \subseteq V(T) \cup \{x, y\} \),
2. \( e(T^*) \in \{m + 1, m + 2\} \),
3. \( \chi(T^*) \supseteq \chi(T) \cup \{\chi(xy)\} \), and
4. \( T^* \) is properly edge-colored under \( \chi \).

**Proof.** We first define \( \text{TreeExtend} \) and then prove the claimed properties.

Recall that a vertex \( v \) of a tree \( T \) is called a center if \( v \) is at a distance at most \( \lceil \text{diam}(T)/2 \rceil \) from every other vertex of \( T \), where \( \text{diam}(T) \) is the diameter of \( T \). Note that there will be one such vertex when \( \text{diam}(T) \) is even, and two such vertices if \( \text{diam}(T) \) is odd. We denote the center vertex of \( T \) by \( \text{center}(T) \), where an arbitrary choice is made if there are two such vertices. Additionally,
for an edge $xy \in E(T)$, define $T(x, y)$ to be the subtree of $T$ which is formed by rooting $T$ at $y$ and removing all descendants of $x$.  

1: procedure TreeExtend($\chi, T, xy$) 
2: Fix any proper 2-coloring $\eta: V(T) \rightarrow \{x, y\}$ of $T$ $\triangleright$ For $v \in V(T)$, write $\eta_v = \eta(v)$
3: $T' \leftarrow T$
4: $v \leftarrow \text{center}(T)$
5: loop
6: Query the edge $v\eta_v$
7: if $\chi(v\eta_v) = \chi(xy)$ then
8: return $T + \eta_v$
9: else if $T + \eta_v$ is properly colored then
10: return $T + \eta_v + xy$
11: else
12: There is some edge $vv' \in E(T')$ with $\chi(vv') = \chi(v\eta_v)$
13: if $v'$ is a leaf of $T$ then
14: return $T - vv' + \eta_v + xy$
15: else if $v'$ is a leaf of $T'$ then
16: return $T - vv' + \eta_v + xy + v'\eta_v$
17: else
18: $T' \leftarrow T'(v, v')$
19: $v \leftarrow \text{center}(T'(v, v'))$
20: end if
21: end if
22: end loop
23: end procedure

We show first that TreeExtend($\chi, T, xy$) does in fact return $T^*$ and bound the number of queries made in the process.

A new edge is queried only when reaching the beginning of the loop. If $\text{diam}(T') \in \{1, 2\}$, then the procedure will return $T^*$ before reaching Line 17, thus requiring only one additional query. Furthermore, if we reach Line 17, then we will have $\text{diam}(T'(v, v')) \leq \left\lceil \frac{\text{diam}(T)}{2} \right\rceil$ since $v = \text{center}(T')$. From this we conclude that TreeExtend($\chi, T, e$) returns $T^*$ by querying at most $1 + \left\lceil \log(\text{diam}(T) - 1) \right\rceil$ extra edges, recalling that $\log 0 = 0$.

We now verify the claimed properties of $T^*$. Set $c = \chi(xy)$ and consider the four situations in which TreeExtend($\chi, T, xy$) can return $T^*$.

- $T^*$ is returned on Line 8. Here $V(T^*) = V(T) \cup \{\eta_v\} \subseteq V(T) \cup \{x, y\}$ and $c(T^*) = m + 1$. Additionally, $\chi(v\eta_v) = c$, so since $c \notin \chi(T)$, we know that $T^*$ is properly edge-colored and $\chi(T^*) = \chi(T) \cup \{c\}$. Finally, $T^*$ is in fact a tree since adding the edge $v\eta_v$ does not create a cycle.

- $T^*$ is returned on Line 10. Here $V(T^*) = V(T) \cup \{x, y\}$ and $c(T^*) = m + 2$. Additionally, we know that $\chi(v\eta_v) \neq c$, so since $T + v\eta_v$ is properly edge-colored, we know that $T^*$ is also properly edge-colored and $\chi(T^*) = \chi(T) \cup \{\chi(v\eta_v), c\} \supseteq \chi(T) \cup \{c\}$. Finally, $T^*$ is a tree since we do not create a cycle upon adding the edges $v\eta_v$ and $xy$.

- $T^*$ is returned on Line 14. Here we have $V(T^*) = (V(T) \setminus \{v'\}) \cup \{x, y\}$ and $c(T^*) = m + 1$. Now, since we have $\chi(vv') = \chi(v\eta_v)$ for some $vv' \in E(T')$, we know that, since $T'$ is a subtree of $T$, $\chi(vv') = \chi(v\eta_v) \neq c$ since $c \notin \chi(T)$. Thus, $T^*$ is properly edge-colored since we removed the edge $vv'$, and also $\chi(T^*) = \chi(T) \cup \{c\}$ since $\chi(v\eta_v) = \chi(vv')$. Finally, $T^*$ is
a tree since $v'$ is a leaf of $T$, so $T - vv'$ is still at tree, and then adding the edges $v\eta_v$ and $xy$ do not create a cycle.

- $T^*$ is returned on Line 16. Here we have $V(T^*) = V(T) \cup \{x, y\}$ and $e(T^*) = m + 2$. Now, since $v'$ is a leaf of $T'$ but not a leaf of $T$, this means that we must have previously queried the edge $v'\eta_v$ and found that $\chi(v'\eta_v) = \chi(vv')$. Since $\eta$ is a proper 2-coloring of $V(T)$ and $vv' \in E(T)$, we must have $\eta_v \neq \eta_v$, so $T^*$ is indeed properly edge-colored, and $\chi(T^*) = \chi(T) \cup \{e\}$. Finally, $T^*$ is again a tree since we added the path $v\eta_v, \eta_v v'$ and removed the edge $vv'$.

We state a consequence for later reference.

**Corollary 7.** Let $\chi, T, xy$ be as in Lemma 6 and set $T^* = \text{TreeExtend}(\chi, T, xy)$.

- If $e(T) = 1$, then $T^*$ either has $|\chi(T^*)| = 2$ and is a path with 2 edges, or has $|\chi(T^*)| = 3$ and is a path with 3 edges.
- If $e(T) = 2$, then $T^*$ either has $|\chi(T^*)| \geq 4$, or has $|\chi(T^*)| = 3$ and is a star with 3 edges.

We now have all of the necessary tools to describe Builder’s full strategy and prove Theorem 1'.

**Proof of Theorem 1’.** Builder maintains and grows a good forest $F$. While there are still at least 2 vertices $x, y$ uncovered by $F$, Builder queries the edge $xy$. If its color $\chi(xy)$ is already present among all components of $F$, Builder adds this edge (as a 2-vertex component) to $F$, and repeats. Otherwise, there is some connected component $T \in \text{comp}(F)$ which does not have an edge of color $\chi(xy)$.

Here, Builder uses $\text{TreeExtend}(\chi, T, xy)$ to return a tree $T^*$ and replaces $F$ by $F - T + T^*$. By Lemma 6, $F - T + T^*$ is also a good forest and covers at least one more vertex than $F$, so this process must eventually terminate. Furthermore, the process terminates when $F$ covers all but at most one vertex of $K_n$, and thus, by Lemma 5, Builder has located a copy of $r_iK_2$ in color $i$ for some $i \in [t]$. Let $F^*$ be the forest found by Builder.

We now count the total number of queries used to build $F^*$. For integers $m, k \geq 2$, define $q(m, k) := 2k - 1 + (k - 3)\lg(m - 2)$ and define also $q(1, 1) := 1$. For $T \in \text{comp}(F^*)$, let $q(T)$ denote the number of queries used by Builder to construct $T$.

**Claim 8.** If $e(T) = m$ and $|\chi(T)| = k$, then $q(T) \leq q(m, k)$.

**Proof.** If $k = 1$, then also $e(T) = 1$, so certainly $q(T) = 1 = q(1, 1)$. Thus suppose $k \geq 2$.

Since $\text{TreeExtend}$ always appends at least one new color to a tree, we see that, for some $\ell \leq k$, there were trees $T_1, \ldots, T_\ell$ and edges $e_1, \ldots, e_\ell$ with $T_1$ being a single edge, $T_\ell = T$ and $T_{i+1} = \text{TreeExtend}(\chi, T_i, e_i)$ for all $i \in [\ell - 1]$. Certainly $q(T_1) = 1$, and, by Lemma 6, if $d_i := \text{diam}(T_i)$, then

$$q(T_{i+1}) \leq q(T_i) + 1 + (1 + \left\lfloor \lg(d_i - 1) \right\rfloor),$$

where the extra +1 comes from querying the edge $e_i$. Therefore,

$$q(T) \leq 1 + \sum_{i=1}^{\ell-1} \left( 2 + \left\lfloor \lg(d_i - 1) \right\rfloor \right) = 2\ell - 1 + \sum_{i=2}^{\ell-1} \left\lfloor \lg(d_i - 1) \right\rfloor.$$

By Lemma 6, we know that $e(T_{i+1}) \in \{e(T_i) + 1, e(T_i) + 2\}$, so certainly $d_i \leq e(T_i) \leq m - 1$ for all $i \in [\ell - 1]$.

We now break into two cases based on $T_2$:
• $|\chi(T_2)| = 2$: Since $T_2 = \text{TreeExtend}(\chi, T_1, e_1)$, we know that $T_2$ must be a path on 2 edges by Corollary 7; thus $d_2 = 2$. As such,

$$q(T) \leq 2\ell - 1 + \sum_{i=3}^{\ell-1} \lfloor \log(d_i - 1) \rfloor \leq 2k - 1 + (k - 3) \log(m - 2) = q(m, k).$$

• $|\chi(T_2)| \geq 3$: Here, again by Corollary 7, we must actually have $|\chi(T_2)| = 3$ and $T_2$ is a path on 3 edges, so $d_2 = 3$. Additionally, in this situation, we must also have $\ell \leq k - 1$, and so we bound

$$q(T) \leq 2\ell + \sum_{i=3}^{\ell-1} \lfloor \log(d_i - 1) \rfloor \leq 2k - 2 + (k - 4) \log(m - 2) \leq q(m, k). \quad \square$$

For positive integers $m, k$, let $F_{m,k}$ denote the forest formed by all trees $T \in \text{comp}(F^*)$ with $e(T) = m$ and $|\chi(T)| = k$. By Lemma 6 and Corollary 7, the only values of $(m, k)$ for which $F_{m,k}$ can be nonempty are: $(1, 1)$, $(2, 2)$, $(3, 3)$ and $(m, k)$ where $4 \leq k \leq m \leq 2k - 3$.

For $4 \leq k \leq m \leq 2k - 3$, a quick calculation shows that

$$\frac{q(m, k)}{m+1} \geq \frac{q(m+1, k)}{m+2} \implies \frac{\max_{m:k \leq m \leq 2k-3} q(m, k)}{m+1} = \frac{q(k, k)}{k+1}.$$  

Finally, another short calculation yields $\frac{q(k, k)}{k+1} \leq \frac{q(t,t)}{t+1}$ for all $k \in [t]$.

Thus, by Claim 8, we find that the total number of queries used to locate $F^*$, and thus the monochromatic matching, is bounded above by

$$\sum_{T \in \text{comp}(F^*)} q(T) \leq \sum_{m,k} |\text{comp}(F_{m,k})| q(m, k) = \sum_{m,k} |V(F_{m,k})| \frac{q(m, k)}{m+1} \leq \frac{q(t,t)}{t+1} \sum_{m,k} |V(F_{m,k})| \leq \frac{1}{t+1}(2t - 1 + (t - 3) \log(t - 2))n. \quad \square$$

**Remark 9.** For any fixed $t \geq 4$, one can embark on a more sensitive analysis to improve the upper bound of $\frac{1}{t+1}(2t - 1 + (t - 3) \log(t - 2))n$. For example, when $t = 4$, we can arrive at an upper bound of $\frac{3}{2}n$ as opposed to $\frac{5}{2}n$ by working through Claim 8 more carefully. However, as $t$ grows, it becomes increasingly difficult to carry out such an analysis.

**Painter’s strategy.** We now prove Theorem 1”, thus showing the tightness of Theorem 1’.

**Proof of Theorem 1”.** Consider partitioning $V(K_n) = V_1 \cup \cdots \cup V_t$ where $|V_i| = 2r - 1$ and $|V_i| = r - 1$ for all $i \geq 2$. Let $\chi: E(K_n) \to [t]$ be given by $\chi(xy) = \max \{i : V_i \cap \{x, y\} \neq \emptyset\}$.

Certainly $\chi$ does not contain a monochromatic $rK_2$. For an edge $e \in E(K_n)$ and a color $c \in [t]$, let $\chi_{e,c}$ denote the coloring of $E(K_n)$ obtained by coloring $e$ by color $c$ and coloring the rest of the edges as in $\chi$. Notice that if $e$ is not completely contained in $V_1$, then $\chi_{e,1}$ has a monochromatic $rK_2$ in color 1, and if $e$ is completely disjoint from $V_c$ for some $c \geq 2$, then $\chi_{e,c}$ has a monochromatic $rK_2$ in color $c$.

Therefore, if Painter is not required to guarantee the existence of a monochromatic $rK_2$, then, as Builder queries edges, Painter colors the edge as in $\chi$ until there is only one unexamined edge, call it $e$. Certainly there is some $c \in [t]$ such that $\chi_{e,c}$ has an $rK_2$ in color $c$, so since $\chi$ does not have any monochromatic $rK_2$, Builder must query all $(n \choose 2)$ edges of $K_n$ in order to determine whether or not Painter’s coloring has a monochromatic matching of size $r$.
Now, suppose Painter is required to guarantee the existence of a monochromatic $rK_2$ and Builder needs only determine which color has said matching. Again, Painter will color the edges that Builder queries as in $\chi$ until the very last edge, which she then gives a color which will form a monochromatic $rK_2$. However, against this strategy, Builder can sometimes deduce which color will have this matching before reaching the very last edge.

If $t = 2$, then Painter’s coloring has an $rK_2$ in color 2 if and only if some edge in $V_1$ gets color 2 (and otherwise the coloring must have an $rK_2$ in color 1). Therefore, to determine which color contains the matching, Builder must either query every edge in $V_1$ or query every edge not completely contained in $V_1$. Hence, Builder must query at least
\[
\min\left\{\left(\frac{|V_1|}{2}\right), \left(\frac{n}{2}\right) - \left(\frac{|V_1|}{2}\right)\right\} = \frac{1}{9}(2n + 1)(n - 1),
\]
edges.

For $t = 3$, Builder must query all edges not in $E[V_2, V_3]$ (the edges with one vertex in $V_2$ and the other in $V_3$); otherwise, there is some unexposed edge which is either completely contained in $V_1$ or meets $V_1$ in only one vertex and is disjoint from either $V_2$ or $V_3$. Therefore, Builder must query at least
\[
\left(\frac{n}{2}\right) - |V_2||V_3| = \frac{1}{16}(7n + 1)(n - 1)
\]
edges.

Lastly, for $t \geq 4$, since every edge is disjoint from at least two of $V_1, V_2, V_3, V_4$, Builder must query all $\left(\binom{n}{2}\right)$ edges to determine which color contains the matching.

While Theorem 1'' shows that determining whether or not a monochromatic matching of size larger than is guaranteed to exist requires $\Omega(n^2)$ queries, we wonder if our actual upper bound of $O(n \log t)$ in Theorem 1' is tight when trying to locate a monochromatic matching of the guaranteed size.

**Question 10.** Suppose that $t, r \geq 2$ and $n = (t + 1)r - t + 1$. Does Painter have a strategy for $t$-coloring $E(K_n)$ which requires Builder to query at least $\Omega(n \log t)$ edges to locate a monochromatic $rK_2$?

Notice that, in this situation, it does not matter whether or not we require Builder to locate such a matching or simply determine which color must have this matching: the answer is the same. Indeed, suppose that Builder has queried some collection of edges giving rise to color classes $C_1, \ldots, C_t$, none of which contain a matching on $r$ edges, yet Builder has somehow deduced that, say, color $t$ cannot contain said matching, so the matching must exist among colors $1, \ldots, t - 1$. Painter can then simply color all remaining edges with color $t$, resulting in a coloring of $E(K_n)$ with color classes $C'_1, \ldots, C'_t$. By Theorem 1 or 1', there is some $i$ for which $C'_i$ contains a copy of $rK_2$, but we have $C'_i = C_i$ for $i \in [t - 1]$, so this matching must exist in color $t$; thus showing that Builder was incorrect.

### 3 Monochromatic trees

#### 3.1 Two colors

Here we give the proof of Theorem 2', which was communicated to us by Micek and Pegden [7].
Proof of Theorem 2. Strategy for Painter. As mentioned at the end of Section 2, since any 2-coloring of $E(K_n)$ must have a monochromatic spanning tree, it does not matter whether we require Builder to locate said spanning tree, or simply determine which of the two colors contains it.

Painter’s strategy is as follows: color the first $n - 2$ queried edges red, the next $n - 2$ queried edges blue, and then color the remaining edges arbitrarily.

Since a spanning tree has $n - 1$ edges, Builder cannot have located a monochromatic spanning tree within the first $2n - 4$ queries, and thus needs to query at least $2n - 3$ edges.

Strategy for Builder. Let $\chi : E(K_n) \to [2]$. Builder begins by choosing $v \in V(K_n)$ arbitrarily and queries all $n - 1$ edges incident to $v$. For $i \in \{1, 2\}$, set $C_0^i = \{v\} \cup \{u \in V(K_n) : \chi(uv) = i\}$. Builder proceeds recursively as follows: if there is some $x \in C_i^r \setminus C_i^b$ and $y \in C_i^b \setminus C_i^r$, Builder queries the edge $xy$ and sets $C_{i+1}^r = C_i^r \cup \{x, y\}$ and $C_{i+1}^b = C_i^b \setminus \{x, y\}$.

Notice that, for every $r$, $C_i^r$ is a connected component in color $i$, so if ever $|C_i^r| = n$ for either $i = 1$ or $i = 2$, then Builder has located a spanning tree in color $i$. Set $a(r) = |C_1^r| + |C_2^r|$, so $a(0) = n + 1$ and if $a(r) \geq 2n - 1$, then it must be the case that $|C_i^r| = n$ for either $i = 1$ or $i = 2$.

If Builder has not located a monochromatic spanning tree by step $r$, then since $C_1^r \cup C_2^r = V(K_n)$, we must have $a(r + 1) = a(r) + 1$. We conclude that Builder can locate a monochromatic spanning tree using at most $(n - 1) + (n - 2) = 2n - 3$ queries. \qed

3.2 Three colors

Builder’s strategy. We begin with the key lemma which motivates Builder’s strategy. Recall that $k(n) := \frac{n}{2} + 1$ if $n \equiv 2 \pmod{4}$ and $k(n) := \left\lceil \frac{n}{2} \right\rceil$ otherwise.

Lemma 11. For $n \geq 3$, suppose that there are $U_1, \ldots, U_6 \subseteq V(K_n)$ (some of which may be empty) with $E(K_n) = \bigcup_{i = 1}^6 \binom{U_i}{2}$. If $U_1 \cap U_2 = \emptyset$, then $|U_i| \geq k(n)$ for some $i \in \{6\}$.

Proof. Assume for the sake of contradiction that $|U_i| < k := k(n)$ for each $i \in \{6\}$. Since $k - 1 \leq \left\lfloor \frac{n}{2} \right\rfloor$, and $U_1 \cap U_2 = \emptyset$, we can find $A \supseteq U_1$ and $B \supseteq U_2$ with $A \cap B = \emptyset$ and $|A| = \left\lfloor \frac{n}{2} \right\rfloor$ and $|B| = \left\lceil \frac{n}{2} \right\rceil$.

Consider the edges between $A$ and $B$, denoted by $E[A, B]$. Since $|U_i| \leq k - 1$, by convexity we must have

$$\left| \binom{U_i}{2} \cap E[A, B] \right| \leq \left\lceil \frac{(k - 1)^2}{4} \right\rceil,$$

for every $i$. However, the edges induced by $U_3, \ldots, U_6$ must cover all edges between $A$ and $B$, so we must have

$$4 \left\lceil \frac{(k - 1)^2}{4} \right\rceil \geq \left\lceil \frac{n^2}{4} \right\rceil,$$

contradicting the definition of $k = k(n)$ for any value of $n \pmod{4}$. \qed

Hence, Builder will work to find six subsets of $V(K_n)$, each being contained within a connected component of some color class, wherein each pair of vertices are contained in one of these sets. The following lemma presents the primary tool in Builder’s strategy.

Lemma 12. Let $\chi : E(K_n) \to [3]$ and suppose that Builder has queried some edges giving rise to graphs $C_1, C_2, C_3$ where $C_i$ is the graph formed by the queried edges of color $i$.

Suppose that $X \subseteq V(C_1)$ and $Y \subseteq V(C_2)$ are subsets of connected components of $C_1$ and $C_2$, respectively. There exists a procedure $\text{CompExtend}(\chi, X, Y)$ which returns a tuple $(X_1, X_2, X_3)$ of subsets of $V(K_n)$ using at most $2|X| + 2|Y|$ additional queries. If $C_i^\ast$ denotes the graph formed by the exposed edges of color $i$ after calling $\text{CompExtend}(\chi, X, Y)$, then $(X_1, X_2, X_3)$ satisfies:
1. \( X_1, X_2, X_3 \subseteq X \cup Y \),
2. \( X_1 \supseteq X \) and \( X_2 \supseteq Y \),
3. for all \( i \in [3] \), \( X_i \) is a subset of some connected component of \( C_i^* \), and
4. one of the following:
   (a) \( X_1 = X \cup Y \),
   (b) \( X_2 = X \cup Y \), or
   (c) \( X_3 \supseteq (X \setminus X_2) \cup (Y \setminus X_1) \).

Proof. We first define \( \text{CompExtend} \) and then prove the claimed properties.

1: \textbf{procedure} \( \text{CompExtend}(\chi, X, Y) \)
2: \( X_1 \leftarrow X \)
3: \( X_2 \leftarrow Y \)
4: \( X_3 \leftarrow \emptyset \)
5: \textbf{loop}
6: \textbf{if} there is \( u \in X_1 \setminus (X_2 \cup X_3) \) and \( v \in (X_2 \cap X_3) \setminus X_1 \) \textbf{then}
7: \( \text{Query} \) the edge \( uv \)
8: \( X_{\chi(uv)} \leftarrow X_{\chi(uv)} \cup \{u, v\} \)
9: \textbf{else if} there is \( u \in X_2 \setminus (X_1 \cup X_3) \) and \( v \in (X_1 \cap X_3) \setminus X_2 \) \textbf{then}
10: \( \text{Query} \) the edge \( uv \)
11: \( X_{\chi(uv)} \leftarrow X_{\chi(uv)} \cup \{u, v\} \)
12: \textbf{else if} \( X_3 \subseteq X_1 \cap X_2 \) and there is \( u \in X_1 \setminus X_2 \) and there is \( v \in X_2 \setminus X_1 \) \textbf{then}
13: \( \text{Query} \) the edge \( uv \)
14: \textbf{if} \( \chi(uv) \in \{1, 2\} \) \textbf{then}
15: \( X_{\chi(uv)} \leftarrow X_{\chi(uv)} \cup \{u, v\} \)
16: \textbf{else}
17: \( X_3 \leftarrow \{u, v\} \)
18: \textbf{end if}
19: \textbf{else}
20: \textbf{return} \( (X_1, X_2, X_3) \)
21: \textbf{end if}
22: \textbf{end loop}
23: \textbf{end procedure}

Items (1) and (2) are straightforward to check. To verify Item (3), notice that in Lines 8, 11 and 15, if we extended \( X_i \) to include some vertex \( u \), then this is because there is some \( v \in X_i \) with \( \chi(uv) = i \). In particular, since \( X, Y \) were subsets of connected components of \( C_1, C_2 \), respectively, we have that at every stage, \( X_i \) is a subset of a connected component in the currently exposed edges of color \( i \).

Item (4) follows from the observation that this is the only way that the procedure will ever break the loop and return \( (X_1, X_2, X_3) \). Thus, to verify Item (4), we simply must show that the procedure does indeed terminate. We do so by showing that the procedure terminates after at most \( 2|X| + 2|Y| + 1 \) loops, implying also that it queried at most \( 2|X| + 2|Y| \) additional edges.

For an integer \( r \), let \( X_i^r \) denote the value of \( X_i \) in \( \text{CompExtend}(\chi, X, Y) \) \textit{before} running through the loop for the \((r + 1)\)st time, so \( X_0^0 = X, X_0^0 = Y \) and \( X_3^0 = \emptyset \). Notice that we always have \( X_1^r \subseteq X_1^{r+1} \) and \( X_2^r \subseteq X_2^{r+1} \) and \( X_1^r \cup X_2^r = X \cup Y \supseteq X_3^r \).

9
Set \( a(r) := 2|X_1^r| + 2|X_2^r| + |X_3^r \setminus (X_1^r \cap X_2^r)| \). Certainly \( a(0) = 2|X| + 2|Y| \) and for any \( r \),
\[
a(r) = 2|X_1^r \cup X_2^r| + 2|X_1^r \cap X_2^r| + |X_3^r \setminus (X_1^r \cap X_2^r)| \\
= 2|X_1^r \cup X_2^r| + |X_1^r \cap X_2^r| + |X_3 \setminus (X_1^r \cap X_2^r)| \\
\leq 3|X_1^r \cup X_2^r| + |X_1^r \cap X_2^r| \leq 4|X \cup Y| \leq 4|X| + 4|Y|.
\]

We claim that if CompExtend(\( X, X, Y \)) has not output \((X_1, X_2, X_3)\) after the \((r+1)\)st loop (that is, the procedure will run through the loop for an \((r+2)\)nd time), then \( a(r+1) \geq a(r) + 1 \), thus implying the claim. We break into the following cases depending on how loop \( r \) terminates:

1. If the procedure reaches Line 8, then either:
   - (a) \( X_3^{i+1} = X_3^i \cup \{u\} \). Here, since \( u \notin X_2^i \cup X_3^i \), we have \( |X_3^{i+1} \setminus (X_1^i \cap X_2^i)| = |X_3^i \setminus (X_1^i \cap X_2^i)| + 1 \), so \( a(r+1) = a(r) + 1 \).
   - (b) \( |X_1^{i+1}| \geq |X_1^i| + 1 \) for some \( i \in \{1, 2\} \). Thus, even though \( |X_3^{i+1} \setminus (X_1^i \cap X_2^i)| \) may decrease by 1, we still have \( a(r+1) \geq a(r) + 2 - 1 = a(r) + 1 \).

2. If the procedure reaches Line 11, then \( a(r+1) \geq a(r) + 1 \) by a symmetric argument.

3. If the procedure reaches Line 15, then we have \( |X_1^{i+1}| \geq |X_1^i| + 1 \) for some \( i \in \{1, 2\} \) and also \( |X_3^{i+1} \setminus (X_1^i \cap X_2^i)| = |X_3^{i+1} \setminus (X_1^i \cap X_2^i)| = 0 \), so \( a(r+1) = a(r) + 2 \).

4. Finally, if the procedure reaches Line 17, then \( X_1^{i+1} = X_1^i \) for \( i \in \{1, 2\} \) and \( X_3^{i+1} = \{u, v\} \).
   - Since \( u, v \notin X_1^i \cup X_2^i \), we have \( |X_3^{i+1} \setminus (X_1^i \cap X_2^i)| = 2 \) while \( |X_3^i \setminus (X_1^i \cap X_2^i)| = 0 \), so again \( a(r+1) = a(r) + 2 \).

We now have all of the necessary tools describe Builder’s full strategy and prove Theorem 3’.

Proof of Theorem 3’. Let \( \chi : E(K_n) \rightarrow \{r, g, b\} \) be a 3-coloring. Builder begins by choosing \( v \in V(K_n) \) arbitrarily and queries all \( n-1 \) edges incident to \( v \). Let \( R = \{u \in V(K_n) \setminus \{v\} : \chi(vu) = r\} \) and define \( G \) and \( B \) analogously for colors \( g \) and \( b \). Note that \( R, G, B \) are subsets of the vertices of some connected components in the currently exposed edges in colors \( r, g, b \), respectively. Now, Builder uses CompExtend(\( \chi, R, G \)), CompExtend(\( \chi, G, B \)) and CompExtend(\( \chi, B, R \)) (with the appropriate relabeling of the colors) to find tuples \((R_1, G_2, B_3), (G_1, B_2, R_3)\) and \((B_1, R_2, G_3)\), respectively, as in Lemma 12. This requires at most
\[
(2|R| + 2|G|) + (2|R| + 2|B|) + (2|B| + 2|G|) = 4(|R| + |B| + |G|) = 4(n-1)
\]
additional queries, thus bringing the total number of queries to at most \( 5(n-1) \).

We claim that Builder has located a monochromatic tree on at least \( k(n) \) vertices. Suppose that \( C_r, C_g, C_b \) are the graphs formed by the exposed edges in colors \( r, g, b \), respectively. If \( R_1 \cup R_2 \cup R_3 \) is a subset of a connected component of \( C_r \), set \( R_1^* = R_1 \cup R_2 \cup R_3 \cup \{v\} \) and \( R_2^* = \emptyset \), and otherwise set \( R_1^* = R_1 \cup R_2 \cup \{v\} \) and \( R_2^* = R_3 \). Define \( G_i^* \) and \( B_i^* \) analogously for \( i \in \{1, 2\} \).

In any case, by Lemma 12, we know that for each \( i \in \{1, 2\} \), \( R_i^* \) is a subset of the vertices of a connected component of \( C_r \), \( G_i^* \) is a subset of the vertices of a connected component of \( C_g \), and \( B_i^* \) is a subset of a connected component of \( C_b \), thus we need only show that at least one of these sets has size at least \( k(n) \). We do this by appealing to Lemma 11.

By definition, \( R_1^* \cap R_2^* = G_1^* \cap G_2^* = B_1^* \cap B_2^* = \emptyset \), so we need only show that
\[
E(K_n) = \left( R_1^* \cup R_2^* \right) \cup \left( G_1^* \cup G_2^* \right) \cup \left( B_1^* \cup B_2^* \right),
\]
i.e. every pair of vertices of \( K_n \) are contained together in one of these six sets. Let \( x, y \in V(K_n) \) be two distinct vertices. If, say, \( x = v \), then \( y \in R \cup B \cup G \), and so \( x, y \) are contained together
in one of $R_1^*, G_1^*, B_1^*$. If we have $x, y \in R$, then certainly $x, y \in R_1^*$, and similarly if $x, y \in G$ or $x, y \in B$.

Thus, suppose that, without loss of generality, $x \in R, y \in G$ (the other situations follow symmetrically). Certainly $x \in R_1 \subseteq R_1^*$ and $y \in G_2 \subseteq G_1^*$, so suppose that $x \notin G_1^*$ and $y \notin R_1^*$. Then by Item 4 in Lemma 12, we know that we must have $B_3 \supseteq (R \setminus G_2) \cup (G \setminus R_1)$, so $x, y \in B_3$. Hence, either $x, y \in B_1^*$ or $x, y \in B_2^*$.

**Painter’s strategy.** We conclude by proving Theorem 3, thus showing the tightness of Theorem 3.

**Proof of Theorem 3.** Let $\chi$ be a 3-coloring of $E(K_n)$ formed by starting with a proper edge-coloring of $K_4$ using three colors, “blowing up” each vertex into a cluster of size roughly $n/4$, and then coloring the edges within the clusters arbitrarily. Formally speaking, start with a partition $[n] = V_1 \sqcup V_2 \sqcup V_3 \sqcup V_4$ with $|V_i| \in \{\lfloor n/4 \rfloor, \lceil n/4 \rceil \}$ for all $i$, and let $\chi$ be the 3-coloring of $E(K_n)$ given by

- $\chi(e) = r$ if $e \in E[V_1, V_2] \cup E[V_3, V_4]$,
- $\chi(e) = b$ if $e \in E[V_1, V_3] \cup E[V_2, V_4]$,
- $\chi(e) = g$ if $e \in E[V_1, V_4] \cup E[V_2, V_3]$, and
- $\chi(e)$ is arbitrary otherwise.

It is straightforward to verify that $\chi$ does not contain a monochromatic tree on $k(n) + 1$ vertices.

For an edge $e$ and color $c \in \{r, b, g\}$, let $\chi_{e,c}$ denote the coloring where $e$ gets color $c$ and every other edge is colored as in $\chi$. Notice that if $e$ is any edge not completely contained in some $V_i$, then $\chi_{e,c}$ actually contains a spanning tree in color $c$ whenever $c \neq \chi(e)$.

As Builder queries an edge $e$, Painter colors $e$ as in $\chi$, unless $e$ is the last unexposed edge which is not completely contained in one of the $V_i$’s. In this situation, Painter gives $e$ either color $c_1$ or $c_2$ where $c_1, c_2 \neq \chi(e)$.

Thus, Painter’s coloring will always contain a monochromatic tree on $k(n) + 1$ vertices (in fact, it will always contain a monochromatic spanning tree), but Builder must query every edge not completely contained in some $V_i$ to determine which color contains said tree. As such, Builder must query at least $\sum_{i \neq j \in [4]} |V_i||V_j| / 6 |\frac{n}{4}|^2$ edges to do so.

While Theorem 3 shows the tightness of $k(n)$ in Theorem 3, we wonder more precisely how many queries are necessary when finding a monochromatic tree of size $k(n)$.

**Question 13.** What is the smallest $c$ such that for any 3-coloring of $E(K_n)$, Builder can use at most $(c - o(1))n$ queries to locate a monochromatic tree on at least $k(n)$ vertices?

Theorem 3 shows that $c \leq 5$, and certainly $c \geq 3/2$ since Painter can simply color the first $k(n) - 1$ edges red, the next $k(n) - 1$ edges blue, and the next $k(n) - 1$ edges green. Analogously to Question 10, since any 3-coloring of $E(K_n)$ must contain a monochromatic tree on $k(n)$ vertices, this question is ambivalent as to whether Builder must actually locate said tree or just determine which color contains it.

Theorems 2 and 3 were extended by Gyárfás in [5] to show that any $t$-coloring of $E(K_n)$ must contain a monochromatic tree on at least $\frac{n}{t}$ vertices. Füredi [3] showed that this bound can be slightly improved in the cases where an affine plane of order $t - 1$ does not exist. This suggests the natural question:

**Question 14.** What is the least number of queries necessary for Builder to locate a monochromatic tree on at least $\frac{n}{t}$ vertices in a $t$-coloring of $E(K_n)$?
We suspect, much like in the case of 2 and 3 colors, that Builder can locate such a tree using $O_t(n)$ queries.

References

[1] J. Beck. Achievement games and the probabilistic method. In *Combinatorics, Paul Erdős is eighty, Vol. 1*, Bolyai Soc. Math. Stud., pages 51–78. János Bolyai Math. Soc., Budapest, 1993.

[2] E. Cockayne and P. J. Lorimer. The Ramsey number for stripes. *Journal of the Australian Mathematical Society*, 19(2):252–256, 1975.

[3] Z. Füredi. Covering the complete graph by partitions. *Discrete Mathematics*, 75(1-3):217–226, 1989.

[4] L. Gerencsér and A. Gyárfás. On Ramsey-type problems. *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.*, 10:167–170, 1967.

[5] A. Gyárfás. Particiófedések és lefogóhalmazok hipergráfokban. *Tanulmányok-MTA Számítástechn. Automat. Kutató Int. Budapest*, (71):62, 1977.

[6] A. Kurek and A. Ruciński. Two variants of the size Ramsey number. *Discuss. Math. Graph Theory*, 25(1-2):141–149, 2005.

[7] P. Micek and W. Pegden. Personal communication.