

Consider an algorithm A .

- We use $M_A(d, n)$ to denote the "worst-case" number of tests required by A , namely, the maximum number of tests over all sample n points with d positives. If d is an upper bound, then we use $M_A(\bar{d}, n)$.
(exactly)
- An algorithm is reasonable if it contains no test whose outcome can be predicted from the outcomes of other tests either conducted "previously or simultaneously".

For example, if S is tested with negative outcome, then no proper subset of S should be tested again. (Except the cases when errors are possible.)

- Define $M(d, n) = \min_A M_A(d, n)$.
- We can also consider the sample space of a group testing problem S .

$$M_A(S), M(S) = \min_A M_A(S).$$

- An algorithm A is called a minimax algorithm for S if $M_A(S) = M(S)$.

Fact 1 $M(S) \geq \lceil \log |S| \rceil$, log stands for log₂.

Proof. Since we have $|S|$ sample points to start with and each test

divides the sample points into two disjoint sets, the proof follows. ■

(*) Fact 1 is the well-known "information theory bound".
or simply informatic bound.

- Fact 2. $S \subseteq S' \Rightarrow M(S) \leq M(S') \rightarrow$ Every algorithm for S' is also an algorithm for S except we may remove some "unreasonable" algorithm for S .
- Fact 3. $n \leq n' \Rightarrow M(d, n) \leq M(d, n')$.
- Fact 4. $M(1, n) = \lceil \log_2 n \rceil$.

Open problem Determine $M(2, n)$.

- The fundamental group testing problem is to determine $M(d, n)$.

Exercise 1. Determine $M(2, 6)$ and $M(\bar{2}, 6)$.

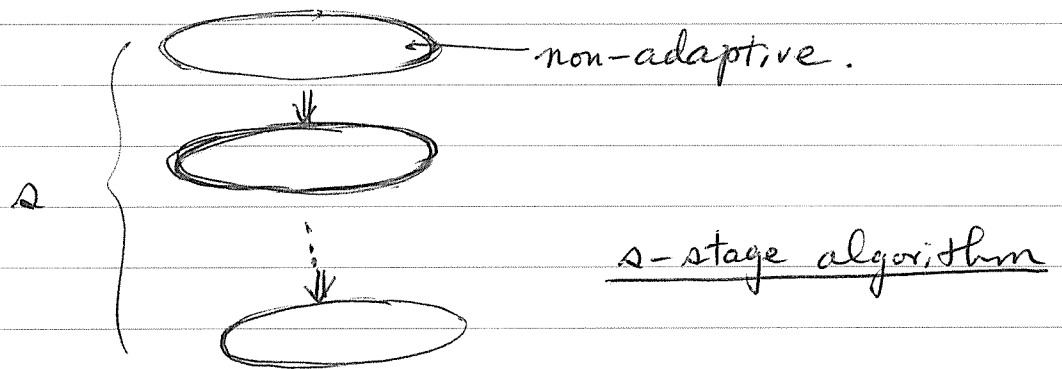
Fact 5. $M(\bar{d}, n) \leq M(d, n) + 1$. [Hwang et al., 1981]

Exercise A. Verify Fact 5.

There are two general types of group testing algorithms. In a "sequential" algorithm, the tests are conducted "one by one", and the outcomes of previous tests are assumed known at the time of determining the current test. In a "non-adaptive" algorithm, no such information is available in determining a test. Therefore, we can conduct all the tests simultaneously. (Save time!?)

A compromise between them is the so-called "multistage"

algorithm: Tests are divided into several stages where the stages are considered sequential but all tests in the same stage are treated as non-adaptive.



Thus, a test can use outcomes of tests conducted in previous stages, but not those in the same stage.

Fact 6. (Nothing to prove!)

If an item can go through "only" one test, then individual testing is the only feasible algorithm. On the other hand, if an item can go through at most s tests, then an s -stage algorithm can be applied. See [Li] for instance. (Industrial applications)

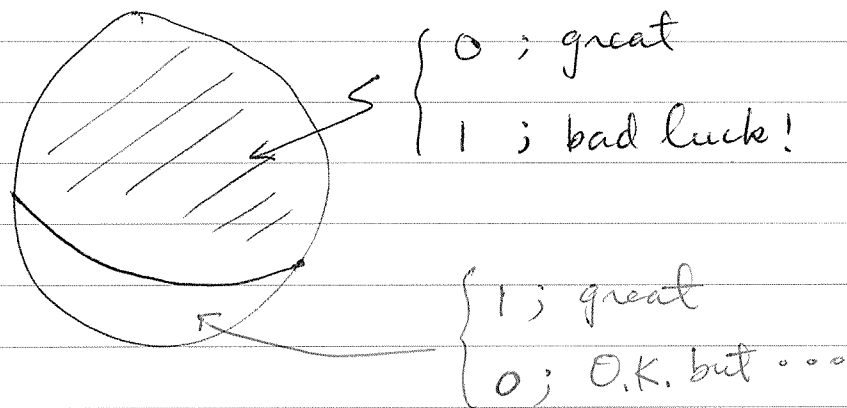
C. H. Li, A sequential method for screening experimental variables,
J. Amer. Statist. Assoc. 57 (1962), 455-477.

Group Testing with constraints

1. Each group has a size limitation, say k .
2. The items are arranged as the vertices of a "graph", only certain subgraphs (sub-groups) can be tested, for example, a path.
3. The outcome of a test can be generalized to k -ary outcome $(0, 1, 2, \dots, k-1)$, i^+ denotes at least i positives. The original outcome can be recognized as a binary outcome: $(0, 1^+)$.
Note that a ternary (3-ary) outcome has been used in the communication (computer) which contains multiaccess channels.
4. The outcomes may contain "errors".
5. There are thresholds for being positive or negative.
6. There are items which play the role "bad" item: inhibitors.
7. Many others.
8.  See Introduction for more details.

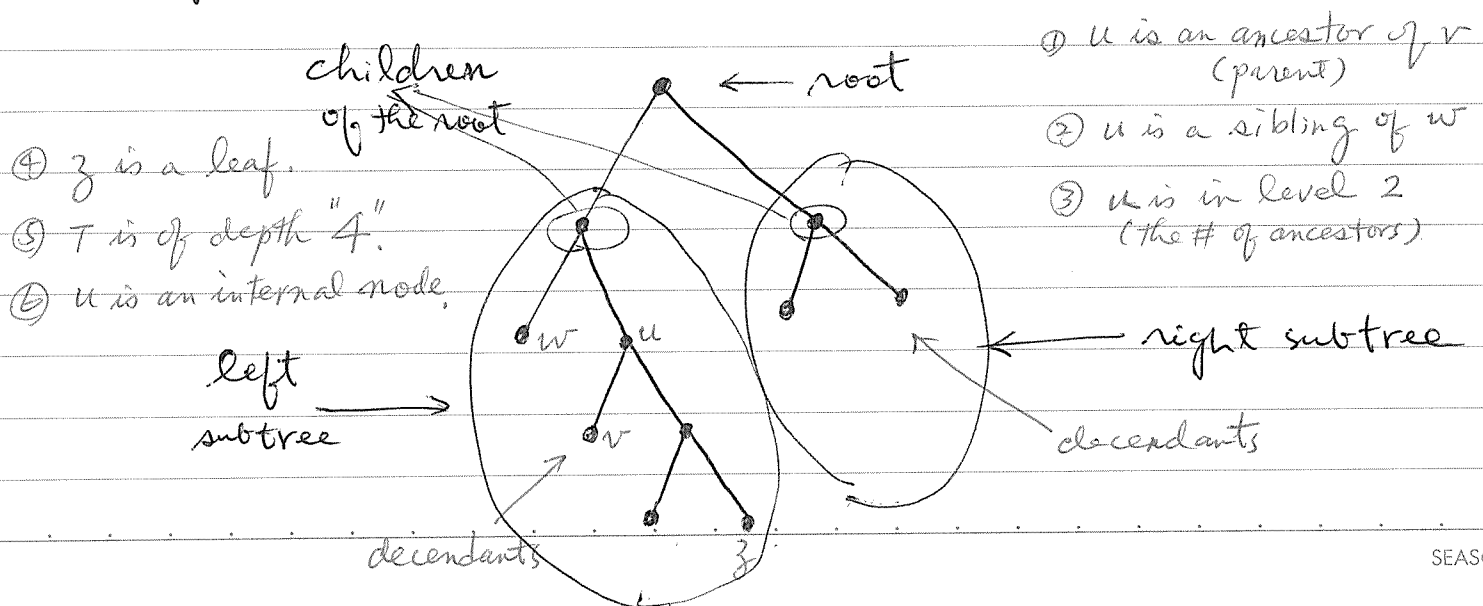
General Sequential Algorithms

- Group testing takes advantage of that by identifying groups containing no positives.
- So, how large a "group" should be?

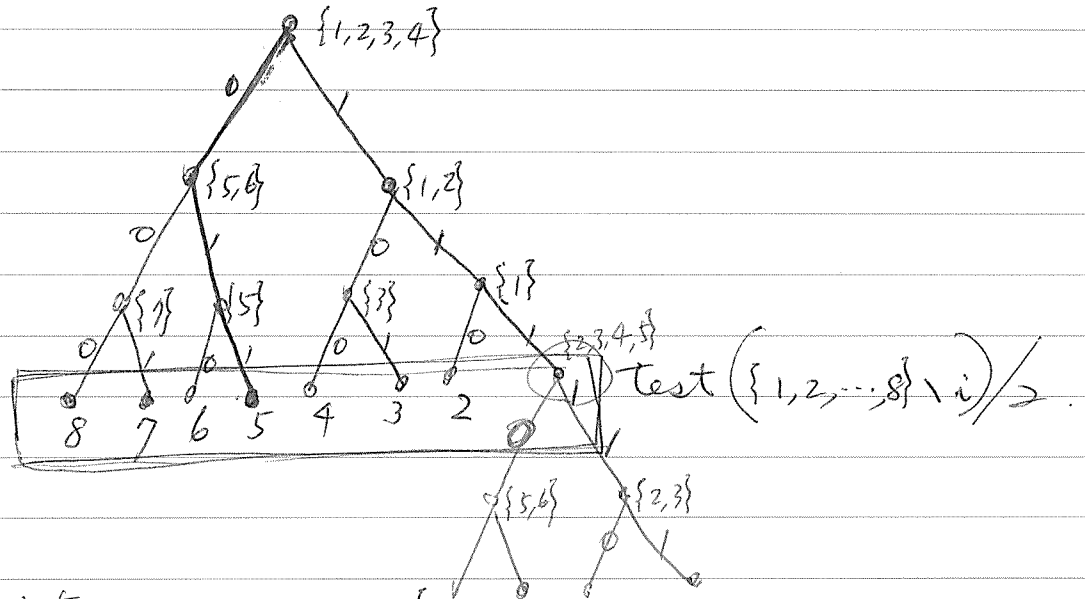


- Keeping a balance between these two conflicting goals is what most algorithms strive for.

(*) A classical GT can be represented by a binary tree (Sequential Algorithm)

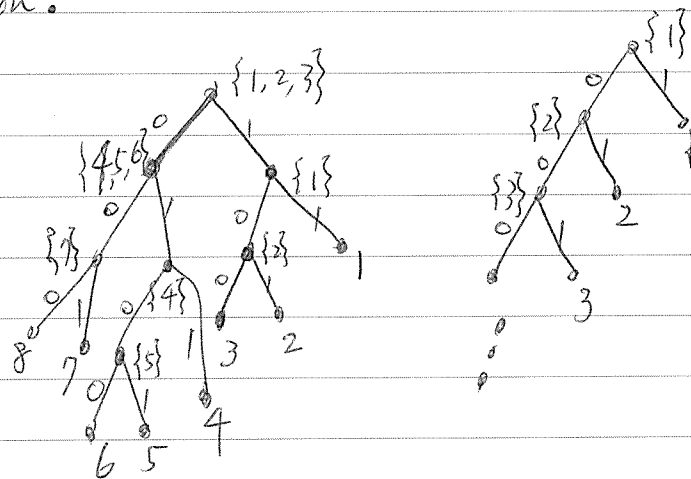


One positive item in $\{1, 2, \dots, 8\}$.



Two positive items in $\{1, 2, \dots, 8\}$.

In general, an adaptive algorithm can only be applied to find positives one by one except the test is carried out via a lucky selection.

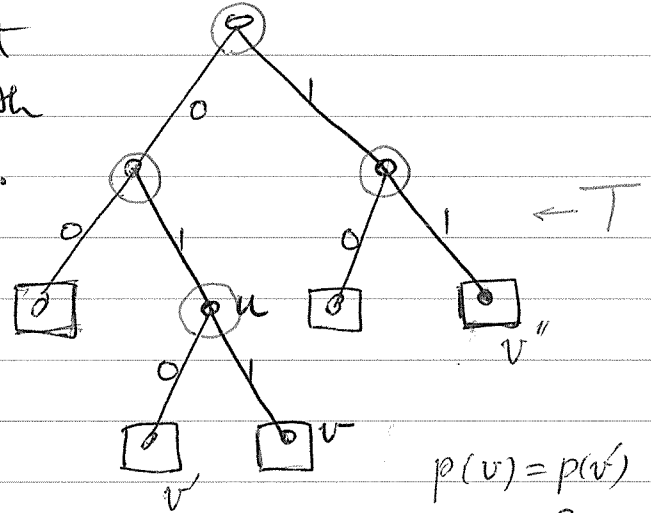


The rules of representation (Sample space S , Algorithm (T))

① Each internal node u is associated with a test $t(u)$. Two children are two outcomes of the test and the left one is "negative".

② The test history of u , $H(u)$, is the set of tests and outcomes associated with the nodes and links on the path of u .

③ Each node u is also associated with an event $S(u)$ which consists all the members of S consistent with $H(u)$. $|S(v)| \leq 1$ for each leaf v .



• $t(u)$ splits $S(u)$ into two disjoint subsets.

(*) Let $\{v_1, v_2, \dots, v_k\}$ be the set of leaves in a binary tree.

Then $\sum_{i=1}^k \frac{1}{2^{p(v_i)}} = 1$. ($p(v_i)$ is the length of the path from the root to v_i .)

Algorithm (Reasonable, admissible)

Fact 7. $M(d, n) \leq M(d, n+1)$.

Proof. Adding a good item to any group does not affect the outcome.

(Note) The equality may not hold. $M(1, 8) < M(1, 9)$.

Fact 8. Let $M(m; d, n)$ denote the min. number of tests to identify d positives among n items when a particular subset of m items is known to be contaminated. Then

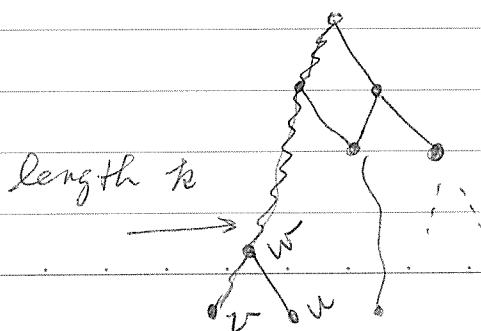
$$M(m; d, n) \geq 1 + M(d-1, n-1) \text{ for } m \geq 2 \text{ and } 0 < d < n.$$

Proof. Since $M(m; d, n) \geq M(2; d, n)$, it suffices to consider

$M(2; d, n)$. Let $\{a, b\}$ be a contaminated subset ^(group). Let T be an

algorithm such that $M(2; d, n) = k$, i.e., k is the maximum length

of a leaf in T . (Note that T will not test $\{a\}$ or $\{b\}$ as a first test! For otherwise, it takes much more tests than $M(d-1, n-1) + 1$.)



The details of proving this Fact takes some effort. The main idea is the contaminated set should take part in determining the positives if some of ^{them} do need $M_T(2; d, n)$ tests to determine them.

Intuitively, even we know a positive subset, it still needs

$M(d-1, n-1) + 1$ tests, to determine all the d positives.

(We skip the details here.)

Fact 9. $M(d, n) \leq n-1$. (By individual test and d is fixed.)

Fact 10. $M(n-1, n) = n-1$. (Determine d , s.t., $M(d, n) \geq n-1$.)

Proof. By induction on n . $n=1$ is true. (Using the hypothesis)

$$M(n-1, n) \geq M(n; n-1, n) \geq M(n-2, n-1) + 1 \quad (\text{Fact 8})$$

$$= n-2+1 = n-1. \quad \blacksquare$$

Fact 11. $M(d, n) \geq 1 + M(d-1, n-1) \geq M(d-1, n)$ for $0 < d < n$.

Proof. By induction on d , and $d=1$ is true for second inequality.

The first one follows from Fact 8 and $M(d, n) = M(n; d, n)$ if

$d \geq 1$. We prove Fact 11 in general. (For $d > 1$)

Let T be an algorithm which tests one item to start. Then.

(先测一个 Item!)

$$M(d-1, n) \leq M_T(d-1, n)$$

$$= 1 + \max\{M(d-1, n-1), M(d-2, n-1)\}$$

$$= 1 + M(d-1, n-1).$$

Fact 12. Suppose that $n > d+1$. Then $M(d, n) = n-1 \Rightarrow M(d, n-1) =$
 (Individual tests)
 $n-2$. (Note that d may be large.)

Proof. Assume that $M(d, n-1) \neq n-2$, then $\underline{M(d, n-1) < n-2}$. Let

T be the algorithm which tests a single item to start. This

implies that $M(d, n) \leq M_T(d, n) = 1 + \max\{M(d, n-1), M(d-1, n-1)\}$

By Fact 11, $M(d, n) \leq 1 + M(d, n-1) < 1 + (n-2)$ (assumption)

$$= M(d, n). \rightarrow \leftarrow$$

Fact 13. $M(d, n) = M(d-1, n) \Rightarrow M(d, n) = n-1$.

Proof. By induction on $n-d$ and it is true when $n-d=1$.

(Fact 10)
 without using assumption

By Fact 11, $M(d, n) = M(d-1, n) \Rightarrow M(d, n) = M(d-1, n-1) + 1$.

Let T be a minimax algorithm for the (d, n) problem. First,
 in T , a test of a group of m items is carried out.

Case 1. $m > 1$.

a contradiction

$$M_T(d, n) \geq 1 + M(m; d, n) \geq 2 + M(d-1, n-1), \text{ since}$$

$$M(d, n) = M(d-1, n-1) + 1.$$

Case 2. $m = 1$

$$\begin{aligned} M(d, n) &\geq M_T(d, n) = 1 + \max \{ M(d, n-1), M(d-1, n-1) \} \\ &= 1 + M(d, n-1). \end{aligned}$$

Now, since $d < n-1$ and $M(d, n-1) = M(d-1, n-1)$, we have

$M(d-1, n-1) = n-2$ by induction. Therefore,

$$M(d, n) = 1 + M(d-1, n-1) = n-1. \quad \blacksquare$$

Fact 14 If $M(d, n) < n-1$, then $M(d, n) \geq 2l + M(d-l, n-l)$ for $0 < l \leq d < n$.

Proof: By Fact 10, $M(n-1, n) = n-1$, we have $d < n-1$, i.e. $n-d > 1$.

The proof follows by showing $M(d, n) \geq 2 + M(d-1, n-1)$. (?)

($M(d-1, n-1) \geq 2 + M(d-2, n-2)$ inductively)

Let T be a minimax algorithm for the (d, n) problem which starts with a test on m items. First, if $m > 1$, then by

Fact 8, we have $M_T(d, n) \geq 1 + 1 + M(d-1, n-1)$. So, let $m=1$ and

suppose that $M_T(d, n) < 2 + M(d-1, n-1)$. This implies that

$$1 + M(d-1, n-1) \geq M_T(d, n) = 1 + \max \{M(d, n-1), M(d-1, n-1)\} \\ = 1 + M(d, n-1).$$

By the fact $M(d, n-1) \geq M(d-1, n-1)$, we have $M(d-1, n-1) = M(d, n-1)$.

Hence $M(d-1, n-1) = n-2$, (Fact 13) and thus $M(d, n) = n-1$. $\rightarrow \leftarrow$

Fact 15 $M(d, n) \geq \min \{n-1, 2l + \lceil \log \left(\frac{n-l}{d-l} \right) \rceil\}$ for $0 \leq l \leq d < n$. $\square$

Proof. $M(d-l, n-l) \geq \lceil \log \left(\frac{n-l}{d-l} \right) \rceil$. $\square$

Fact 16 $M(d, n) + 1 \geq M(\bar{d}, n) \geq M(d, n+1)$.

Proof.

Claim. For each algorithm T of the (d, n) problem, there exists an algorithm T' for the $(\bar{d}, n)$ problem such that $M_T(d, n) + 1 \geq M_{T'}(\bar{d}, n)$.

If the above claim is true, then $M(d, n) + 1 \geq M(\bar{d}, n)$. As

to the second inequality, it follows by ignoring one item. $\square$