Finding at least one excellent element in two rounds

Gyula O.H. Katona*
Rényi Institute
Budapest, Hungary
ohkatona@renyi.hu

Dedicated to the memory of Jagdish N. Srivastava

Abstract

Suppose that some elements of the set $[n] = \{1, 2, \ldots, n\}$ are excellent. Their number and positions are unknown. Subsets of $[n]$ can be used for tests. If this test is $A \subset [n]$ then the result of the test is YES if at least one of the excellent ones is in A. Otherwise the answer is NO. The goal of the search is to find at least one of the excellent elements, or to claim that there is none. We prove that the number of tests is at least n in the non-adaptive case. On the other hand, if two rounds can be used, then the number of tests in the worst case is at least $(2 + o(1))\sqrt{n}$, and this is sharp.

Keywords: Search problem, two rounds, tranversal

1 Introduction

Suppose that we have n marbles, some of them are radioactive, and we need one radioactive marble. A possible test is taking an arbitrary subset and checks if this set is radiating, that is, if there is at least one radioactive

*The work was supported by the Hungarian National Foundation for Scientific Research grant numbers NK0621321, 048826, 78439.

among them. The mathematical problem, of course, is the minimum number of tests sufficient to find at least one of the radioactive ones.

Let $[n] = \{1, 2, \dots, n\}$ be an n-element set. Some of them are *excellent*. In the traditional model of *Search Theory* one has to find all of these excellent elements from the answers to some questions of type "does the subset $F \subseteq [n]$ contain at least one excellent element?". The following survey papers and books contain many pretty examples, applications and results of this type: [6], [2], [1], [4].

Ray Chambers [3] raised the question what happens if not all excellent elements have to be found, only one. In the present paper we give the first (partial) answer to Chambers' question. Another model of this type is considered in the forthcoming paper [5].

In the case of every search problem there are two essentially different approaches. If the search is *adaptive* then the choice of the subsequent questions may depend on the answers to the previous questions. More precisely, first a set F is tested. If the answer is Y, that is F contains at least one excellent element then a set F_1 is tested afterwards, while in the case when the answer is N, that is F contains no excellent element then another set F_0 is tested. This is repeated. If e.g. the answer to the question F_1 is Y then the next question is F_{11}, and so on. This is continued until the goal of the search is achieved, in the present paper when at least one excellent element is determined, or it is established that there is none. The question sets form a tree-like structure. This is called a *search algorithm*. Of course the number of questions may depend on the actual placement of the excellent elements. The number of questions in the worst case (length of the longest path in the tree) is the *length of the search*. The *(adaptive) complexity of the search problem* is the the minimum of the length of the search taking for all search algorithms solving the given problem.

The non-adaptive search, however, is just a family $\mathcal{F}$ of the question sets. They are all tested and the goal of the search should be achieved based on these answers. The *length of the non-adaptive search* is just $|\mathcal{F}|$. The *non-adaptive complexity of the search problem* is $\min |\mathcal{F}|$ for all families solving the given problem. Since the non-adaptive search is also an adaptive search, the adaptive complexity cannot exceed the non-adaptive complexity for any search problem.

For illustration let us consider now the easiest search problem when it is known that there is exactly one excellent element. Let us denote this problem by $P(1 \to 1)$. The number of possible outcomes is n. At each

step of the (adaptive) algorithm we can exclude at most half of the possible outcomes, the adaptive complexity $A(n)$ of $P(1 \to 1)$ satisfies the inequality $\lceil \log n \rceil \leq A(n)$. Now we give a non-adaptive search consisting of the same number of subsets. For convenience replace the elements $1 \leq j \leq n$ of the underlying set by $0 \leq j-1 \leq n-1$. Write them in a binary form of length $\lceil \log n \rceil$. The set F_i contains those elements of $\{0, \ldots, n-1\}$ which have a 1 in the binary form of i. The family $\{F_1, \ldots, F_{\lceil \log n \rceil}\}$ really solves the search problem since the answers for these questions determine all the entries in the binary form of the excellent element. This gives an upper estimate on the non-adaptive complexity $N(n)$ of $P(1 \to 1)$. Summarizing, we have

$$\lceil \log n \rceil \leq A(n) \leq N(n) \leq \lceil \log n \rceil,$$

giving a complete solution for this case.

Let us go back to the main search problem of the present paper. The number of excellent elements is unknown, our goal is either to find one excellent element or to declare that there is none. This problem is denoted by $P(? \to 1)$. The total number of possible outcomes is $n+1$, therefore the adaptive complexity $\alpha(n)$ of $P(? \to 1)$ is at least $\log(n+1)$. Surprisingly, the method above also works for this case, (with a slight modification).

Theorem 1.1 $\alpha(n) = \lceil \log(n+1) \rceil$.

Proof. An adaptive search algorithm will be given with maximum length $\lceil \log(n+1) \rceil$.

The algorithm will be slightly different when n is a power of 2. Then move, again, the underlying set to the interval $\{0, 1, \ldots, n-1\}$, and consider the binary form (of length $\log n$) of these integers. The initial question set F consists of the integers having 1 in the first position of their binary form. If the answer is Y then all other questions will be subsets of F. The next question in this case is F_1 consisting of the integers having 1 in the first two positions. If the answer to the first question F is N then the next question is F_0 which consists of the integers containing 01 in the first two positions. Continue in this way, choosing the next question set to the subset of the previous one and having 1 in the next position. (Less formally we always halve the set of the elements which can be excellent.) After $\log n$ steps, if one of the answers was Y then we have arrived to a unique element with a binary form containing at least one 1. However, if all the answers were N, then we have arrived to the number 0, we had no information if this was

excellent or not. In this case the last question is $F^* = \{0\}$. The answer decides if 0 is excellent or there is none. The number of steps, the length of the search algorithm is $\log n + 1$ what is equal to $\lceil \log(n+1) \rceil$ in this case.

Suppose now that n is not a power of 2. Then the original underlying set $[n]$ can be considered, since the binary forms of these numbers have a length $\lceil \log n \rceil$. Use the same algorithm as before. If at least one the answers was Y then a unique excellent element has been determined. Otherwise we can claim that there is no excellent element, since in this case there is no element with binary form containing only 0. Of course $\lceil \log n \rceil = \lceil \log(n+1) \rceil$. □

The situation is dramatically different in the case of non-adaptive search. We will show in Section 2 that the non-adaptive complexity of this problem is n. Observe that the additional information of knowing that there is exactly one excellent element reduces this to $\lceil \log(n+1) \rceil$!

There is an intermediate approach between the adaptive and the non-adaptive search algorithms. When the number of "rounds" is bounded. A search with two rounds will be considered in Section 3. In the first round one asks all members of a family $\mathcal{F}$, then, depending the sequence of answers the members of another family $\mathcal{G}$ are asked. The length of this two-round search is $|\mathcal{F}|$ plus the largest $|\mathcal{G}|$. It will be shown in Section 3 that its minimum for our problem is of order $\sqrt{n}$.

2 The non-adaptive case

It is easy to characterize the families $\mathcal{F}$ which are non-adaptive algorithms for the problem $P(1 \to 1)$. The necessary and sufficient condition is that for any pair x, y of distinct elements of $[n]$ there is a member $F \in \mathcal{F}$ such that it separates them, that is either $x \in F, y \notin F$ or $x \notin F, y \in F$ holds.

This characterization is less trivial for the problem $P(? \to 1)$. The following definition will be useful towards this end. We say that the family $\mathcal{T} \subseteq 2^{[n]}$ has a *fixed point* if there is one element contained in every member of $\mathcal{T}$:

$$\bigcap_{T \in \mathcal{T}} T \neq \emptyset.$$

Lemma 2.1 *The non-empty family $\mathcal{F} \subseteq 2^{[n]}$ is a non-adaptive algorithm for the problem $P(? \to 1)$ iff the following two conditions hold.*

$$(i) \bigcup_{F \in \mathcal{F}} F = [n],$$

(ii) for every partition $\mathcal{F} = \mathcal{F}_1 \cup F_2(\mathcal{F}_1 \neq \emptyset)$ *the family*

$$\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2) = \{T : \ T \cap F \neq \emptyset \text{ for every } F \in \mathcal{F}_1, T \cap F = \emptyset \text{ for every } F \in \mathcal{F}_2\}$$

has a fixed point, that is

$$\bigcap_{T \in \mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)} T \neq \emptyset. \tag{2.1}$$

Proof. Before starting the proof, let us mention that if the choice of members of $\mathcal{F}_1$ is "contradictory" (say e.g. when a member of $\mathcal{F}_1$ is a subset of a member of $\mathcal{F}_2$ then $\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)$ is empty. Then the empty intersection in (2.1) is interpreted as $[n]$, (2.1) automatically holds.

The necessity of (i) is trivial: if $x \in [n]$ is not covered by any of the members of $\mathcal{F}$ then the answers cannot decide if x is excellent or not.

To prove the necessity of (ii) suppose that the answer is Y for the members of $\mathcal{F}_1$ and N for the members of $\mathcal{F}_2$. Then we know that the (non-empty) set T of excellent elements intersects every member of $\mathcal{F}_1$ and disjoint to every member of $\mathcal{F}_2$. On the other hand any such set can be the set of excellent elements. That is the family of possible sets of excellent elements is $\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)$. If (2.1) does not hold, no element is surely excellent.

On the other hand, if both (i) and (ii) hold then we can find the solution. If all the answers are N then we know by (i) that there is no excellent element. Otherwise define $\mathcal{F}_1$ as the family of sets with answer Y where $\mathcal{F}_1 \neq \emptyset$. The possible set of excellent elements must be in $\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)$. Since (2.1) is non-empty for this family, it contains an element f, which is an element of every possible set of excellent elements, it must be excellent. □

A *transversal* $T \subseteq [n]$ of the family $\mathcal{F} \subseteq 2^{[n]}$ is a set satisfying $F \cap T \neq \emptyset$ for every $F \in \mathcal{F}$. The family of all transversals of $\mathcal{F}$ is denoted by $\mathcal{T}(\mathcal{F})$. the following proposition is trivial, but might have some interest on its own right, independently on search problems.

Proposition 2.2 *Let* $\emptyset \neq \mathcal{F} \subseteq 2^{[n]}, \{\emptyset\} \notin \mathcal{F}$. *If* $\mathcal{T}(\mathcal{F})$ *has a fixed point then* $\mathcal{F}$ *has a one-element member.*

Proof. Let a be a fixed point of $\mathcal{T}(\mathcal{F})$. Then $[n] - \{a\}$ cannot be a member of $\mathcal{T}(\mathcal{F})$. It meets every subset of $[n]$ except for $\{a\}$, therefore $\{a\}$ must be a member of $\mathcal{F}$. □

The following lemma is an immediate consequence.

Lemma 2.3 *If the non-empty family $\mathcal{F} \subseteq 2^{[n]}, \{\emptyset\} \notin \mathcal{F}$ is a non-adaptive algorithm for the problem $P(? \to 1)$ then it contains a one-element member.*

Proof. Use (ii) of Lemma 2.1 with $\mathcal{F}_1 = \mathcal{F}$. The family $\mathcal{T}(\mathcal{F}, \emptyset) = \mathcal{T}(\mathcal{F})$ has a fixed point. Proposition 2.2 finishes the proof. □

If $\mathcal{F}$ is a family of subsets, $f \in [n]$, let $\mathcal{F} - f$ denote the family $\{F - f : F \in \mathcal{F}, f \in F, |F| \geq 2\}$.

Lemma 2.4 *If the non-empty family $\mathcal{F} \subseteq 2^{[n]}, \{\emptyset\} \notin \mathcal{F}$ is a non-adaptive algorithm for the problem $P(? \to 1)$ on the set $[n]$, $\{f\} \in \mathcal{F}$ holds then $\mathcal{F} - f$ is a non-adaptive algorithm for the problem $P(? \to 1)$ on the set $[n] - \{f\}$.*

Proof. Choose f according to Lemma 2.3. Use (ii) of Lemma 2.1 under the assumption $\{f\} \in \mathcal{F}_2$. Then the members of $\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)$ do not contain f. It is easy to see that $\mathcal{T}(\mathcal{F}_1, \mathcal{F}_2)$ is the same as $\mathcal{T}(\mathcal{F}_1 - f, \mathcal{F}_2 - f)$ (restricted to $[n] - f$). Therefore (ii) holds for the family $\mathcal{F} - f$ on $[n] - \{f\}$. It is trivial that (i) also holds. Hence, using Lemma 2.1 once more (backwards), the statement of lemma is obtained. □

Theorem 2.5 *If $\mathcal{F}$ is a non-adaptive algorithm for the problem $P(? \to 1)$ on $[n]$ then $|\mathcal{F}| \geq n$.*

Proof. Use induction on n. The case $n = 1$ is trivial. Suppose $n > 1$. By Lemma 2.4 we know that $\mathcal{F} - f$ is a non-adaptive algorithm for the problem $P(? \to 1)$ on $[n] - \{f\}$, by the inductional hypothesis $|\mathcal{F}| - 1 \geq n - 1$ implying the statement for $\mathcal{F}$ and n. □

3 Two rounds

The search algorithm with two rounds consists of a family $\mathcal{F} \subseteq 2^{[n]}, |\mathcal{F}| = m$ and a set of families $\mathcal{G}(s) \subseteq 2^{[n]}$ defined for every $s \in \{\mathrm{N}, \mathrm{Y}\}^m$. The members of $\mathcal{F}$ are asked in the first round. If the sequence of answers is s then the members of $\mathcal{G}(s)$ are asked in the second round, and either an excellent element is found after all the answers or these answers show that there is none. We say that $(\mathcal{F}, \{\mathcal{G}(s) : s \in \{\mathrm{N}, \mathrm{Y}\}^{|\mathcal{F}|}\}, [n])$ is a *search algorithm with two rounds*. (Later, in some cases $[n]$ will be replaced by some $X \subset [n]$. Then, obviously, $\mathcal{F} \subseteq 2^X$ is supposed.) The length of this algorithm is

$$|\mathcal{F}| + \max_s |\mathcal{G}(s)|. \tag{3.1}$$

The *two-round complexity* of a search problem is the minimum of the lengths of the algorithms solving the given problem. It is obvious that the two-round complexity is between the adaptive and non-adaptive complexities. Therefore the two round complexity of $P(? \to 1)$ is at least $\lceil \log n \rceil$ and and not more than n.

The goal of the present section is to show that the two-round complexity $\tau(n)$ of $P(? \to 1)$ is different from both the adaptive and non-adaptive complexities and has the order $\sqrt{n}$. The proof is broken into several lemmas.

Suppose that the answers for the members of $\mathcal{F}_{\rm Y} \subseteq \mathcal{F}$ are Y, while the answers for the members of $\mathcal{F}_{\rm N} \subseteq \mathcal{F}$ are N. (Here $\mathcal{F}_{\rm Y} \cup \mathcal{F}_{\rm N} = \mathcal{F}$ and $\mathcal{F}_{\rm Y} \cap \mathcal{F}_{\rm N} = \emptyset$ are obvious.) Then we know that every member of $\mathcal{F}_{\rm Y}$ contains an excellent element and the members of $\mathcal{F}_{\rm N}$ contain none. At least one excellent element should be found using this information. This problem will be denoted by $P((?, \mathcal{F}_{\rm Y}, \mathcal{F}_{\rm N}) \to 1)$. The second round in our two-round algorithm is a family $\mathcal{G}(s)$ where s is determined by the families $\mathcal{F}_{\rm Y}$ and $\mathcal{F}_{\rm N}$. This part of the two-round algorithm for solving the problem $P(? \to 1)$ is a (one-round) non-adaptive algorithm for the problem $P((?, \mathcal{F}_{\rm Y}, \mathcal{F}_{\rm N}) \to 1)$.

The series of lemmas is starting with a generalization of Lemma 2.1. The generalization is broken into two parts according to whether $\mathcal{F}_{\rm Y}$ is empty or not.

Lemma 3.1 *Let $\mathcal{F}_{\rm Y} \neq \emptyset, \mathcal{F}_{\rm N} \subseteq 2^{[n]}, \mathcal{F}_{\rm Y} \cap \mathcal{F}_{\rm N} = \emptyset$. The non-empty family $\mathcal{G}$ is a non-adaptive algorithm for the problem $P((?, \mathcal{F}_{\rm Y}, \mathcal{F}_{\rm N}) \to 1)$ iff the following condition holds. For every partition $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ the family*

$$\mathcal{T}(\mathcal{F}_{\rm Y}, \mathcal{F}_{\rm N}, \mathcal{G}_1, \mathcal{G}_2) =$$

$$\{T : \ T \cap F \neq \emptyset \text{ for every } F \in \mathcal{F}_{\rm Y}, T \cap G \neq \emptyset \text{ for every } G \in \mathcal{G}_1,$$

$$T \cap F = \emptyset \text{ for every } F \in \mathcal{F}_{\rm N}, \ T \cap G = \emptyset \text{ for every } G \in \mathcal{G}_2\}$$

has a fixed point, that is

$$\bigcap_{T \in \mathcal{T}(\mathcal{F}_{\rm Y}, \mathcal{F}_{\rm N}, \mathcal{G}_1, \mathcal{G}_2)} T \neq \emptyset. \tag{3.2}$$

Proof. If answer Y is obtained for the members of $\mathcal{G}_1$ and N for the members of $\mathcal{G}_2$ then the set of excellent elements is a member of $\mathcal{T}(\mathcal{F}_\mathrm{Y}, \mathcal{F}_\mathrm{N}, \mathcal{G}_1, \mathcal{G}_2)$ and every member can be the set of excellent elements. If (3.2) does not hold, then no element can be identified as excellent. Therefore it must hold. The opposite direction of the proof is trivial: if $\mathcal{T}(\mathcal{F}_\mathrm{Y}, \mathcal{F}_\mathrm{N}, \mathcal{G}_1, \mathcal{G}_2)$ is non-empty then (3.2) determines at least one excellent element. If it is empty then the answers are contradicting. □

Lemma 3.2 *Let $\mathcal{F}_\mathrm{Y} = \emptyset, \mathcal{F}_\mathrm{N} \subseteq 2^{[n]}$. The non-empty family $\mathcal{G}$ is a non-adaptive algorithm for the problem $P((?, \emptyset, \mathcal{F}_\mathrm{N}) \to 1)$ iff the following conditions hold.*

$$(iii) \bigcup_{G\in\mathcal{G}} G \supseteq [n] - \bigcup_{F\in\mathcal{F}_\mathrm{N}} F,$$

(iv) For every partition $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ the family

$$\mathcal{T}(\emptyset, \mathcal{F}_\mathrm{N}, \mathcal{G}_1, \mathcal{G}_2) = \{T : \ T \cap G \neq \emptyset \text{ for every } G \in \mathcal{G}_1,$$

$$T \cap F = \emptyset \text{ for every } F \in \mathcal{F}_\mathrm{N}, \ T \cap G = \emptyset \text{ for every } G \in \mathcal{G}_2\}$$

has a fixed point, that is

$$\bigcap_{T\in\mathcal{T}(\emptyset,\mathcal{F}_\mathrm{N},\mathcal{G}_1,\mathcal{G}_2)} T \neq \emptyset. \tag{3.3}$$

Proof is the same as before. □

If $\mathcal{H} \subseteq 2^{[n]}, C \subset [n]$ then let $\mathcal{H} - C$ be the following family of subsets of $[n] - C$: $\mathcal{H} - C = \{H - C : \ H \in \mathcal{H}, H \not\subseteq C\}$. (In the case when $C = \{c\}$ then $\mathcal{H} - c$ is simply denoted by $\mathcal{H} - c$.)

Lemma 3.3 *If the non-empty family $\mathcal{G}$ is a non-adaptive algorithm for the problem $P((?, \mathcal{F}_\mathrm{Y}, \mathcal{F}_\mathrm{N}) \to 1)$ on the underlying set $[n]$, moreover $C = \bigcup_{F\in\mathcal{F}_\mathrm{N}} F$ then $\mathcal{G} - C$ is a non-adaptive algorithm for the problem $P((?, \mathcal{F}_\mathrm{Y} - C, \emptyset) \to 1)$ on the underlying set $[n] - C$.*

Proof. Suppose that $\mathcal{F}_\mathrm{Y} \neq \emptyset$. Then (3.2) holds for $\mathcal{T}(\mathcal{F}_\mathrm{Y}, \mathcal{F}_\mathrm{N}, \mathcal{G}_1, \mathcal{G}_2)$ for every partition of $\mathcal{G}$. The members of cannot have elements in $C = \bigcup_{F\in\mathcal{F}_\mathrm{N}} F$, therefore $\mathcal{T}(\mathcal{F}_\mathrm{Y}, \mathcal{F}_\mathrm{N}, \mathcal{G}_1, \mathcal{G}_2) = \mathcal{T}(\mathcal{F}_\mathrm{Y} - C, \emptyset, \mathcal{G}_1 - C, \mathcal{G}_2 - C)$ and (3.2) holds

for this reduced family of subsets of $[n]-C$. Applying Lemma 3.1 again, we obtain the statement of the lemma, since every partition of $\mathcal{G}-C$ can be obtained in the form of $(\mathcal{G}_1-C)\cup(\mathcal{G}_2-C)$ where $\mathcal{G}_1\cup\mathcal{G}_2$ is a partition on $\mathcal{G}$.

In the other case, when $\mathcal{F}_{\mathrm{Y}}=\emptyset$, Lemma 3.2 can be used. Then (iii) implies

$$\bigcup_{G\in\mathcal{G}-C} G \supseteq [n]-C.$$

Moreover, (3.3) holds for $\mathcal{T}(\emptyset,\mathcal{F}_{\mathrm{N}},\mathcal{G}_1,\mathcal{G}_2)=\mathcal{T}(\emptyset,\emptyset,\mathcal{G}_1-C,\mathcal{G}_2-C)$. Another application of Lemma 3.2 (or simply Lemma 2.1) proves the statement. □

Lemma 3.4 *Let $\mathcal{F}_{\mathrm{Y}}\neq\emptyset$. If the non-empty family $\mathcal{G}$ ($\{\emptyset\}\notin\mathcal{F}_{\mathrm{Y}}\cup\mathcal{G}$) is a non-adaptive algorithm for the problem $P((?,\mathcal{F}_{\mathrm{Y}},\emptyset)\to 1)$ then there is an element $a\in[n]$ such that $\{a\}\in\mathcal{F}_{\mathrm{Y}}\cup\mathcal{G}$.*

Proof. By Lemma 3.1 $\mathcal{T}(\mathcal{F}_{\mathrm{Y}},\emptyset,\mathcal{G},\emptyset)=\mathcal{T}(\mathcal{F}_{\mathrm{Y}}\cup\mathcal{G})$ has a fixed point. Proposition 2.2 implies the statement. □

Lemma 3.5 *Suppose $\mathcal{F}_{\mathrm{Y}}\neq\emptyset$. If the non-empty family $\mathcal{G}$ is a non-adaptive algorithm for the problem $P((?,\mathcal{F}_{\mathrm{Y}},\emptyset)\to 1)$ on the underlying set $[n]$, moreover $\{a\}\in\mathcal{G}$ then $\mathcal{G}-a$ is a non-adaptive algorithm for the problem $P((?,\mathcal{F}_{\mathrm{Y}}-a,\emptyset)\to 1)$ on the underlying set $[n]-\{a\}$.*

Proof. Apply Lemma 3.1 with $\{a\}\in\mathcal{G}_2$. Then the members of $\mathcal{T}(\mathcal{F}_{\mathrm{Y}},\emptyset,\mathcal{G}_1,\mathcal{G}_2)$ do not contain a. Therefore $\mathcal{T}(\mathcal{F}_{\mathrm{Y}},\emptyset,\mathcal{G}_1,\mathcal{G}_2)=\mathcal{T}(\mathcal{F}_{\mathrm{Y}}-a,\emptyset,\mathcal{G}_1-a,\mathcal{G}_2-a)$ holds for this reduced family of subsets of $[n]-\{a\}$. Applying Lemma 3.1 again, we obtain the statement of the lemma, since every partition of $\mathcal{G}-a$ can be obtained in the form of $(\mathcal{G}_1-a)\cup(\mathcal{G}_2-a)$ where $\mathcal{G}_1\cup\mathcal{G}_2$ is a partition on $\mathcal{G}$. □

Lemma 3.6 *Suppose $\mathcal{F}_{\mathrm{Y}}\neq\emptyset$. If the non-empty family $\mathcal{G}$ is a non-adaptive algorithm for the problem $P((?,\mathcal{F}_{\mathrm{Y}},\emptyset)\to 1)$ on the underlying set $[n]$ then*

$$\min_{F\in\mathcal{F}_{\mathrm{Y}}}|F|-1\leq|\mathcal{G}|$$

holds.

Proof. Use induction on n. The case $n = 1$ is trivial. Suppose that the statement is true for $n - 1$ and prove it for n. By Lemma 3.4 there is an $\{a\} \in \mathcal{F}_{\mathrm{Y}} \cup \mathcal{G}$. If $\{a\} \in \mathcal{F}_{\mathrm{Y}}$ then the smallest size in $\mathcal{F}_{\mathrm{Y}}$ is 1, the statement of the lemma is trivial. $\{a\} \in \mathcal{G}$ can be supposed. Use Lemma 3.5. The size of the smallest set in $\mathcal{F}_{\mathrm{Y}} - a$ is either the same as in $\mathcal{F}_{\mathrm{Y}}$ or less by one. Hence we have

$$\min_{F \in \mathcal{F}_{\mathrm{Y}}} |F| - 2 \leq |\mathcal{G} - a|,$$

and $|\mathcal{G} - a| = |\mathcal{G}| - 1$ finishes the proof. □

If $\mathcal{F}$ is a family of k members, define $H_1 = F_1$ as (one of) the smallest member(s). Let F_2 be one of the members of $\mathcal{F}$ minimizing $\{|F - F_1| : \ F \in \mathcal{F}, F \neq F_1\}$ and introduce the notation $H_2 = F_2 - F_1$. Define F_3 as one of the members of $\mathcal{F}$ minimizing $\{|F - (F_1 \cup F_2)| : \ F \in \mathcal{F}, F \neq F_1, F_2\}$. Again, let $H_3 = F_3 - (F_1 \cup F_2)$. The sets $F_4, \ldots, F_k$ and $H_4, \ldots, H_k$ are defined similarly. Finally, the family $\mathcal{H}(\mathcal{F})$ is simply $\{H_1, H_2, \ldots, H_k\}$.

Lemma 3.7 *The members of $\mathcal{H}(\mathcal{F})$ are pairwise disjoint. Moreover*

$$\bigcup_{i=1}^{k} H_i = \bigcup_{i=1}^{k} F_i$$

holds.

Proof. The first part of the lemma is a trivial consequence of the definitions. To see the second part use induction on j to prove

$$\bigcup_{i=1}^{j} H_i = \bigcup_{i=1}^{j} F_i.$$

□

Lemma 3.8 *Let $(\mathcal{F}, \{\mathcal{G}(s) : \ s \in \{\mathrm{N}, \mathrm{Y}\}^{|\mathcal{F}|}\}, [n])$ be a search algorithm with two rounds and use the notation $A = \bigcup_{F \in \mathcal{F}}$. Then*

$$\max\{|H| - 1 : \ H \in \mathcal{H}(\mathcal{F}), n - |A|\} \leq \max_{s} |\mathcal{G}(s)|. \tag{3.4}$$

Proof. If $s = (\mathrm{N}, \mathrm{N}, \ldots, \mathrm{N})$ then $\mathcal{G}(s)$ is a non-adaptive algorithm for the problem $P((?, \emptyset, \mathcal{F}) \to 1)$. By Lemma 3.3 $\mathcal{G} - A$ is a non-adaptive algorithm

for the problem $P((?, \emptyset, \emptyset) \to 1)$ on the underlying set $[n] - A$. However, $P((?, \emptyset, \emptyset) \to 1)$ is simply $P(? \to 1)$ and Theorem 2.5 results in

$$n - |A| \leq |\mathcal{G}((\mathrm{N}, \mathrm{N}, \ldots, \mathrm{N}))|. \tag{3.5}$$

Suppose that $s = (\mathrm{N}, \mathrm{N}, \ldots, \mathrm{N}, \mathrm{Y}, \ldots, \mathrm{Y})$ where the number of answers N is $j (0 \leq j < k = |\mathcal{F}|)$. Here $\mathcal{G}(s)$ is a non-adaptive algorithm for the problem $P((?, \{F_{j+1}, \ldots, F_k\}, \{F_1, \ldots, F_j\}) \to 1)$. Let $C_j = \bigcup_{i=1}^{j} F_j$. By Lemma 3.3 $\mathcal{G} - C_j$ is a non-adaptive algorithm for the problem $P((?, \{F_{j+1}, \ldots, F_k\} - C_j, \emptyset) \to 1)$ on the underlying set $[n] - C_j$. Observe that, by definition, the smallest size in the family $\{F_{j+1}, \ldots, F_k\} - C_j$ is $|H_{j+1}|$. By Lemma 3.6

$$|H_{j+1}| - 1 \leq |\mathcal{G}(s)|. \tag{3.6}$$

Take the largest lower bound obtained from (3.5) and (3.6). □

Theorem 3.9 *The two-round complexity of the problem $P(? \to 1)$ on n elements is*

$$\tau(n) = \min_{k \geq 1, \text{integer}} \left\lfloor k + \frac{n}{k+1} \right\rfloor. \tag{3.7}$$

Proof. Suppose that the first round of the search consists of the members of the family $\mathcal{F}$ where $|\mathcal{F}| = k$. Add the sizes of the sets occurring on the left hand side of (3.4):

$$\sum_{i=1}^{k} (|H_i| - 1) + n - |A|. \tag{3.8}$$

Using Lemma 3.7 this can be written in the form

$$|\bigcup_{i=1}^{k} H_i| - k + n - |A| = |A| - k + n - |A| = n - k. \tag{3.9}$$

(3.8) is a sum of $k+1$ numbers, one of them must be at least

$$\left\lceil \frac{n-k}{k+1} \right\rceil = \left\lfloor \frac{n}{k+1} \right\rfloor, \tag{3.10}$$

by (3.9). Since (3.1) is the complexity of this two-round search algorithm, k should be added to (3.10) and (3.7) is obtained as a lower bound for $\tau(n)$.

It is easy to give an algorithm reaching this bound. Take a k minimizing (3.7) and divide $n+1$ by $k+1$ in the following way: $n+1 = q(k+1)+r(0 < r \leq k+1)$. Then $n = qk + r - 1 + q$. Consider the partition of $[n]$ into $k+1$ classes, where $|F_1| = \ldots |F_{r-1}| = q+1, |F_r| = \ldots = |F_k| = q$, and the $(k+1)$'st one, B has also size q. One can verify that q is equal to (3.10).

In the first round $F_1, \ldots, F_k$ are asked. If all answers are N then ask the elements of the B one by one in the second round. This is $k+q$ questions. If one of the answers is Y then ask the elements of that F one by one until only one remains unasked. If one of the answers is Y then an excellent element is found. If the answers are N for all the elements of F and only one element is left, we do not have to ask it since the answer Y for F makes it sure that this last element is excellent. It is easy to see that the number of questions is at most $k+q$ in these cases. □

Consequence 1

$$\left\lfloor 2\sqrt{n} - 1 \right\rfloor \leq \tau(n) \leq \left\lfloor 2\sqrt{n} - \frac{1}{2} \right\rfloor .$$

Proof. By differentiation one can see that the function $f(x) = x + \frac{n}{x+1}$ has its minimum at $\sqrt{n} - 1$ and it is convex in the interval $[\sqrt{n} - 2, \sqrt{n}]$ if $n > 1$. Therefore the integer x minimizing $f(x)$ is in the interval $[\sqrt{n} - \frac{3}{2}, \sqrt{n} - \frac{1}{2}]$ and this minimum is between $f(\sqrt{n} - 1) = 2\sqrt{n} - 1$ and $\max\{f(\sqrt{n} - \frac{3}{2}), f(\sqrt{n} - \frac{1}{2})\} = f(\sqrt{n} - \frac{3}{2}) = \sqrt{n} - \frac{3}{2} + \frac{n}{\sqrt{n} - \frac{1}{2}} \leq 2\sqrt{n} - \frac{1}{2}$. □

4 A remark and a conjecture

The definition of the r-round search algorithm is obvious. The r-round complexity of the problem $P(? \to 1)$ on $[n]$ is denoted by $\tau_r(n)$. The notation $P((? \geq 1) \to 1)$ stands for the problem when the only information about the number of excellent elements is that there is at least one.

Conjecture 1 *If r is fixed, n tends to infinity then*

$$\tau_r(n) \sim r n^{\frac{1}{r}}.$$

The conjectured shortest algorithm is recursively defined in the following way. Start with a partition of $[n]$ consisting of almost equal parts and let the first

round ask the members of the partition except one class C which is one of the smaller ones. If all the answers are N then use the optimal $r-1$-round algorithm for the problem $P(? \to 1)$ on C. on the other hand, if the answer is Y for one question in the first round then use the optimal $r-1$-round algorithm for the problem $P((? \geq 1) \to 1)$ on the corresponding part of the partition.

There is an old theorem of Peter Ungar [7] which is resembling our Theorem 2.5. It also states that "if the number of excellent elements might be large" then there is no better algorithm than check every element one by one. His model is, however, different from ours. Let $0 < p < 1$ be a probability. Every element of $[n]$ can be excellent independently, with probability p. The goal is to find all excellent elements with an adaptive algorithm. Of course the length of the algorithm depends on the random event: how many and which element are excellent. The (probabilistic) average number of questions for a given algorithm is denoted by $\ell_{\mathrm{A}}(n)$. Its minimum for all algorithms finding all excellent elements from $[n]$ is $L(n)$. Ungar's theorem claims that $L(n) \geq n$ if $\frac{1}{3}(3-\sqrt{5}) \leq p$.

References

[1] Ahlswede, R., Wegener, I., 1987. Search Problems, John Wiley.

[2] Aigner, M., 1988. Combinatorial Search, John Wiley and Teubner.

[3] Chambers, Ray, personal communication.

[4] Du Dingzhu, Hwang, Frank K., 2000. Combinatorial Group Testing and its Applications, Second edition, World Scientific, 2000.

[5] Gerbner D., Keszegh B., Pálvölgyi D., Wiener, G., Search with density tests, manuscript.

[6] Katona G.O.H., 1973. Combinatorial Search Problems, in: A survey of combinatorial theory, J.N.Srivastava (Ed.), North Holland/American Elsevier, Amsterdam/New York, pp. 285-308.

[7] Ungar, P., 1960. The cut-off point for group testing. Commun. Pure Appl. Math. 13, 49-54.