Combinatorial Group Testing

Michael T. Goodrich
University of California, Irvine

Group Testing

- *Input:* ***n*** items, numbered 0,1, …, n-1, at most ***d*** of which are **defective**.
- *Output:* the indices of the defective items.
- Items can be grouped into subsets, each of which can be tested to see it contains a defective item or not.
- *Goal:* minimize the total number of tests
- *Original problem:* Testing blood samples.

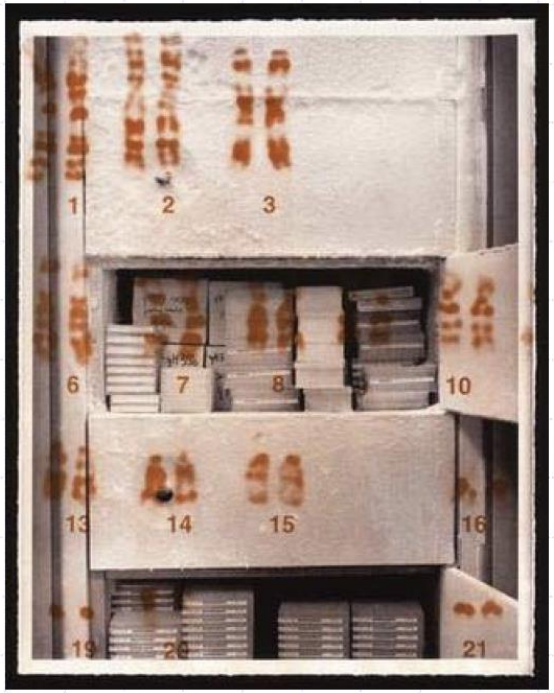
FIGURE 26.—U.S. Army cadets from Marquette University ready to give blood at the Milwaukee, Wis., donor center.

Testing Schemes

- **Non-adaptive**: All tests must be done in parallel.
- **Adaptive**: Tests can be done sequentially.
  - Adaptive is easier, but the non-adaptive approach has wider applications.
  - In some applications, high probability of success is sufficient.
  - Others require 0% errors.
- **Semi-adaptive**: Do tests in a sequence of rounds, where the tests in each round are done in parallel.

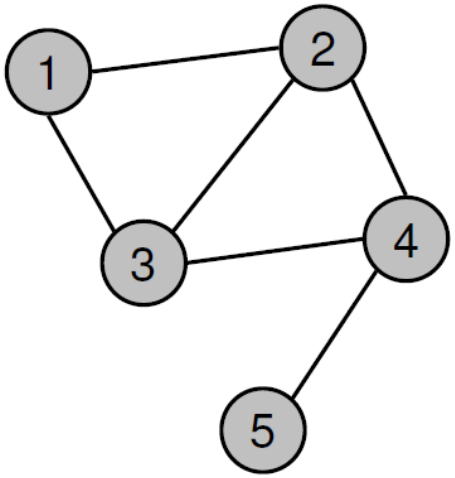
Applications

- Screening vaccines for contamination
- Filtering clone libraries of DNA sequences (identifying which ones contain a certain DNA sequence)
- Computer security – for data forensics
- Computer fault diagnosis



Application: Learning Attribute Vectors

- Viewed abstractly, we are trying to learn a database X of g **attribute vectors**:



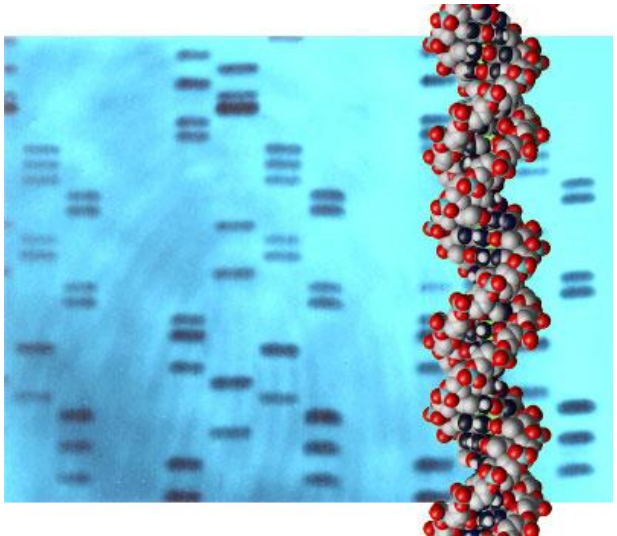
	1	2	3	4	5
1	0	1	1	0	0
2	1	0	1	1	0
3	1	1	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0

Facebook Application

- Each member has a “vector” of friendships
- For any member M, the system returns a bit for whether M has a friend in common with the attacker, even if M restricts this information to friends-of-friends
- We can use non-adaptive scheme to learn friendship relationships in any sub-community in Facebook

DNA Application

- DNA sequences are stored in a database, D.
- For any sequence Q, the database returns a score for how close Q is to each sequence in D
- We form a binary vector w.r.t. places where mutations happen relative to a reference string R
- We can use non-adaptive scheme to learn DNA strings in D.



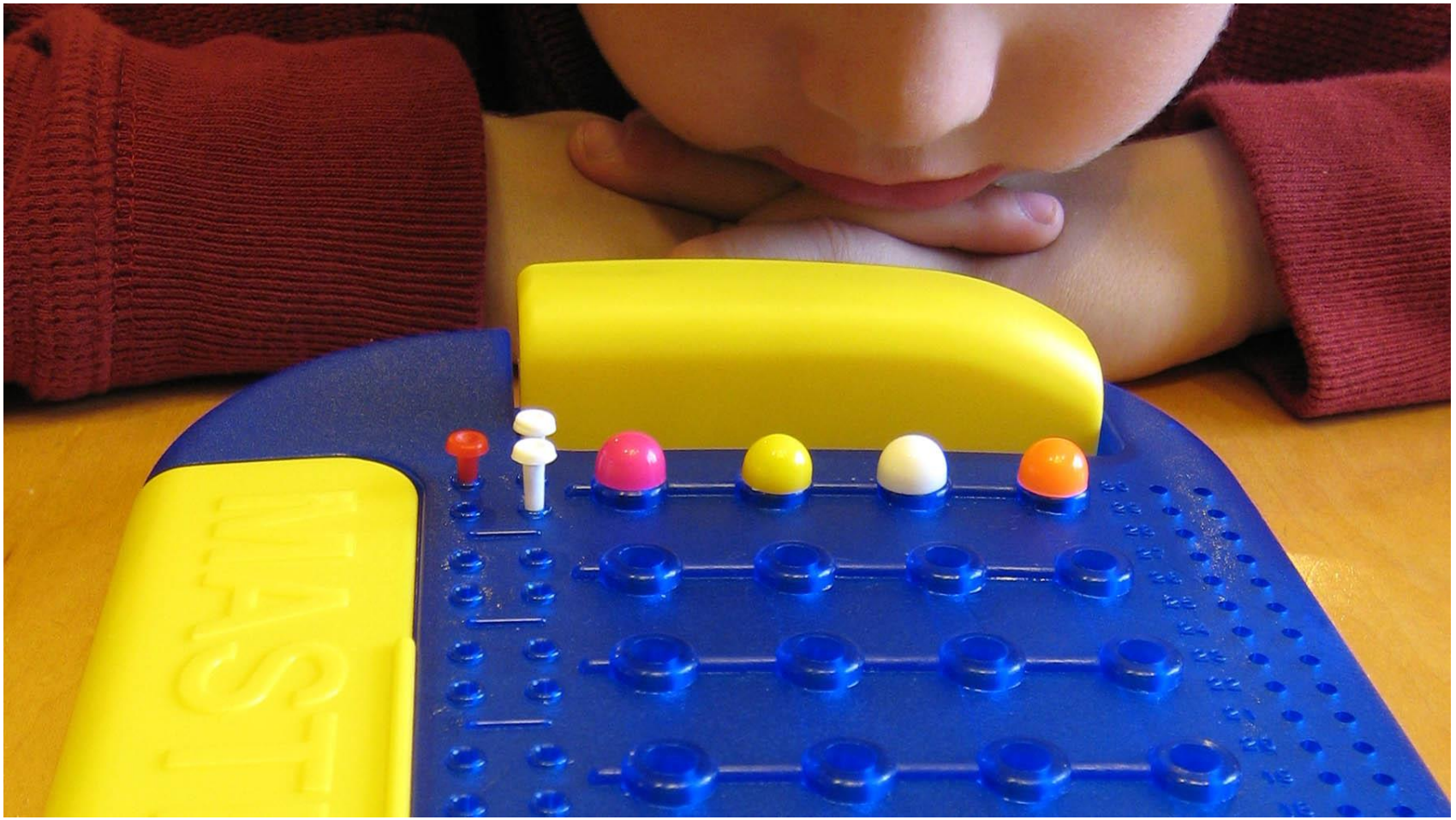
Netflix Application

- Movie ratings vectors are stored in a database, D.
- For any vector V, the database returns a score for how close V is to each vector in the database
- We can form a binary attribute vector for movies
- We can use non-adaptive scheme to learn ratings vectors in D.

Mastermind

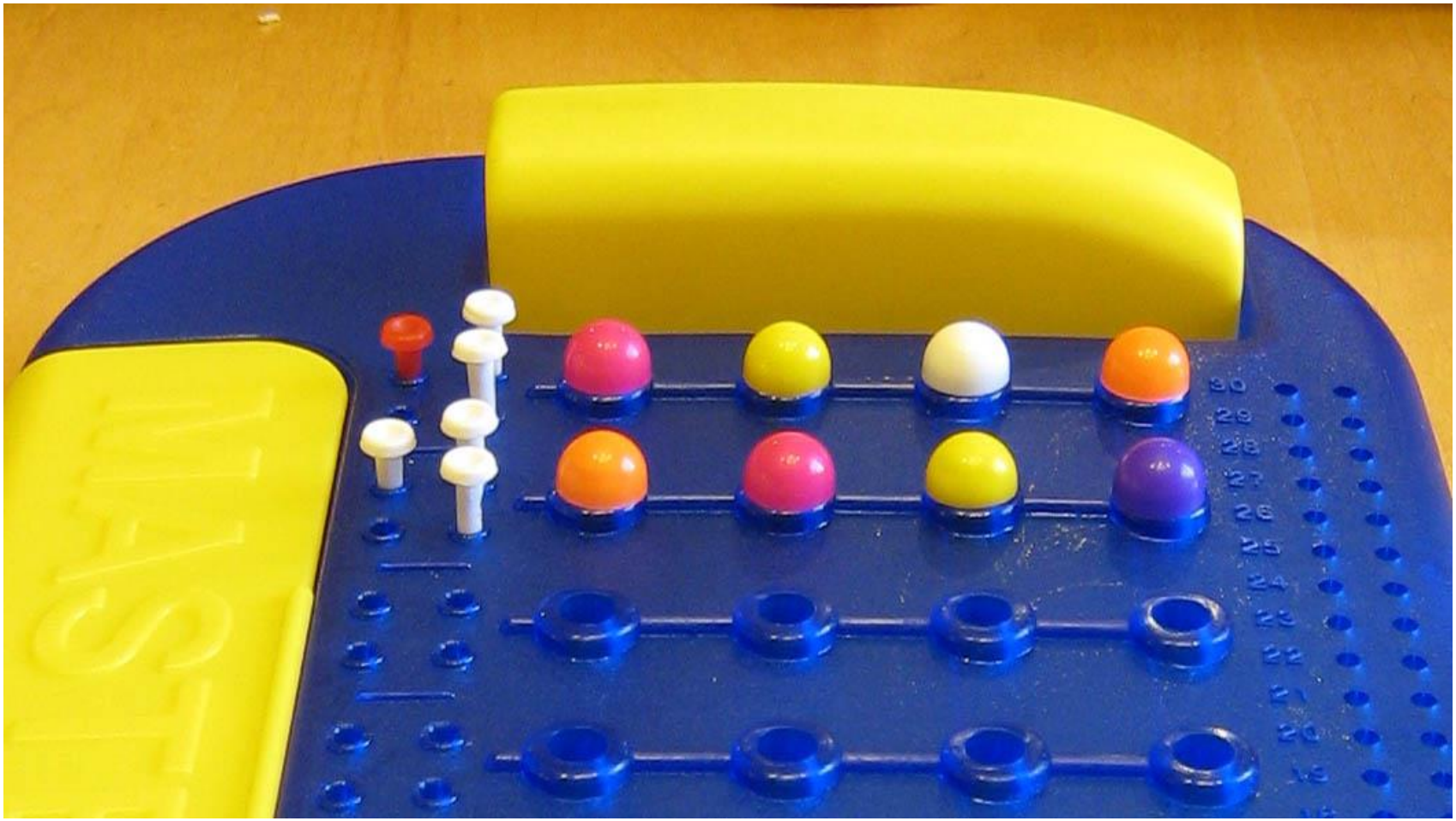
- Mastermind is a two-player game
  - A **codemaker**: creates a vector X of length N using an alphabet of size K (called "colors")
  - A **codebreaker**: guesses vectors Q_1, Q_2, …, of length N using same alphabet
- Codemaker scores each guess:
  - **Black score**: number of characters in right place
  - **White score**: number of characters not in right place but with right color

Mastermind example



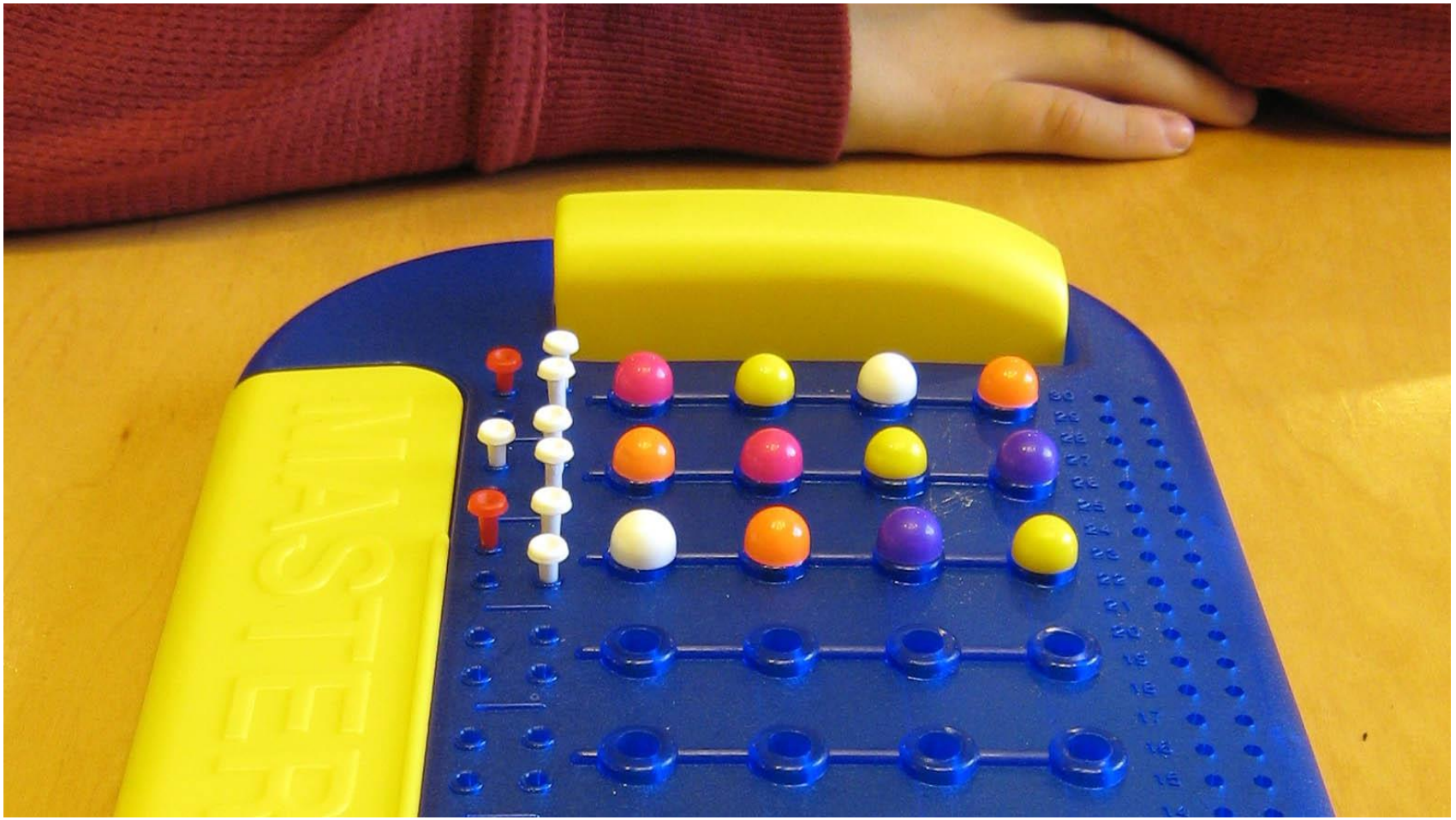
Turn number 1

Mastermind example



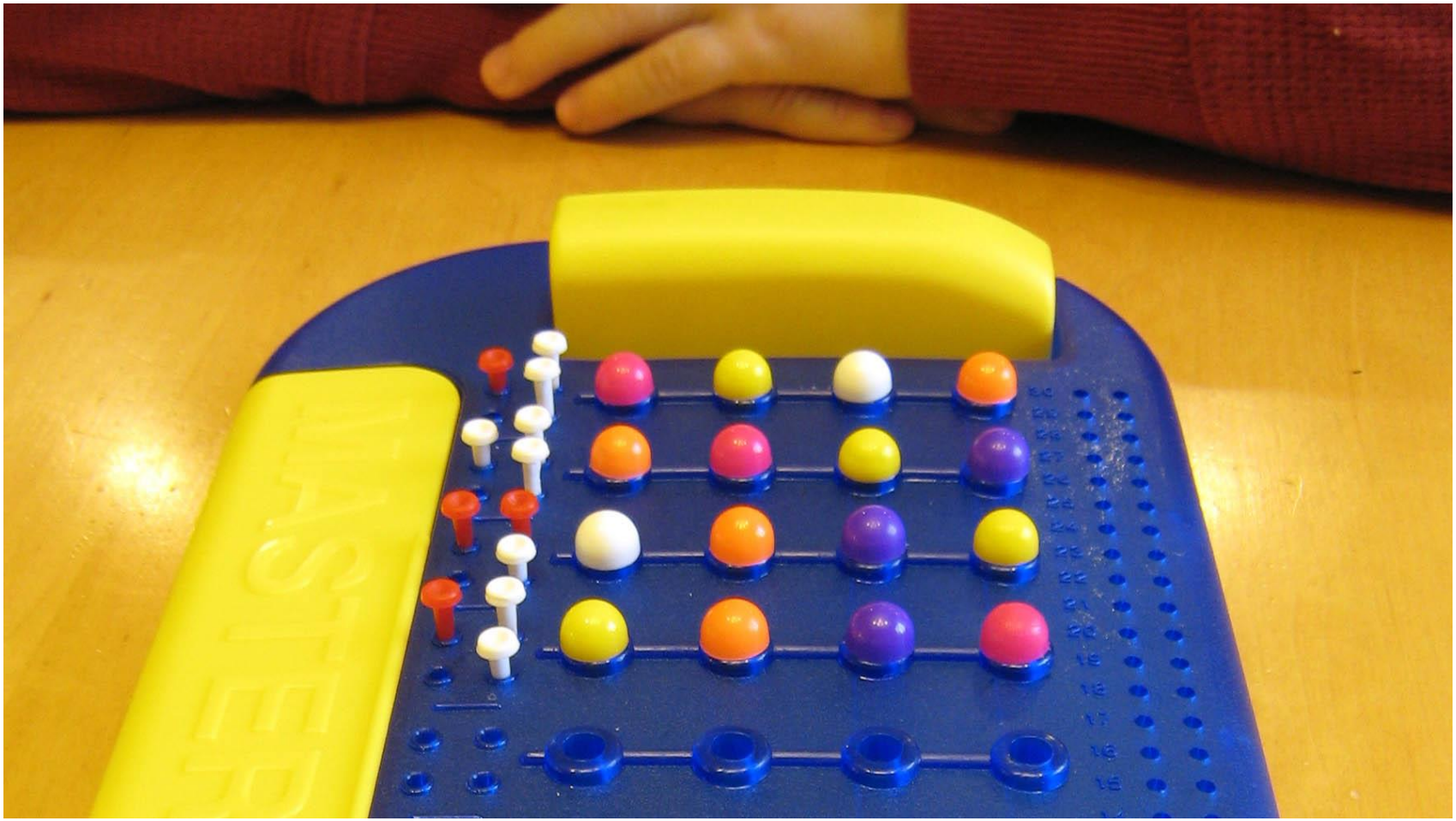
Turn number 2

Mastermind example



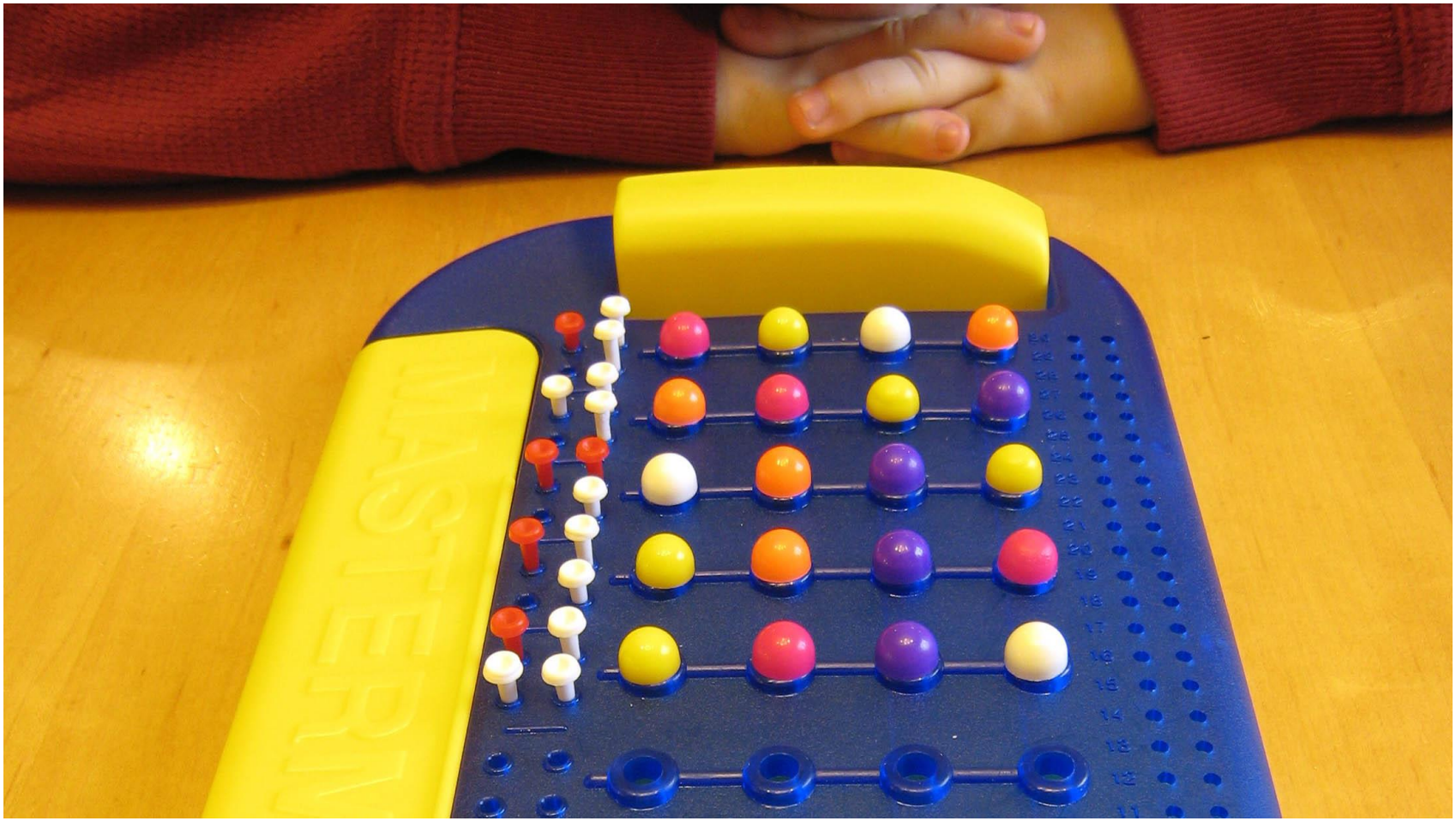
Turn number 3

Mastermind example



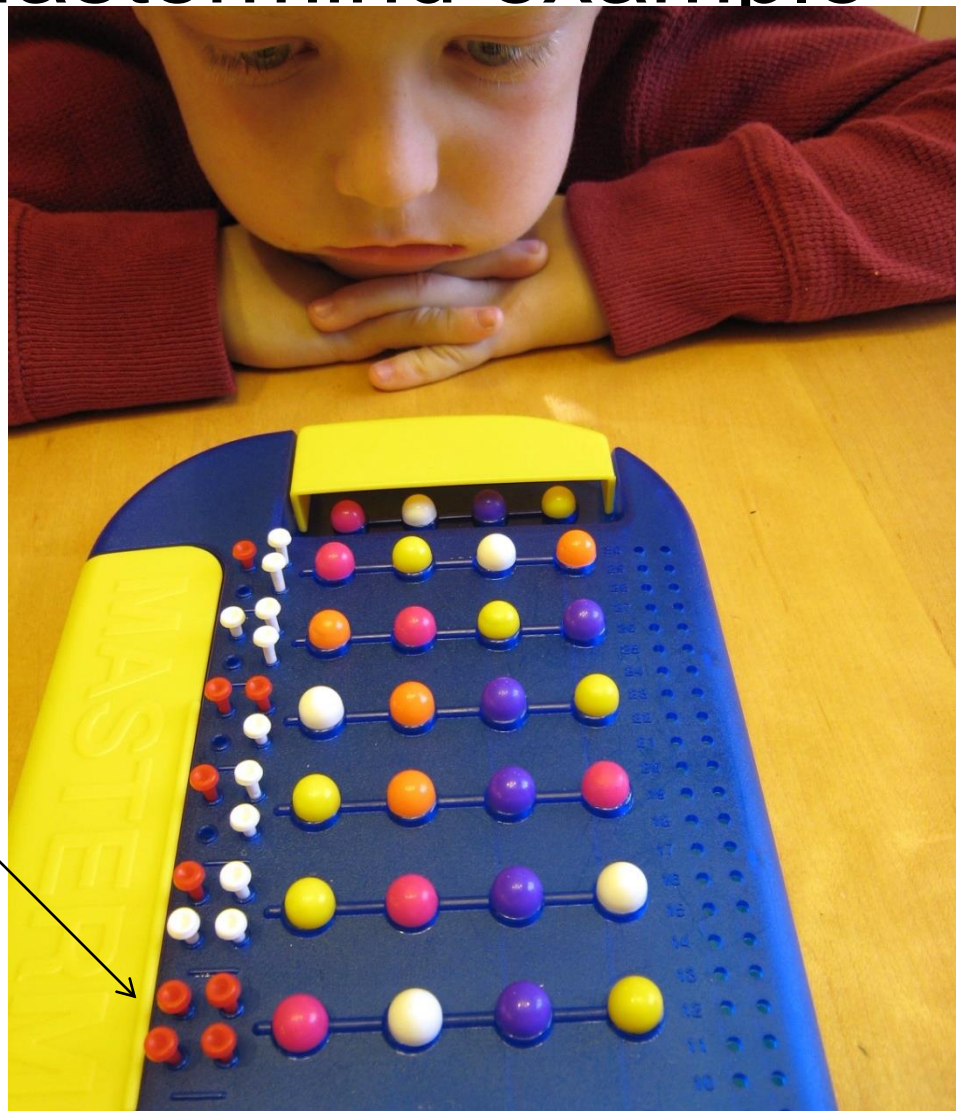
Turn number 4

Mastermind example



Turn number 5

Mastermind example

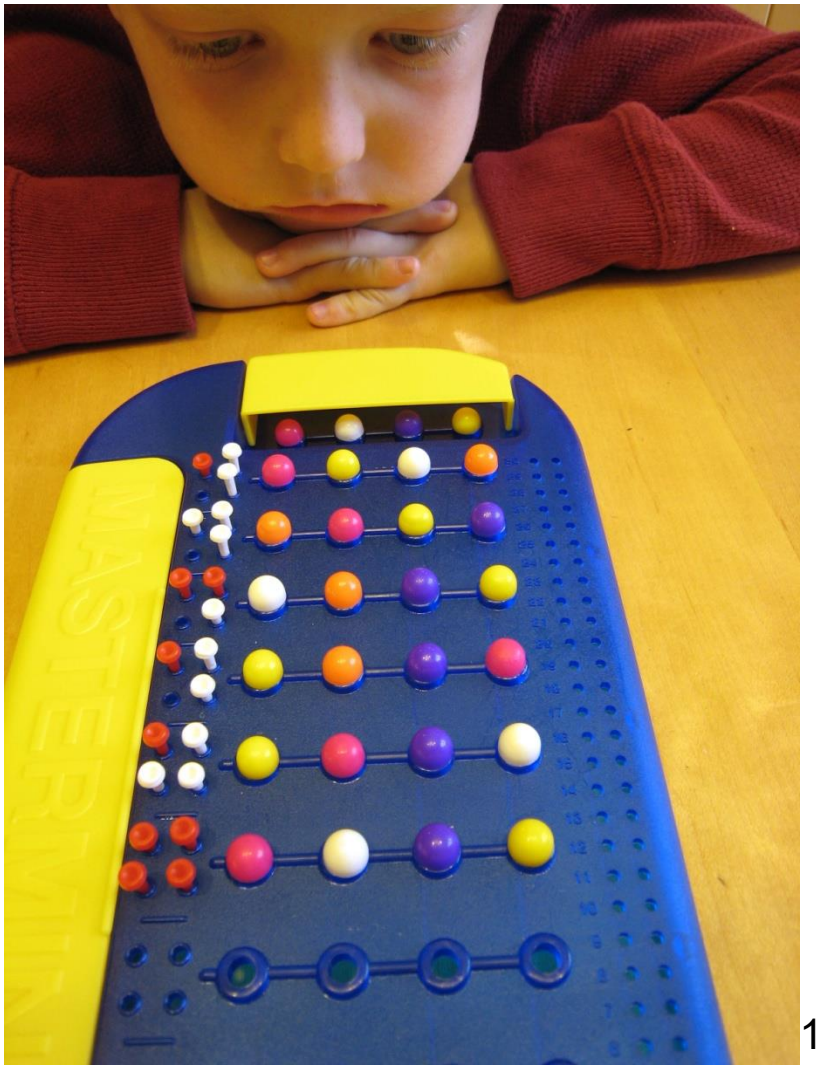


Restricted Mastermind

- **Goal**: Try to guess the secret vector as quickly as possible, in terms of
  - N = length of vector
  - K = number of colors

– **Restriction**: use only:

  - Black score: number of characters in right place
  - Non-adaptive guesses



Efficiency Measures

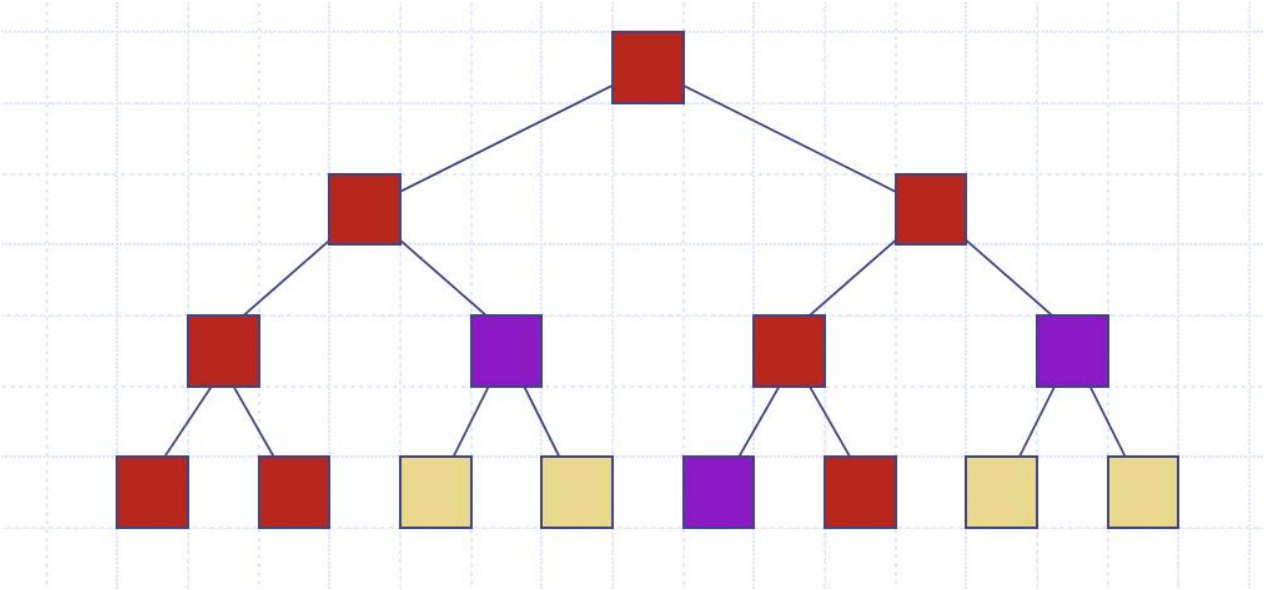
- t(n,d) = number of tests to identify up to d defectives among n items.
  - t(n,d) must be $\Omega(\min\{n, d \log (n/d)\})$.
- A(n,t) = analysis time needed to determine which items are defective (after the tests are done).
  - time-optimal if A(n,t) is O(t).

Simple Adaptive Algorithm

- For the fully-adaptive case:

1. Place a complete binary tree “on top of” the items
2. Do a top-down search to defectives

- This will use t(n,d) = O(d log (n/d)) tests.

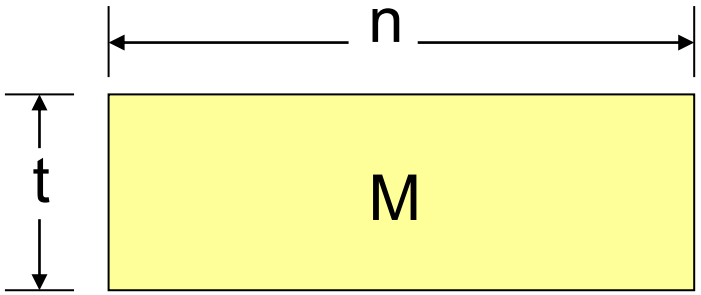


A Simple Nonadaptive Case

- For non-adaptive case when d=1:
  - Consider item numbers in binary
  - Test i is set of items w/ bit i = 1
  - Positive (defective) and negative (non-defective) tests identify the binary index of the defective item
  - t(n,d) is O(log n)
  - d=2 and d=3 cases are much harder…

Matrix View of A Single Round of Testing

- A single round of a testing regimen can be viewed as a t x n binary matrix M:
  - M[i,j] = 1 if and only if test i includes item j



Every row is a test

Every column is an item

Rake-and-Winnow Algorithm

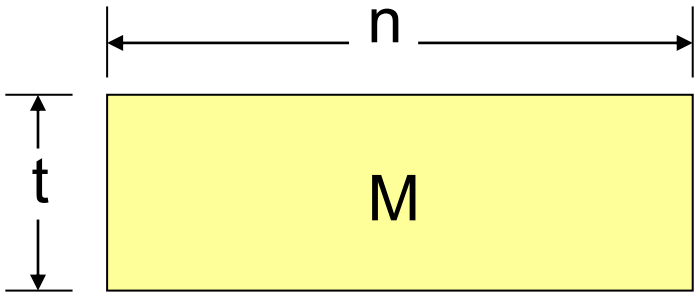
- Uses a randomized approach motivated by Bloom filtering.
- Also uses a matrix M, but in 2 rounds
- Given a set D of d columns in M and a column j, say j is **distinguishable** from D if there is a row i such that M[i,j]=1 but M[i,j']=0 for each j' in D.
- M is **(d,k)-resolvable** if, for any d-sized subset D, there are fewer than k columns that are not distinguishable from D.

Two-Stage Algorithm

- Use a (d,k)-resolvable matrix M in the first round and make a test for each row.
- Discard all the items in negative (non-defective) tests.
- There are at most d+k remaining items.
- Test each remaining item individually.

Constructing the Matrix

- Given t (set in the analysis), let M be a 2t x n matrix defined randomly:
  - For each column j, choose t/d rows of M at random and set these entries to 1.
  - that is, we "inject" j into those t/d tests



Every row is a test

Every column is an item

Alternative Approach

- Alternatively, we could set M[i,j] = 1 independently with probability p=1/d.

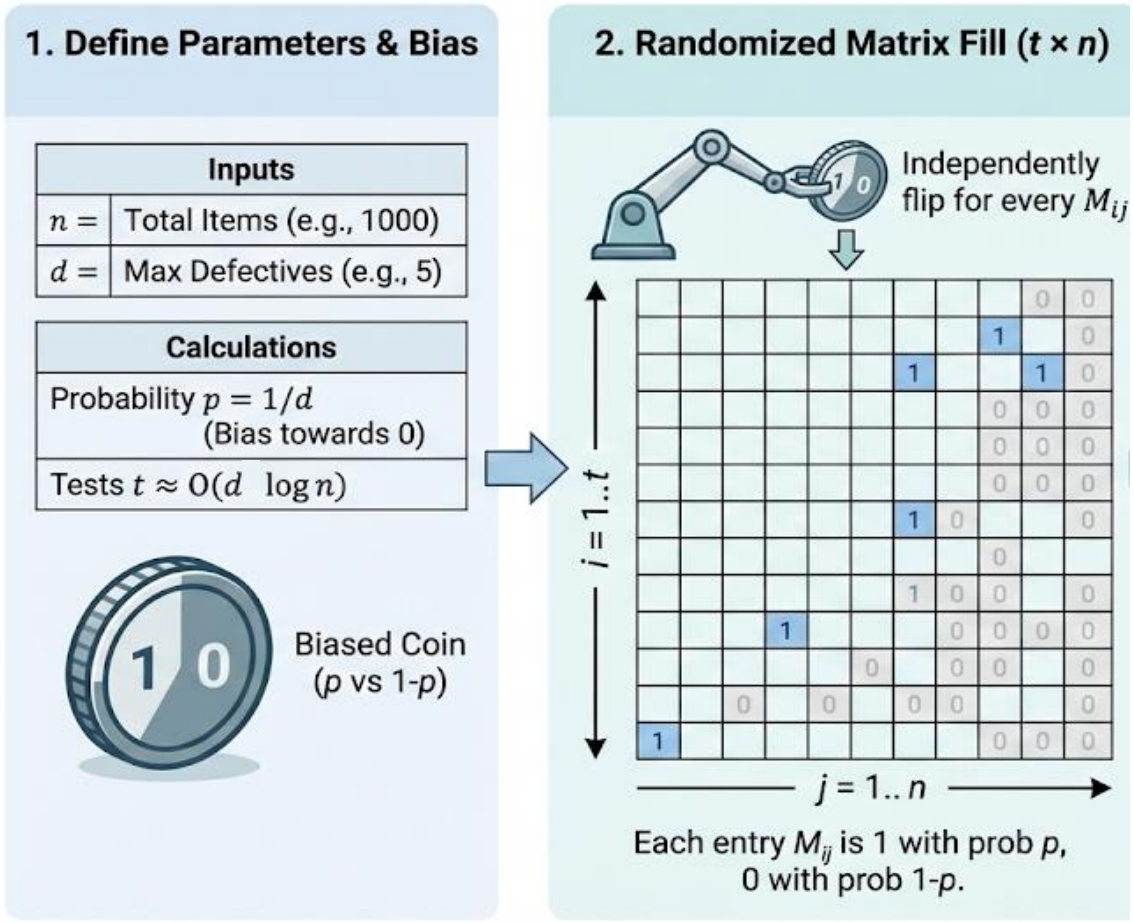


Image generated using Nano Banana Pro.

Analysis

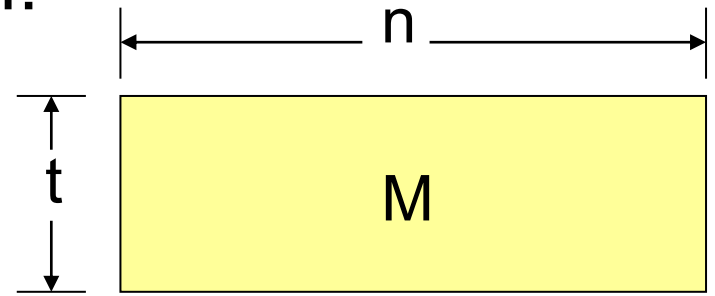
- Let D be a fixed set of d defectives samples. For each (column) item i in C−D, let X_i denote the indicator random variable that is 1 if i is falsely identified as a positive sample by M (that is, i is not included in the set of (negative) items distinguished from those in D), and is 0 otherwise.
- X_i's are independent and any X_i is 1 (a false positive) with probability at most $2^{-t/d}$.
- Therefore, the expected value of the sum, X, E[X], is at most $\mu = n/2^{t/d}$. By a Chernoff bound,

$$\Pr(X \geq k) \leq \left(\frac{e\hat{\mu}}{k}\right)^k = \frac{\left(\frac{en}{k}\right)^k}{2^{(t/d)k}}.$$

- Thus, whp, M is (d,d)-resolvable and we can find all the defectives with 2d tests in a second round.

Recall the Matrix View of A Single Round of Testing

- A non-adaptive testing regimen can be viewed as a t x n binary matrix M:
  - M[i,j] = 1 if and only if test i includes item j



- M is **d-disjunct** if the Boolean sum of any d columns does not contain any other column.
  - An item is defective iff all its tests are positive

High-Probability Nonadaptive

- With high probability, there exists a **nonadaptive group test** using $O(d^2 \log n)$ tests.

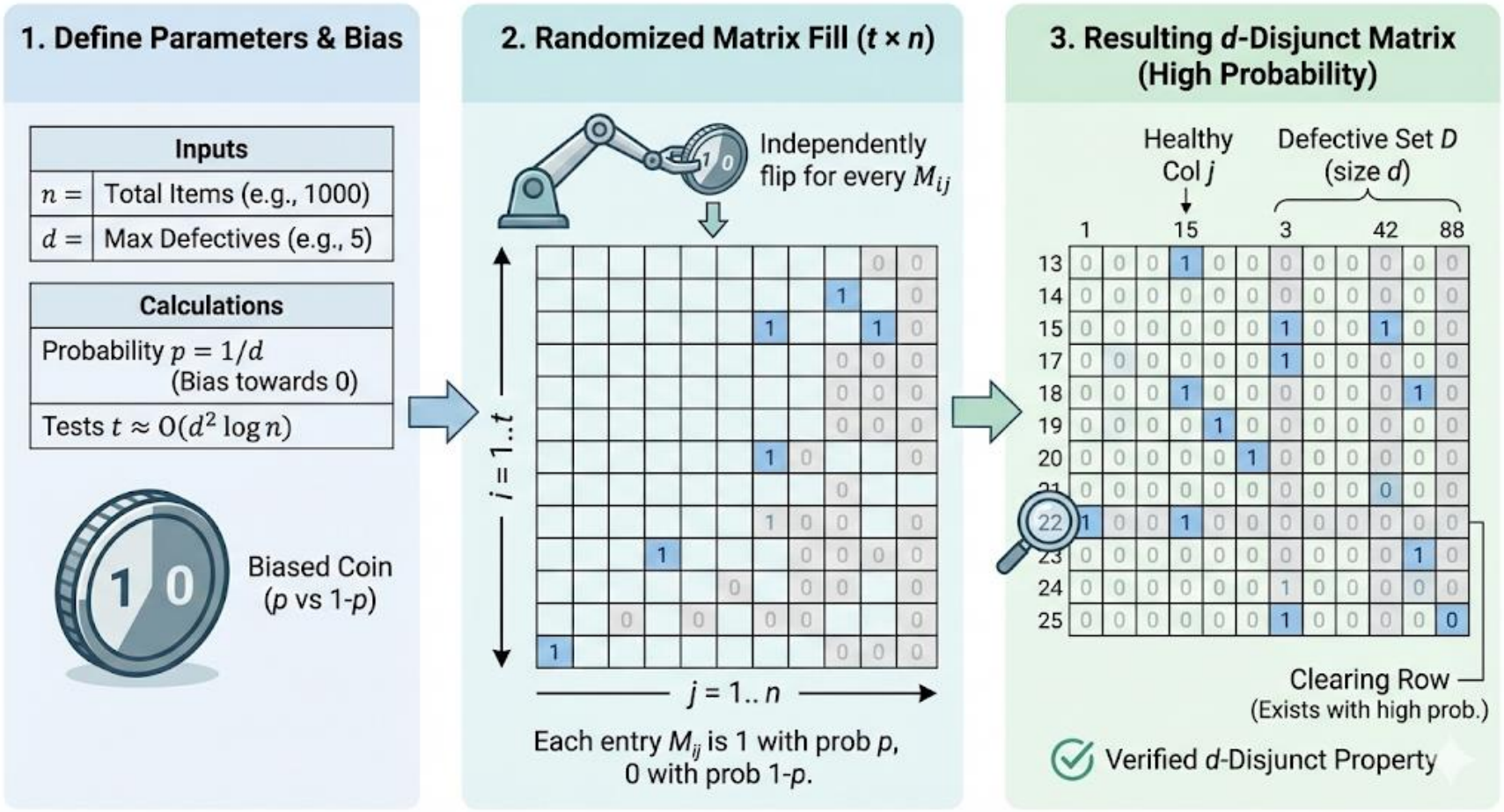


Image generated using Nano Banana Pro.

Constructing the Matrix

To construct a $t \times n$ matrix M that identifies up to d defectives:

1. **Set parameters:**
   - n: Number of items.
   - d: Max number of defectives.
   - p: Probability of including an item in a test. Set $p = \frac{1}{d}$.
   - t: Number of tests. Set $t \approx O(d^2 \log n)$.
2. **Fill the matrix:**
   - For every entry M_{ij} (where $1 \leq i \leq t$ and $1 \leq j \leq n$), set $M_{ij} = 1$ with probability p and 0 with probability $1 - p$.
   - Do this independently for all entries.

Analysis for Nonadaptive Algorithm

1. The Goal (The Condition) Recall the definition of a d-disjunct matrix: For any "healthy" item j and any set of d "defective" items D, there must be **at least one row** where item j is included (1) and *none* of the items in D are included (all 0). This specific row "clears" item j.

2. Analyzing a Single Row Let's look at just one row i.

- Probability that item j is included: p
- Probability that all items in D are excluded: $(1-p)^d$
- Probability that this row "clears" item j:

$$P_{\text{success}} = p(1-p)^d$$

Analysis, Part 2

3. Optimizing p To minimize the number of tests needed, we want to maximize this success probability. Using calculus, the maximum occurs when $p \approx \frac{1}{d+1} \approx \frac{1}{d}$. Substituting $p = 1/d$:

$$P_{\text{success}} \approx \frac{1}{d}\left(1 - \frac{1}{d}\right)^d \approx \frac{1}{d} \cdot \frac{1}{e} = \frac{1}{ed}$$

So, a single row has roughly a $\frac{1}{ed}$ chance of doing the job.

4. Analyzing t Rows The probability that *all* t rows **fail** to clear item j against set D is:

$$P_{\text{fail}} = (1 - P_{\text{success}})^t \approx \left(1 - \frac{1}{ed}\right)^t \approx e^{-\frac{t}{ed}}$$

Analysis, Part 3

5. The Union Bound (Accounting for all combinations) We need the matrix to work for **every** possible combination of 1 healthy item and d defective items. The number of such combinations is roughly $n \cdot \binom{n}{d} \approx n^{d+1}$.

We apply the Union Bound:

$$(\text{Total Combinations}) \times P_{\text{fail}} < 1/n$$

$$n^{d+1} \cdot e^{-\frac{t}{ed}} < 1/n$$

Taking logarithms and solving for t, we get the fundamental bound for randomized group testing:

$$t = O(d^2 \log n)$$