Basic Research in Computer Science

The Complexity of Identifying Large Equivalence Classes

Peter G. Binderup
Gudmund Skovbjerg Frandsen
Peter Bro Miltersen
Sven Skyum

BRICS Report Series **RS-98-34**

ISSN 0909-0878 **December 1998**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark
Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/98/34/`

The Complexity of Identifying Large Equivalence Classes*

Peter G. Binderup Gudmund S. Frandsen
Peter Bro Miltersen Sven Skyum

BRICS, Basic Research in Computer Science,
Centre of the Danish National Research Foundation,
Department of Computer Science, University of Aarhus.

December, 1998

Abstract

We prove that at least $\frac{3k-4}{k(2k-3)}\binom{n}{2} - \mathrm{O}(k)$ equivalence tests and no more than $\frac{2}{k}\binom{n}{2} + \mathrm{O}(n)$ equivalence tests are needed in the worst case to identify the equivalence classes with at least k members in set of n elements. The upper bound is an improvement by a factor 2 compared to known results. For $k = 3$ we give tighter bounds. Finally, for $k > \frac{n}{2}$ we prove that it is necessary and it suffices to make $2n - k - 1$ equivalence tests which generalizes a known result for $k = \lceil \frac{n+1}{2} \rceil$.

1 Introduction

Consider a "Master Mind"-like combinatorial game between two players, Alice and Bob. Bob hides n coloured pegs in n positions. Pegs come in infinitely many different colours. The aim for Alice is then to decide whether Bob has hidden at least k pegs of the same colour. To this end, Alice may pose only one kinds of question to Bob, which is, if a peg in one position has the same colour as a peg in another position. What is the number of questions Alice needs to ask in the worst case?

*This paper is based on the conference paper [7] by the last three authors and the Master's Thesis [3] by the first author. Email: {binderup,gudmund,bromille,sskyum}@brics.dk. Supported by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT).

Upper bounds will be given through analyzing efficient strategies for Alice. A strategy for Alice describes what the next question should be when given the answers to previous questions.

To obtain lower bounds, we will use what is known as *adversary strategies.* What does this mean? Assume Bob doesn't hide the pegs in the beginning but that he makes Alice believe that he did. Then Alice will not discover that Bob hasn't hidden the pegs as long as Bob's answers are consistent with at least one way of placing n coloured pegs in n positions. The least number of questions Bob can force Alice to pose (no matter what strategy Alice uses) through answering shrewdly but consistently, is a lower bound. The description of how Bob should answer a question from Alice given answers to previous questions is what is called an adversary strategy. Originally the interest in the problem was for $k = \lceil \frac{n+1}{2} \rceil$ motivated by the design of fault tolerant computer systems where a majority of processors must agree on the output [11].

Moving to more formal terminology, we will use a computational model, *where the only allowed operation is equivalence test on pairs of elements.* The problems are parameterized by two parameters n and k. We define the following complexity measures:

Definition 1.1 *Let $E(k,n)$, $2 \leq k \leq n$, denote the minimum number of equivalence tests needed in the worst case for an algorithm to decide whether a set of n elements contains an equivalence class with at least k members.*

Definition 1.2 *Let $R(k,n)$, $2 \leq k \leq n$, denote the minimum number of equivalence tests needed in the worst case for an algorithm to find a representative of each equivalence class with at least k members in a set of n elements.*

Definition 1.3 *Let $C(k,n)$, $2 \leq k \leq n$, denote the minimum number of equivalence tests needed in the worst case for an algorithm to find a representative and the cardinality of each equivalence class with at least k members in a set of n elements.*

It is obvious that in the example above $E(k,n)$ is exactly the number of questions Alice needs to ask Bob in the worst case. Furthermore, it is clear that $E(k,n) \leq R(k,n) \leq C(k,n)$. We prove that $E(k,n) = R(k,n)$ for all k and $E(3,n) = R(3,n) = C(3,n)$. Whether equality holds between $E(k,n)$ and $C(k,n)$ is open for $3 < k < \frac{n}{2}$.

$C(k,n)$ is nonincreasing in k – if one has a representative of all equivalence classes of size at least k and knows the cardinality of these classes, it will

be possible without extra tests to select those representatives representing equivalence classes of size at least $k+1$. However, no such monotonicity knowledge of $E(k,n)(=R(k,n))$ is known.

Clearly $E(2,n) = R(2,n) = C(2,n) = \binom{n}{2}$ and $E(n,n) = R(n,n) = C(n,n) = n-1$. The fact that no elements are equivalent can only be known to an algorithm after all elements have been tested against each other. That all elements are equivalent can be tested by testing one element against the remaining $n-1$ elements.

However, the problem becomes nontrivial for $2 < k < n$. The following table summarizes our results together with the results known to us from the literature. Several of the bounds listed are slightly weaker than those actually obtained. This is to make comparisons easier.

Range	Upper bound on C	Lower bound on $E = R$	Reference
$k = 3$	$\frac{3}{5}\binom{n}{2} + \mathrm{O}(n)$	$\frac{7}{12}\binom{n}{2} - \mathrm{O}(n)$	Theorems 3.1 and 3.2
All k	$\frac{4}{k}\binom{n}{2}$		[5]
All k	$\frac{2}{k}\binom{n}{2} + 1$	$\frac{3k-4}{k(2k-3)}\binom{n}{2} - \mathrm{O}(k)$	Theorems 4.1 and 5.1
$\frac{n}{2} > k > \frac{n}{3}$	$4n - 5k - 2$	$4n - 4k - 2$	Theorems 4.1 and 5.1
$k = \lceil \frac{n+1}{2} \rceil$	$\lfloor \frac{3}{2}(n-1) \rfloor$	$\lfloor \frac{3}{2}(n-1) \rfloor$	[6, 9]
$k > \frac{n}{2}$	$2n - k - 1$	$2n - k - 1$	Theorems 4.1 and 5.1

If it is known in advance that there is only two kinds of elements (two equivalence classes) then better bounds are known for $k = \lceil \frac{n+1}{2} \rceil$. It is necessary and it suffices to make $n - B(n)$ equivalence tests to find a representative of the larger class if n is odd ($B(n)$ denotes the number of ones in the binary representation of n) [12].

If the elements are from an ordered set and two elements are equivalent if and only if they are equal, then if $\leq$-tests are allowed, it is possible to sort n elements in time $\mathrm{O}(n \log n)$ and thereby get full information about the equivalence classes. But is it possible to find representatives of equivalence classes with at least k members even faster? The answer to this is affirmative and given in [10], where it is proven that the number of tests is $\Theta(n \log \frac{n}{k})$.

1.1 Notation

When we solve a problem concerning a set of n elements using equivalence tests on pairs we may keep track of previous answers to queries in an undirected, complete graph on n vertices. Each vertex represents an element and the edges are coloured black, red, and green. Initially, all edges are black.

Whenever a test $x?y$ is answered with $\equiv$ ($\not\equiv$), the corresponding edge in the graph is coloured green (red). Furthermore, *a vertex is green* if it has at least one outgoing green edge – otherwise the vertex is red. So, there are no black vertices. A *red clique* (*black clique*) is a subgraph on red vertices connected exclusively with red (black) edges. We call such a graph an *information graph*.

An algorithm for deciding whether there are k equivalent elements among n may terminate with the answer YES if a tree of at least $k-1$ green edges occurs in the graph. Using the transitivity of $\equiv$, the vertices connected by these edges will be an equivalence class of size at least k. If at some point in time, on the other hand, all cliques of size k in the graph contains a red edge, the algorithm may terminate with the answer NO. The algorithm may not terminate in any other situation.

2 Relations Between Measures

It is clear that $E(k,n) \leq R(k,n) \leq C(k,n)$. That $E(k,n) = R(k,n)$ is less clear but nevertheless the case.

Theorem 2.1 $E(k,n) = R(k,n)$.

Proof Let A be an optimal algorithm deciding whether there exists a large class (size at least k) in a set of n elements or not using at most $E(k,n)$ equivalence tests.

The algorithm A can be used as a black box to create an algorithm B that finds representatives of all large classes. Given the set of elements, B uses A to determine which tests to make. The answers are passed on to A by B, but B might lie to A and thereby avoid that A terminates prematurely. B's strategy for lying is as follows. If B gets a $\not\equiv$-answer, the correct answer is passed on to A. If B gets an $\equiv$-answer, B will lie and give A a $\not\equiv$-answer if A upon reception of an $\equiv$-answer could conclude that there was a large equivalence class.

At termination, A will conclude (perhaps based on false information) that there are no large classes. We will show that B is able to present exactly one representative of every true large class. Since B only changes $\equiv$-answers to $\not\equiv$-answers and not vice-versa, we know that at termination every class known to A will be a subclass of a true class. In other words, the class division known to A is a refinement of the true class division.

Assume that large classes exist and let U be one of them. We will show that B at termination will present exactly one element from U. Since A is

not aware of any large classes it will at some point in time have made a test between two elements in U and have received a false answer. The only way A could have been given a false answer is, if A previously had discovered two subclasses U_1 and U_2 of U, such that $|U_1| + |U_2| \geq k$ and then had tested an element of U_1 against an element of U_2. When A got the false answer from B, B knew the right answer and hence that U_1 and U_2 contained equivalent elements.

Conclusively, B will know large subclasses of all true large classes at termination. We have to show that B only presents one representative for each true large class.

Assume B knows two large subclasses V and W of the same true large class – but doesn't know that they contain equivalent elements. A will know two subclasses V_1 and V_2 of V, such that $|V_1| + |V_2| \geq k$. Similarly, A will know of subclasses W_1 and W_2 in W with the same properties. Assume wlog that $|V_1| \geq |V_2|$ and $|W_1| \geq |W_2|$. Then $|V_1| + |W_1| \geq k$. A and thereby B will therefore have had to make a test between an element of V_1 and an element of W_1. This contradicts that B was assumed not to know that V and W contained equivalent elements. □

Note that if $k = 3$ then A will know a pair in all large classes and will have had to tested this against all other classes - including singletons. So at termination, B will know the cardinality of all large classes. This implies the following corollary:

Corollary 2.1 $E(3, n) = R(3, n) = C(3, n)$.

3 The Complexity of Finding Triplicates

3.1 Upper Bound

Theorem 3.1

$$E(3, n) = R(3, n) = C(3, n) \leq \frac{3}{5}\binom{n}{2} + \mathrm{O}(n).$$

Proof The algorithm to be described in the following will, given n elements, determine whether there exists three equivalent elements among them or not. If the algorithm at any point in time determines three or more equivalent elements it stops and answers YES. The algorithm runs in three phases.

Phase 1: The elements are inserted one by one and two sets S_1 and D_1 are maintained in the following way. The new element e is tested against

elements in S_1. If it is equivalent to one of them, e' say, then e' is removed from S_1 and e together with e' are added to D_1 as a *pair*. e is tested against representatives for the other pairs in D_1. Phase 1 terminates when $|S_1| + \frac{|D_1|}{2} \geq \frac{2n}{5}$.

Phase 2 As Phase 1 but with sets S_2 and D_2. Phase 2 terminates when all the remaining elements have been inserted.

Phase 3 Elements in S_1 and representatives for pairs in D_1 are tested against elements in S_2 and representatives for pairs in D_2. Notice that since no elements in S_1 and S_2 respectively are equivalent to each other there is no need to make tests among elements in S_1 and S_2.

Let $s_i = |S_i|$, $d_i = \frac{|D_i|}{2}$ for $i = 1, 2$ and $d = d_1 + d_2$. Then by the termination criteria for Phase 1 we know that $\frac{2n}{5} \leq s_1 + d_1 < \frac{2n}{5} + 1$ and $s_2 + d_2 = n - s_1 - d_1 - d > \frac{3n}{5} - d - 1$.

We count the number of black edges in the information graph at termination or the number of tests saved by the algorithm:

For every two pairs $\{a_1, b_1\}, \{a_2, b_2\} \in (D_1 \times D_1) \cup (D_2 \times D_2)$ we have saved at least two tests between the elements a_1, b_1, a_2 and b_2. Thus $|black((D_1 \times D_1) \cup (D_2 \times D_2)| \geq d_1(d_1 - 1) + d_2(d_2 - 1)$.

For every two pairs $\{a_1, b_1\}, \{a_2, b_2\} \in D_1 \times D_2$ we have saved three tests. Thus $|black(D_1 \times D_2)| \geq 3d_1d_2$.

For every $a \in S_1$ and $\{a_1, b_1\} \in D_2$ we have saved one. Equivalently for S_2 and D_1. Thus $|black(S_1 \times D_2) \cup (S_2 \times D_2))| \geq s_1d_2 + s_2d_1$.

Finally, $|black(S_1 \times S_2)| = s_1s_2$ since elements in S_1 are not tested against elements in S_2.

Altogether at least

$$d_1(d_1 - 1) + d_2(d_2 - 1) + 3d_1d_2 + s_1d_2 + s_2d_1 + s_1s_2 =$$

$$d^2 - d + (s_1 + d_1)(s_2 + d_2) > d^2 - d + \frac{2n}{5}(\frac{3n}{5} - d - 1)$$

tests are saved. The last expression is minimal for $d = \frac{n}{5} + \frac{1}{2}$ where the value is

$$\frac{n(n-3)}{5} - \frac{1}{4}.$$

Thus at most

$$\binom{n}{2} - (\frac{n(n-3)}{5} - \frac{1}{4}) = \frac{3}{5}\binom{n}{2} + \mathrm{O}(n)$$

tests are needed. □

3.2 Lower Bound

Theorem 3.2

$$\frac{7}{12}\binom{n}{2} - \mathrm{O}(n) \leq E(3,n) = R(3,n) = C(3,n).$$

Proof The following adversary strategy is used:

When an a?b test is made ($\{a,b\}$ a black edge) then the answer given is $\not\equiv$ unless this means that a red clique of size at least $\frac{n}{3}$ will occur in the graph. Then the answer $\equiv$ is given and a and b then become a pair of green vertices.

Let A be an arbitrary algorithm determining whether there are three equivalent elements among n or not.

Let D be the set of green vertices and S the set of red vertices in the information graph at termination of A. Let $d = \frac{|D|}{2}$ and $s = |S| = n - 2d$.

At termination A must have verified that there are no triplicates. For that reason $|red(D \times (D \cup S))| \geq ds + \frac{d(d-1)}{2}$. At the time when a green pair $\{a,b\}$ is formed, at least $\frac{n}{3} - 2$ vertices are connected to both a and b by a red edge. Only half of them are needed in the end to verify that no triplicates exist. Consequently extra $d(\frac{n}{3} - 2)$ red edges between vertices in D and $D \cup S$ can be identified. In S there can be no black 3-clique and no red clique of size $\frac{n}{3}$ or more. The combinatorial theorem and lemma below then imply that $|red(S \times S)| \geq \max\{\binom{s}{2} - \frac{s^2}{4}, \frac{s}{2}(s - \frac{n}{3} - 1)\}$.

Altogether

$$dr + \frac{d(d-1)}{2} + d(\frac{n}{3} - 2) + \max\{\frac{s^2}{4}, \frac{s}{2}(s - \frac{n}{3} - 1)\} \qquad (1)$$

is a lower bound on the number of tests A needs to make.

s must be at least $\frac{n}{3} - 2$. Elementary calculations shows that (1) in the interval $[\frac{n}{3} - 2, n]$ has its minimum for $s = \frac{2n}{3}$ where the value is

$$\frac{7n^2}{24} - \frac{3n}{4}.$$

□

Theorem 3.3 *Let G be an undirected graph with n vertices. For integer $k \geq 2$, if the number of edges is more than*

$$\frac{k-2}{2(k-1)}n^2$$

then there is a clique of size k in G.

Proof The theorem is an immediate consequence of *Turán's Theorem* (see e.g. [4] or [8]).

Lemma 3.1 *Let G be an undirected graph with n vertices and $k > 2$ an integer. If the number of edges is less than $\frac{n}{2}(n-k-1)$ then there exists either 3 independent vertices or a clique of size k.*

Proof Assume no 3 vertices are independent and that G has no clique of size k. Let v be an arbitrary vertex. The vertices that are *not* neighbours to v must form a clique (of size less than k). Thus v has at least $n-k-1$ neighbours. Since v was arbitrary, the lemma follows. □

4 An Upper Bound on $C(k,n)$

The algorithm to be presented is a generalization of the majority algorithm presented in [6]. That is, for $k = \lceil \frac{n+1}{2} \rceil$ the two algorithms do exactly the same.

Theorem 4.1

$$C(k,n) \leq \lfloor \frac{n}{k} \rfloor (2n - k\lfloor \frac{n}{k} \rfloor - 1) \leq \lceil \frac{2}{k} \binom{n}{2} \rceil.$$

Proof Equivalence classes with at least k members are called *large classes* in the following. There can be at most $q = \lfloor \frac{n}{k} \rfloor$ large classes. The algorithm to be described works in two phases. In the first phase at most q distinct candidates to be representatives of the large classes are found. In the second phase, it is checked whether each candidate in fact is a representative of a large class and if so the cardinality of the class is determined.

The elements are inserted successively into the data structure consisting of buckets of elements. All elements within a bucket are equivalent. The buckets are indexed 0 through m. Initially, only bucket B_0 exists ($m = 0$). Only the buckets $B_0, \ldots, B_q$ may contain more than one element – so each bucket $B_{q+1} \ldots, B_m$ (called the *chain*) has exactly one element. B_0 is only temporarily non-empty. Just before an insertion is initiated, B_0 is always empty. Let $e_1, e_2, \ldots, e_m$ be representatives for the buckets. The following property is kept invariantly true through the algorithm:

$$\text{If } i \neq j \text{ and } e_i \equiv e_j \text{ then } |i-j| > q \tag{2}$$

Phase 1: Insertion works as follows. The new element is tested against $e_1, e_2, \ldots, e_{min(q,m)}$. This is done most efficiently (for reasons which will

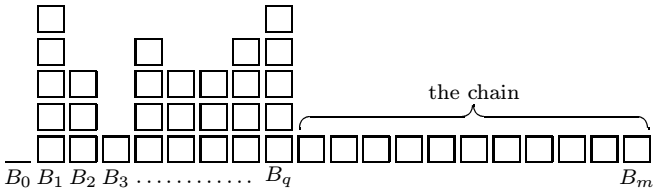

Figure 1: The data structure

become clear) when buckets with more elements are visited before buckets with fewer elements. In case two buckets have the same number of elements, the bucket with the lowest index is visited first. Testing goes on until a test results in that an equivalent element is found. When and if that happens, the element is put into the bucket in question. If the new element is not among $\{e_1, e_2, \ldots, e_{min(q,m)}\}$, it is put into B_0.

In the latter case we have to rearrange the elements in the data structure so that B_0 will become empty again. To this end, we create a new bucket B_{m+1} and shift the contents of the buckets one up. If bucket B_{q+1} exists, it may now contain more than one element. If so, all but one of these elements are moved to B_0, and a new shift is performed. The sequential shifting is continued until B_{q+1} only houses one element. This will happen after at most $q + 1$ shifts since B_0 contains one element to start with. Notice that shifting does not violate property (2).

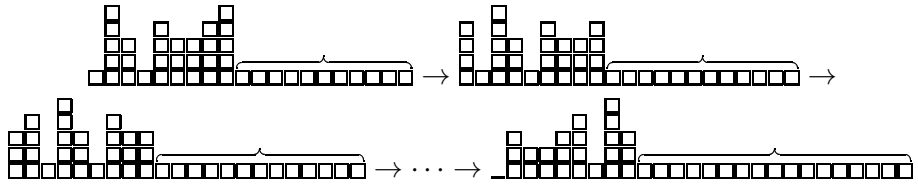

Figure 2: Shifting

Phase 2: All large classes is necessarily represented in the first $min(q, m)$ buckets – there is simply not enough space to house k equivalent elements in the chain because of property (2). If $m \leq q$, we have full information of the data and the algorithm may terminate. Otherwise, $e_1, e_2, \ldots, e_q$ are tested against the elements in the chain. Notice that because of property (2), e_i need not be tested against elements of the first i buckets in the chain (buckets $B_{q+1}, B_{q+2}, \ldots, B_{q+i}$).

Analysis

If $m \leq q$ at termination there is no need for tests in Phase 2. In Phase 1, at most q tests will be needed to insert an element into the data structure. The

element first inserted will obviously be inserted without tests. Therefore, at most $q(n-1) \leq q(2n-kq-1)$ tests are needed in all and so the theorem holds in this simple case.

If $m > q$ we first analyze how many tests are needed in Phase 2. The following lemma is used.

Lemma 4.1 *Let property (2) hold for elements $e_1, e_2, \ldots, e_s$. Let $t > 0$ and e be an arbitrary element. Finally let $c_e = |\{i \mid e_i \equiv e\}|$.*
By at most $\max\{0, s-q(t-1)\}$ tests it can be determined whether $c_e \geq t$ or not and if $c_e \geq t$ the value of c_e can be computed.

Proof We may test e against $e_1, e_2, \ldots, e_s$ one by one. If $e \equiv e_j$ the next q elements are skipped. We stop if there is no hope of finding at least t elements equivalent to e.

Notice that $c_e \leq \frac{s-1}{q+1} + 1$.

Assume u elements are found to be equal to e.

If $u = c_e \geq t$ at least $q(u-1) \geq q(t-1)$ elements are skipped.

If $0 < u < t$ the last v elements are skipped where v satisfies $t-u = \frac{v}{q+1}+1$ or $v = (q+1)(t-u-1)$. qu other elements are also skipped so the total number of skipped elements are $qu+(q+1)(t-u-1) = (q+1)(t-1)-u \geq q(t-1)$.

In both cases at most $s-q(t-1)$ tests are made. □

Let b_i denote the number of elements in bucket B_i, let $d = b_1 + \cdots + b_q - q = n-m$ denote the number of "duplicates" situated in buckets B_1 through B_q, and let $r = n - qk$.

According to Lemma 4.1 $\max\{0, m-(q+i)-q(k-b_i-1)\} = \max\{0, r-d+qb_i-i\}$ tests suffice in Phase 2 to determine whether e_i is a representative for a large class or not and to determine the size of it if it is large.

Now let $a_i = r - d + qb_i - i$. The a_i's are all different and $\max\{a_i, 0\}$ is the number of tests made in Phase 2 to deal with elements equivalent to e_i.

Claim:

$$\Phi = pq + (r-1)\max\{q-m, 0\} + \sum_{i=1}^{q} \max\{a_i, 0\}$$

tests suffice to add the last p elements in Phase 1 and to perform Phase 2.

Initially $\Phi = nq + q(r-1) = \lfloor \frac{n}{k} \rfloor (2n - \lfloor \frac{n}{k} \rfloor k - 1)$ so the theorem follows when the claim has been proven correct.

The claim is proven by induction in p:

When $p = 0$ only Phase 2 remains. Then $\Phi = \sum_{i=1}^{q} \max\{a_i, 0\}$ which is enough for Phase 2 according to discussion above.

Now assume that the claim is true for some p less than n. We will show that then it is true for $p+1$.

Let e be the $p+1$'st last element to be added to the structure. In what follows, unmarked constants refer to values before the $p+1$'st last step and marked constants to values after the step. The induction step is therefore to prove that $\Phi - \Phi'$ is at least the number of tests involved in the step.

There are three cases to consider:

Case 1: $e \equiv e_j$ for some $1 \leq j \leq q$. Then $m' = m$, $d' = d + 1$,

$$b_i{}' = \begin{cases} b_i & \text{if } i \neq j \\ b_j + 1 & \text{if } i = j \end{cases} \quad \text{and} \quad a_i{}' = \begin{cases} a_i - 1 & \text{if } i \neq j \\ a_j + q - 1 & \text{if } i = j. \end{cases}$$

e is tested against s elements $e_{i_1}, e_{i_2}, \ldots, e_{i_s} (= e_j)$, where $a_{i_1} > a_{i_2} > \ldots > a_{i_s} (= a_j)$.

Let t be maximal such that $a_{i_t} > 0$. If $t < s$ then $0 \geq a_{i_{t+1}} > a_{i_{t+2}} > \cdots > a_s = a_j$ and $a_j \leq t - s - 1$. Therefore $\Phi - \Phi' \geq q + t - (t - s + 1 + q - 1) = s$. If $t = s$ then $a_j > 0$ and $\Phi - \Phi' = q + s - q = s$.

In both cases $\Phi - \Phi' \geq s$ and we are done.

Case 2: $e \not\equiv e_i$ for $i = 1, 2, \ldots, m$ and $m < q$. Then $m' = m + 1$, $d' = d$

$$b_i{}' = \begin{cases} 1 & \text{if } i = 1 \\ b_{i-1} & \text{if } i > 1 \end{cases} \quad \text{and} \quad a_i{}' = \begin{cases} r - d + q - 1 & \text{if } i = 1 \\ a_{i-1} - 1 & \text{if } i > 1 \end{cases}$$

Let $t = |\{i \mid a_i > 0\}$. Then $\Phi - \Phi' \geq q + (r-1) + t - (r - d + q - 1) = t + d \geq m$.

The last inequality needs an argument. If $d = 0$ then $t = m$. Increasing d by one can change at most one of the a_i's from being positive since they are different.

The number of tests is m so we are done. □

Case 3: $e \not\equiv e_i$ for $i = 1, 2, \ldots, m$ and $m \geq q$.

Assume $s \leq q + 1$ shifts are made. That is $b_{q-s+1} = 1$ while $b_i > 1$ for $i > q - s + 1$. Then $d' = d - s + 1$, $m' = m + s$,

$$b_i{}' = \begin{cases} b_{i+q-s+1} - 1 & \text{if } i < s \\ 1 & \text{if } i = s \\ b_{i-s} & \text{if } i > s. \end{cases} \quad \text{and} \quad a_i{}' = \begin{cases} a_{i+q-s+1} & \text{if } i < s \\ r - d + q - 1 & \text{if } i = s \\ a_{i-s} - 1 & \text{if } i > s. \end{cases}$$

Let $t = |\{i \mid (s < i \leq q) \wedge (a_{i-s} > 0)\}| = |\{i \mid (i \leq q - s) \wedge (a_i > 0)\}|$.

Then $\Phi - \Phi' = q + t + \max\{a_{q-s+1}, 0\} - \max\{a_s{}', 0\}$.

If $a_s' \leq 0$ this is at least q. If $a_s' > 0$ then $a_i' > 0$ for $i > s$, and $t = q - s$. Therefore $\Phi - \Phi' = q + t + \max\{0, r - d + s - 1\} - (r - d + q - 1) = q + \max\{0, r - d + s - 1\} - (r - d + s - 1)$ which again is at least q.

The number of tests is q so we are done again. □

out that the algorithm are will with a $n - (q + i) - q(k - 2)$ algorithm is

5 A Lower Bound on $E(k, n) = R(k, n)$

by means of an

In [6] it was proven that $E(\lceil \frac{n+1}{2} \rceil, n) = C(\lceil \frac{n+1}{2} \rceil, n) = \lfloor \frac{3}{2}(n-1) \rfloor$. The tight lower bound was obtained using an adversary strategy with 5 cases. This result will come out as a simple special case of the lower bound to be presented. The bound will be obtained by means of a simple adversary strategy.

Theorem 5.1

$$E(k, n) = R(k, n) \geq \lfloor \frac{n}{k} \rfloor ((2n - k\lfloor \frac{n}{k} \rfloor - 1) - k\binom{\lfloor \frac{n}{k} \rfloor}{2} \text{ and}$$

$$E(k, n) = R(k, n) \geq \frac{3k - 4}{k(2k - 3)}\binom{n}{2} - \frac{3k}{4}.$$

Proof number of

For a fixed constant q (chosen later) we use the following adversary strategy:

When an a?b test is made ($\{a, b\}$ a black edge) then answer $\not\equiv$ when this answer is consistent with there being exactly q disjoint classes of size k among the n elements and that the remaining $r = n - pk$ elements are singletons (classes of size one). Otherwise, answer $\equiv$.

At termination an algorithm for solving the problem must have discovered all q classes and can present to us a representative for each of those q classes. Denote the classes $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q$, $\mathcal{C} = \cup \mathcal{C}_i$ and the r singletons R.

to the

The number of red and green edges in the graph at termination will be equal to the number of tests the algorithm has made. For a set F of edges let $red(F)$ ($green(F)$) denote the set of red (green) edges in F.

We count red and green edges in four disjoint sets of edges:

$\bigcup_{i=1}^{q} \mathcal{C}_i \times \mathcal{C}_i$:
For each i, the algorithm knows that $\mathcal{C}_i$ is a class so $|green(\mathcal{C}_i \times \mathcal{C}_i)| \geq k-1$. In total

$$|green(\bigcup_{i=1}^{q} \mathcal{C}_i \times \mathcal{C}_i)| \geq q(k-1).$$

$\mathcal{C} \times R$:
Let $1 \leq i \leq q$ and $a \in R$ be arbitrary and assume that $|red(a \times \mathcal{C}_i)| = 0$. Let $\{b, c\}$ be the first edge in $\mathcal{C}_i \times \mathcal{C}_i$ that became green. The answer should have been $\not\equiv$ since $(\mathcal{C}_i \setminus \{b\}) \cup \{a\}$ was another possibility for the i'th class. If $|red(a \times \mathcal{C}_i)| = 1$ let $b \in \mathcal{C}_i$ be the vertex connected to a by a red edge. Let $\{b, c\}$ be the first green edge connecting b with another vertex in $\mathcal{C}_i$. $\{a, c\}$ is black so the answer is once again wrong since $(\mathcal{C}_i \setminus \{b\}) \cup \{a\}$ is another possibility for the i'th class. Consequently $|red(a \times \mathcal{C}_i)| \geq 2$ for all $a \in R$ and i. Thus

$$|red(\mathcal{C} \times R)| \geq 2qr = 2q(n - kq).$$

$\bigcup_{i<j} \mathcal{C}_i \times \mathcal{C}_j$:
Let $1 \leq i < j \leq q$ be arbitrary and assume that $|red(\mathcal{C}_i \times \mathcal{C}_j)| < k$. Then there is an $(a, b) \in \mathcal{C}_i \times \mathcal{C}_j$ such that $red(a \times \mathcal{C}_j) = red(\mathcal{C}_i \times b) = \emptyset$. Let $\{c, d\}$ be the first edge in $(\mathcal{C}_i \times \mathcal{C}_i) \cup (\mathcal{C}_j \times \mathcal{C}_j)$ that became green. Assume wlog that $\{c, d\} \in \mathcal{C}_i \times \mathcal{C}_i$. $\{c, b\}$ and $\{d, b\}$ are black. Consequently the answer should have been $\not\equiv$ since $(\mathcal{C}_i \setminus \{c\}) \cup \{b\}$ and $(\mathcal{C}_j \setminus \{b\}) \cup \{c\}$ was another possibility for the i'th and j'th class. Thus $|red(\mathcal{C}_i \times \mathcal{C}_j)| \geq k$. In total

$$|red(\bigcup_{i<j} \mathcal{C}_i \times \mathcal{C}_j)| \geq \frac{kq(q-1)}{2}.$$

If we choose $q = \lfloor \frac{n}{k} \rfloor$ and add the three terms together we get

$$E(n,k) \geq \lfloor \frac{n}{k} \rfloor (k-1) + 2\lfloor \frac{n}{k} \rfloor (n - k\lfloor \frac{n}{k} \rfloor) + \frac{k\lfloor \frac{n}{k} \rfloor (\lfloor \frac{n}{k} \rfloor - 1)}{2} \tag{3}$$

from which the first half of the theorem follows.

$R \times R$:
There is no subset of R forming a black clique of size k and no edges are green. By Theorem 3.3 we get that

$$|red(R \times R)| > \binom{r}{2} - \frac{k-2}{2(k-1}r^2 = \frac{r^2}{2(k-1)} - \frac{r}{2} = \frac{(n-kq)^2}{2(k-1)} - \frac{n-kq}{2}.$$

Adding all four terms together, we get the following lower bound on the number of queries, the algorithm needs to make:

$$q(k-1) + k\frac{q(q-1)}{2} + 2q(n-kq) + \frac{(n-kq)^2}{2(k-1)} - \frac{n-kq}{2}.e0.5in \tag{4}$$

This is a second degree polynomial in q that assumes its maximum when

$$q = q_{max} = \frac{(k-1)^2 + n(k-2)}{k(2k-3)}.$$

This value might not be an integer. When $k \geq 2$ this is clearly positive and if $k \leq \frac{n}{3}$ then $\lceil q_{max} \rceil < q_{max} + 1 \leq \frac{n}{k}$. So if $q_{max} + 1$ is substituted for q in (4) we get a valid lower bound when $k \leq \frac{n}{3}$.

When $q_{max} + 1$ is substituted for q in (4) we get after some calculations:

$$\frac{3k-4}{k(2k-3)}\binom{n}{2} - \frac{3k^4 - 8k^3 + 3k^2 + 4k - 1}{2k(k-1)(2k-3)}.$$

When $k > \frac{n}{3}$ the bound given above (3) is stronger so both bounds are valid in general. □

References

[1] M. Ajtai, J. Kómlos, and E. Szémerèdi, A Note on Ramsey Numbers, *Journal of Combinatorial Theory*, Series A **29** (1980) 354-360.

[2] L. Alonso, E. M. Reingold and R. Schott, Determining the Majority. *Information Processing Letters* **47** (1993) 253-255. Majority.

[3] P. G, Binderup, A Combinatorial Problem. Master's Thesis, Dept. of Comp. Sci. Aarhus, May 1998.

[4] B. Bollobás, *Extremal Graph Theory.* Academic Press (1978).

[5] D. Campbell and T. McNeill, Finding a Majority when Sorting is not available. *The Computer Journal* **35** (1991) 186.

[6] M. J. Fischer and S. L. Salzburg, Solution to problem 81-5. *Journal of Algorithms* **3** (1982) 376-379.

[7] G. S. Frandsen, P. B. Miltersen, and S. Skyum, The Complexity of Finding Replicas using Equality Tests. *Proc. Mathematical Foundations of Computer Science 1993, LNCS* **711**, 463-472.

[8] R. L. Graham and V. Rödl, Numbers in Ramsey Theory. Surveys in Combinatorics 1987, *London Mathematical Society Lecture Note Series* **123**, Cambridge University Press, 111-153.

[9] D. W. Matula, An Optimal Algorithm for the Majority Problem. Manuscript, Southern Methodist University, Texas, 1990.

[10] J. Misra and D. Gries, Finding Repeated Elements. *Science of Computer Programming* **2** (1982) 143-152.

[11] J. Moore, Problem 81-5. *Journal of Algorithms* **2** (1981).

[12] M. E. Saks and M. Werman, On Computing Majority by Comparisons. *Combinatorica* **11** (1991) 383-387.

[13] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics.* Cambridge University Press (1994).

Recent BRICS Report Series Publications

RS-98-34 **Peter G. Binderup, Gudmund Skovbjerg Frandsen, Peter Bro Miltersen, and Sven Skyum.** ***The Complexity of Identifying Large Equivalence Classes*****. December 1998. 15 pp.**

RS-98-33 **Hans Hüttel, Josva Kleist, Uwe Nestmann, and Massimo Merro.** ***Migration = Cloning ; Aliasing (Preliminary Version)*****. December 1998. 40 pp. To appear in** ***6th International Workshop on the Foundations of Object-Oriented*****, FOOL6 Informal Proceedings, 1998.**

RS-98-32 **Jan Camenisch and Ivan B. Damgård.** ***Verifiable Encryption and Applications to Group Signatures and Signature Sharing*****. December 1998. 18 pp.**

RS-98-31 **Glynn Winskel.** ***A Linear Metalanguage for Concurrency*****. November 1998.**

RS-98-30 **Carsten Butz.** ***Finitely Presented Heyting Algebras*****. November 1998. 30 pp.**

RS-98-29 **Jan Camenisch and Markus Michels.** ***Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes*****. November 1998. 19 pp.**

RS-98-28 **Rasmus Pagh.** ***Low Redundancy in Dictionaries with*** $O(1)$ ***Worst Case Lookup Time*****. November 1998. 15 pp.**

RS-98-27 **Jan Camenisch and Markus Michels.** ***A Group Signature Scheme Based on an RSA-Variant.*** **November 1998. 18 pp. Preliminary version appeared in Ohta and Pei, editors,** ***Advances in Cryptology: 4th ASIACRYPT Conference on the Theory and Applications of Cryptologic Techniques*****, ASIACRYPT '98 Proceedings, LNCS 1514, 1998, pages 160–174.**

RS-98-26 **Paola Quaglia and David Walker.** ***On Encoding*** $p\pi$ ***in*** $m\pi$**. October 1998. 27 pp. Full version of paper to appear in** ***Foundations of Software Technology and Theoretical Computer Science: 18th Conference*****, FCT&TCS '98 Proceedings, LNCS, 1998.**

RS-98-25 **Devdatt P. Dubhashi.** ***Talagrand's Inequality in Hereditary Settings*****. October 1998. 22 pp.**

RS-98-24 **Devdatt P. Dubhashi.** ***Talagrand's Inequality and Locality in Distributed Computing*****. October 1998. 14 pp.**