Learning Hidden Graphs

Hung-Lin Fu

Department of Applied Mahematics,
Naional Chiao Tung Universiy,
Hsin Chu, Taiwan

What is “Group Testing”?

- A problem:
  I have a favor number in my mind and the number is in $\{1, 2, 3, \cdots, 63\}$. Can you find the number by asking (me) as few ***queries*** as possible? (A *query* asks if the number is in a certain subset of the given set.)

The answer : At most **6**

- (1) Is the number larger than 32?

  Yes or Not?
- (2) If the answer is “Yes”, then ask if the number is larger than 48 or not. On the other hand, if the answer is “Not”, then ask if the number is larger than 16 or not.
- Continued …
- Worst case: 6 (Information bound)

Another Idea

32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

2	3	6	7	10	11	14	15
18	19	22	23	26	27	30	31
34	35	38	39	42	43	46	47
50	51	54	55	58	59	62	63

4	5	6	7	12	13	14	15
20	21	22	23	28	29	30	31
36	37	38	39	44	45	46	47
52	53	54	55	60	61	62	63

8	9	10	11	12	13	14	15
24	25	26	27	28	29	30	31
40	41	42	43	44	45	46	47
56	57	58	59	60	61	62	63

1	3	5	7	9	11	13	15
17	19	21	23	25	27	29	31
33	35	37	39	41	43	45	47
49	51	53	55	57	59	61	63

16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

Group testing (General Model)

- Consider a set N of n items consisting of at most d *positive* (used to be called defective) items with the others being *negative* (used to be called good) items.
- A group test, sometimes called a *pool*, can be applied to an arbitrary set S of items with two possible outcomes; *negative:* all items in S are negative; *positive:* at least one positive item in S, not knowing which one or how many.

Adaptive (Sequential) and Non-adaptive

- Can you see the difference between the above two ways in finding the answer?
- The first one (*adaptive or sequential*): You ask the second question (query) after knowing the answer of the first one and continue …. That is, the previous knowledge will be used later.
- The second one (*non-adaptive*): You can ask all the questions (queries) *at the same time.*

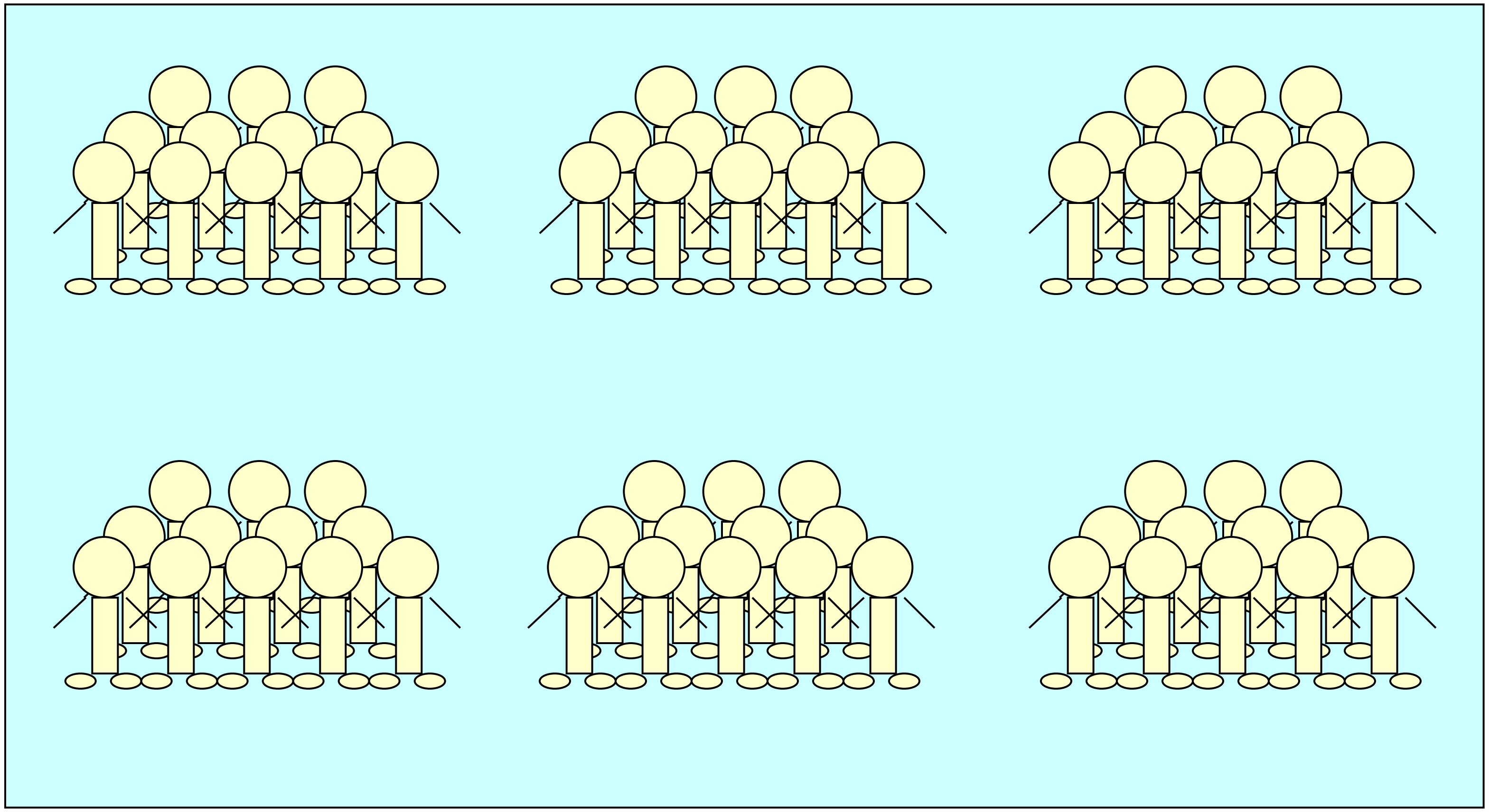
Algorithms

- Adaptive algorithm
- Non-adaptive algorithm
- k-stage algorithm

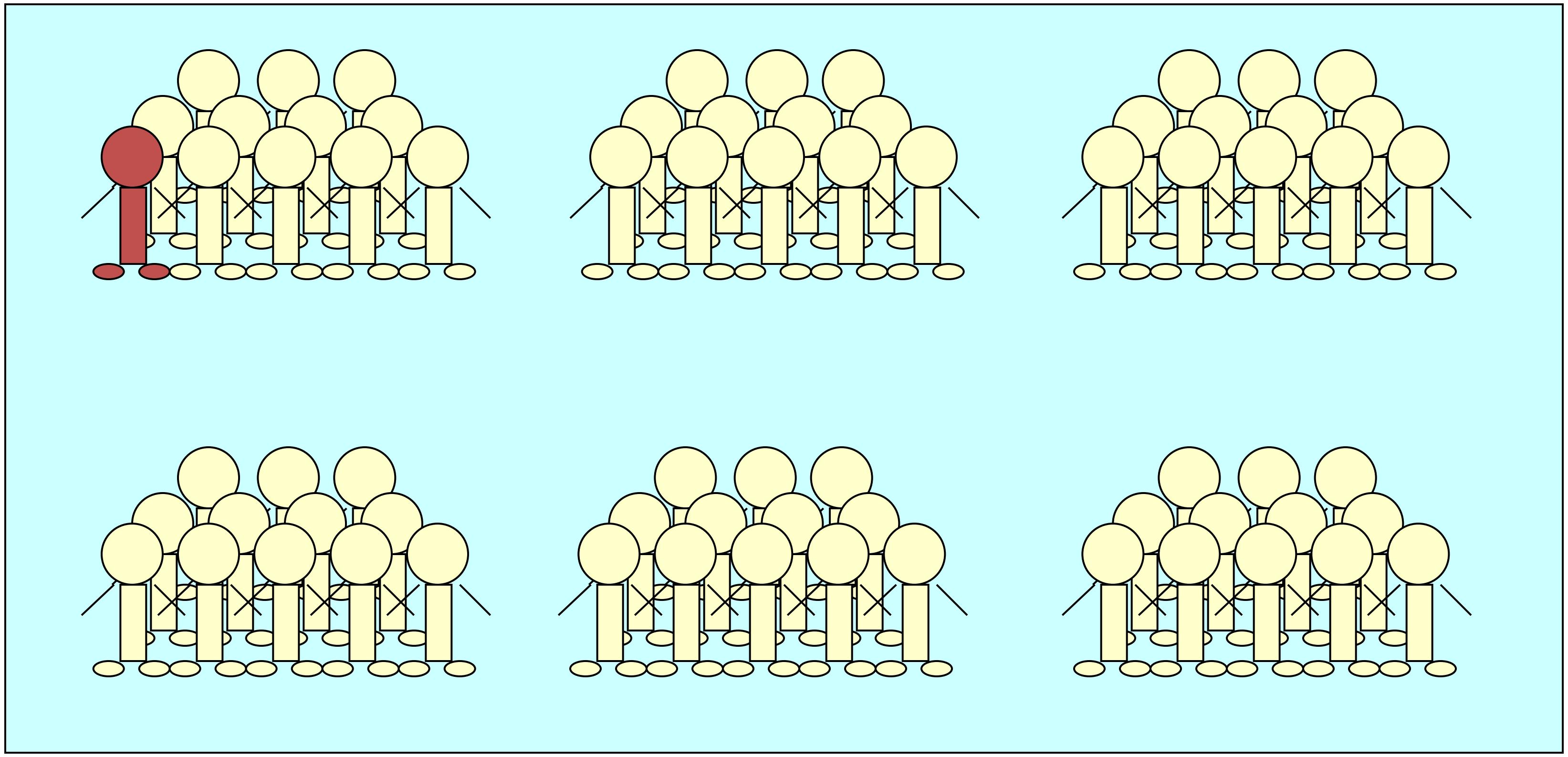
The most popular one is a 2-stage algorithm in which we use one of the above two basic types of algorithm first (normally the second one) and then in the second stage we test the left “suspected” items one by one.

Blood Testing

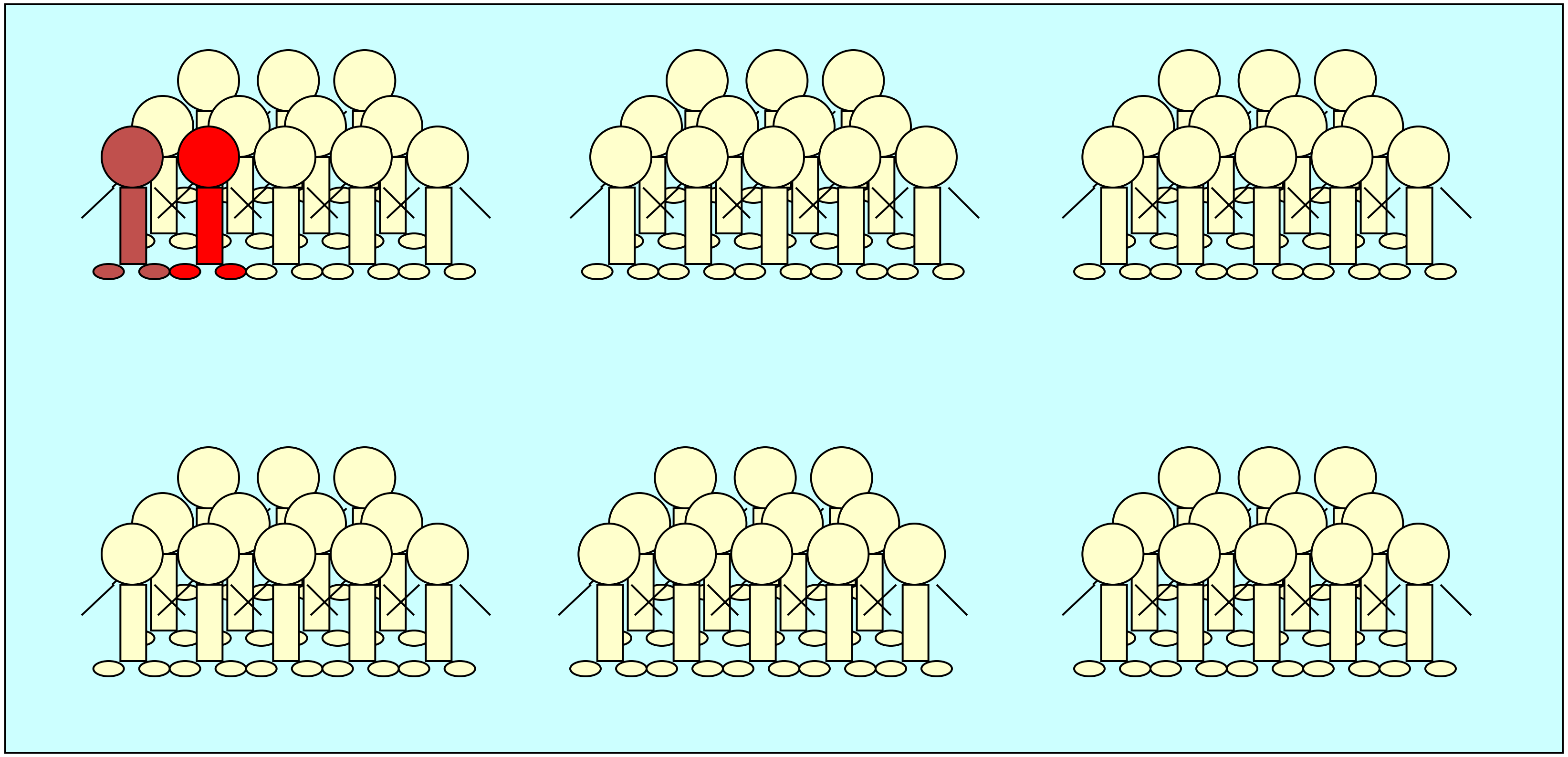
Syphilis Tests (World war II)



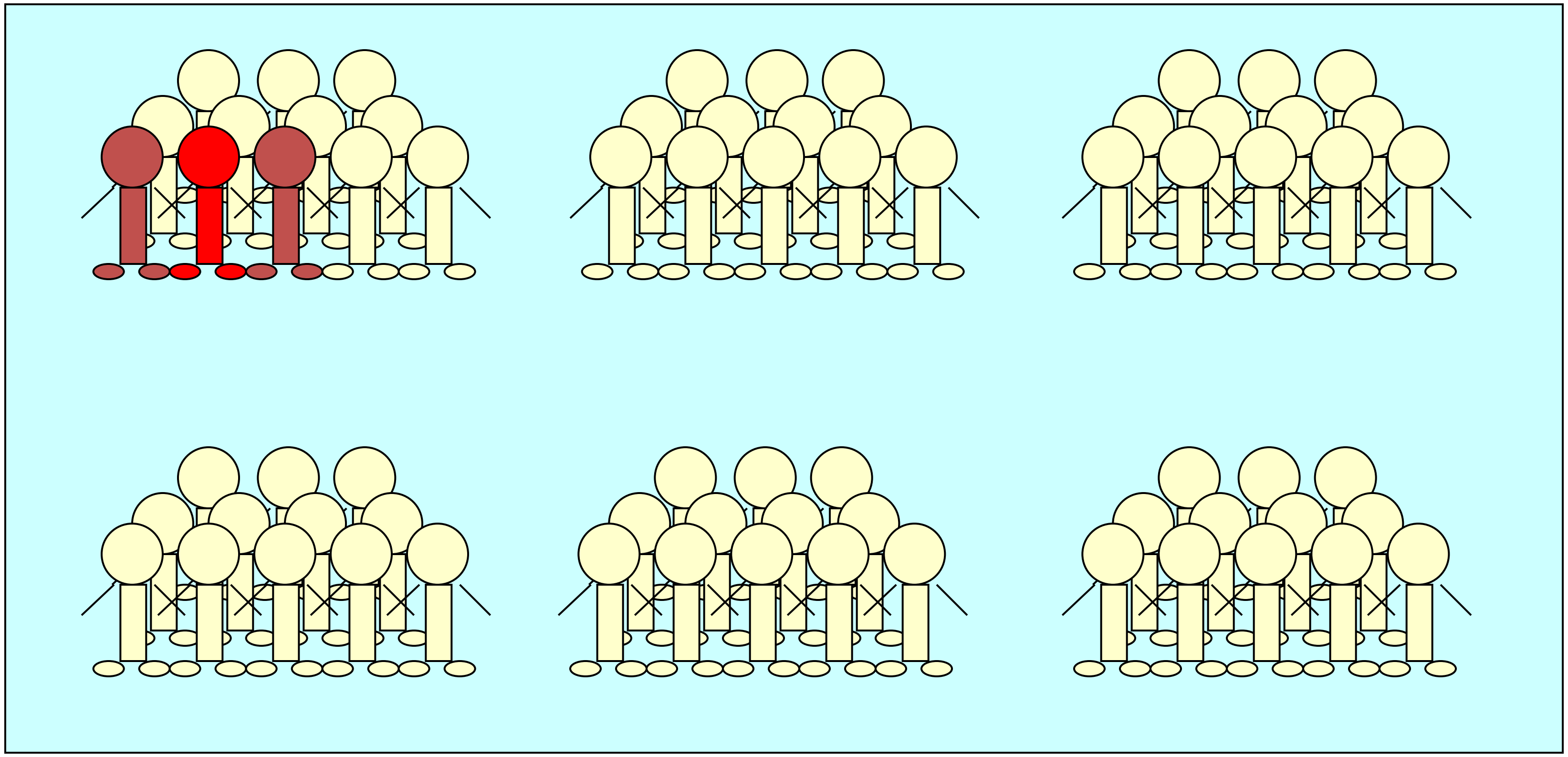
One by one



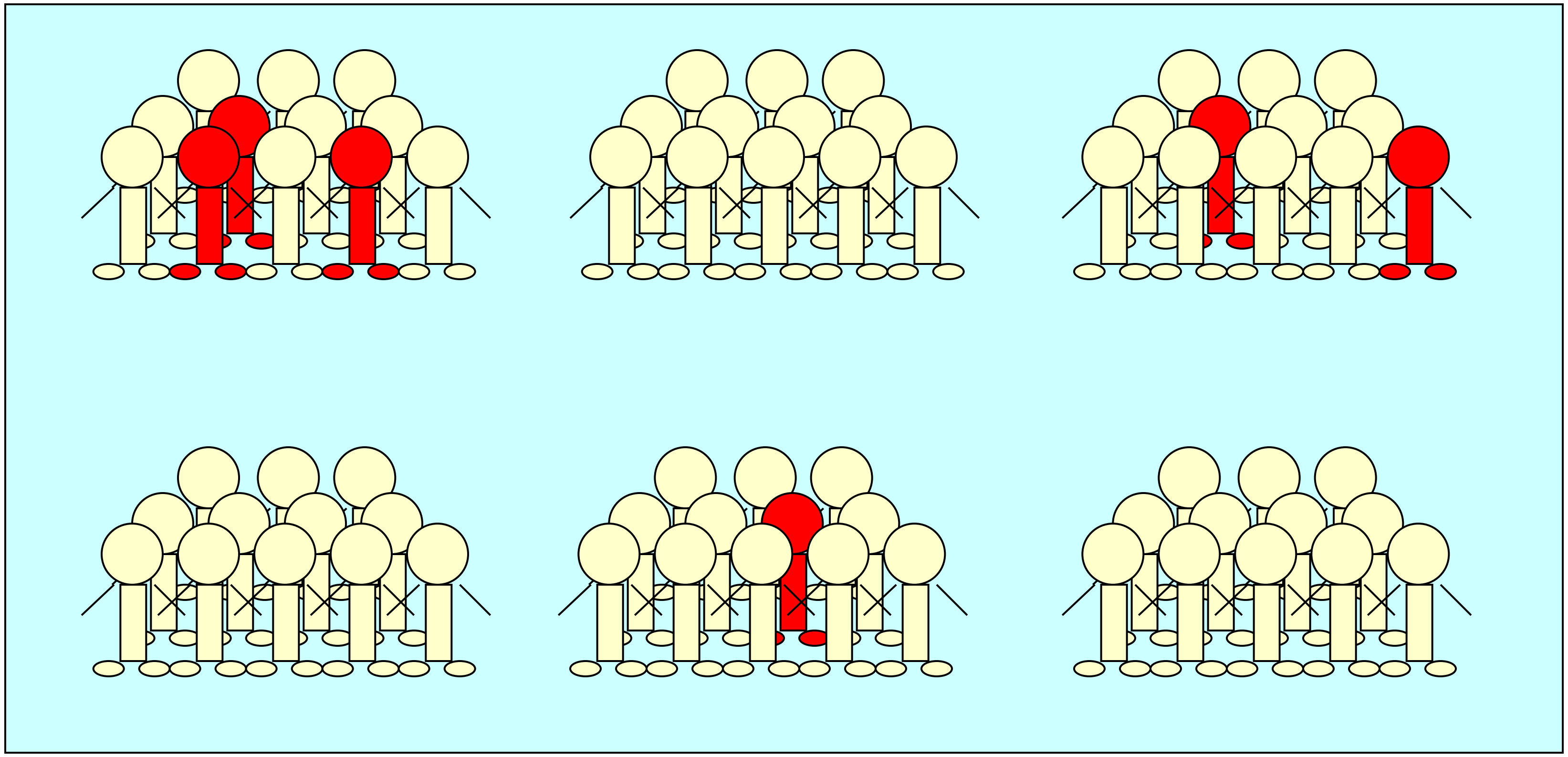
Red *is positive and* Brown *is negative*



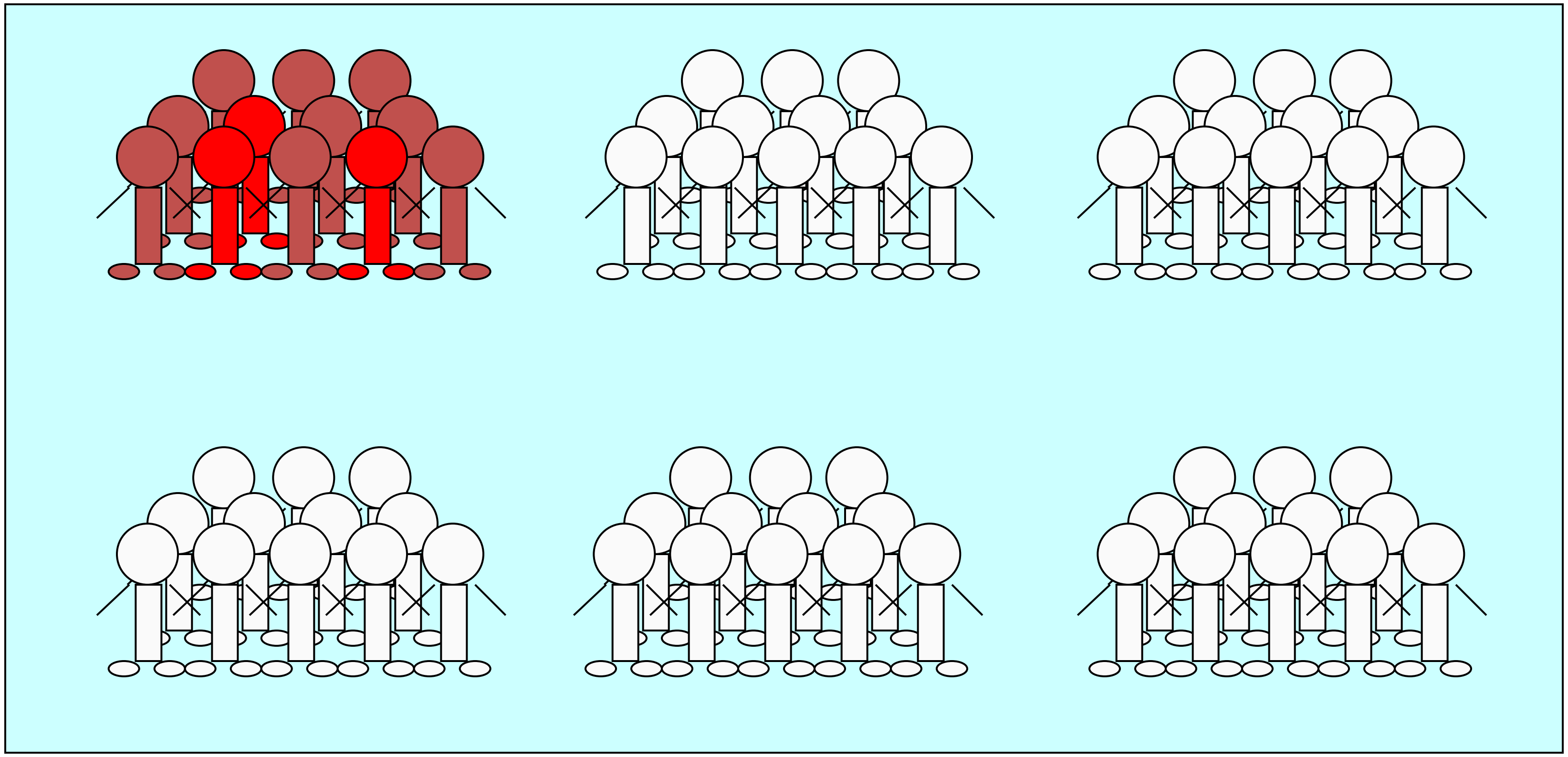
Keep going



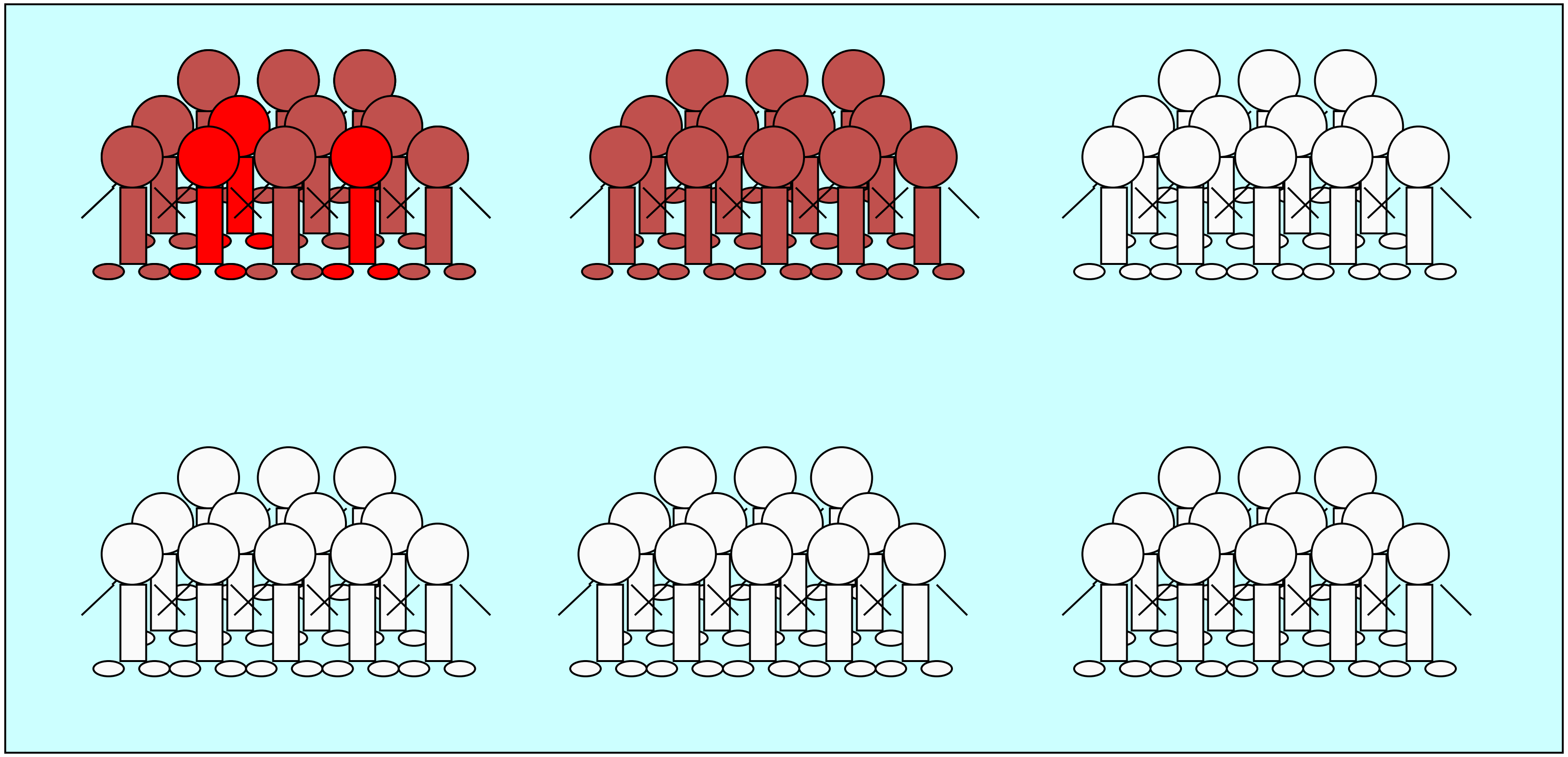
More positives (72 tests)



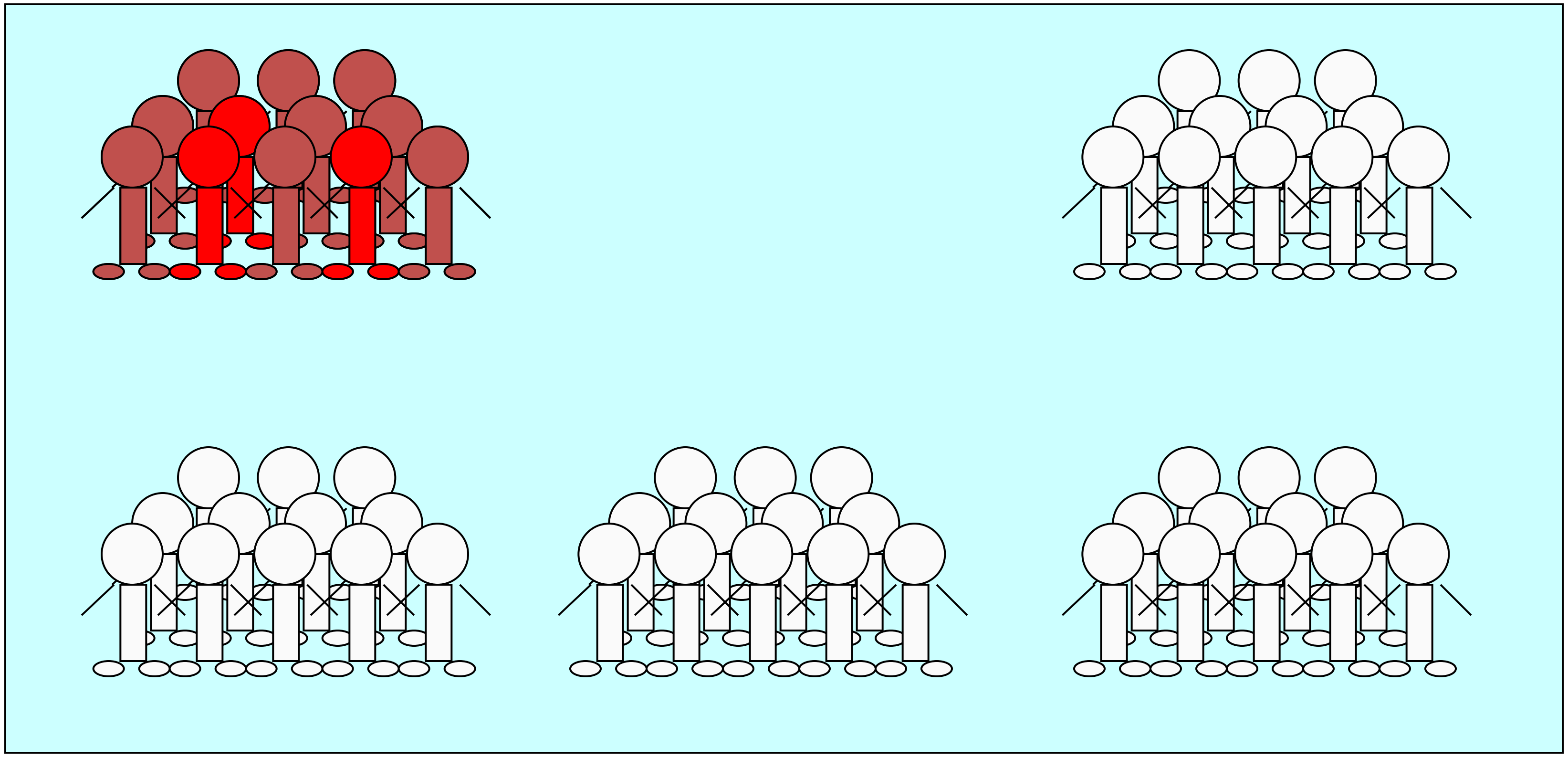
Test a group at one time



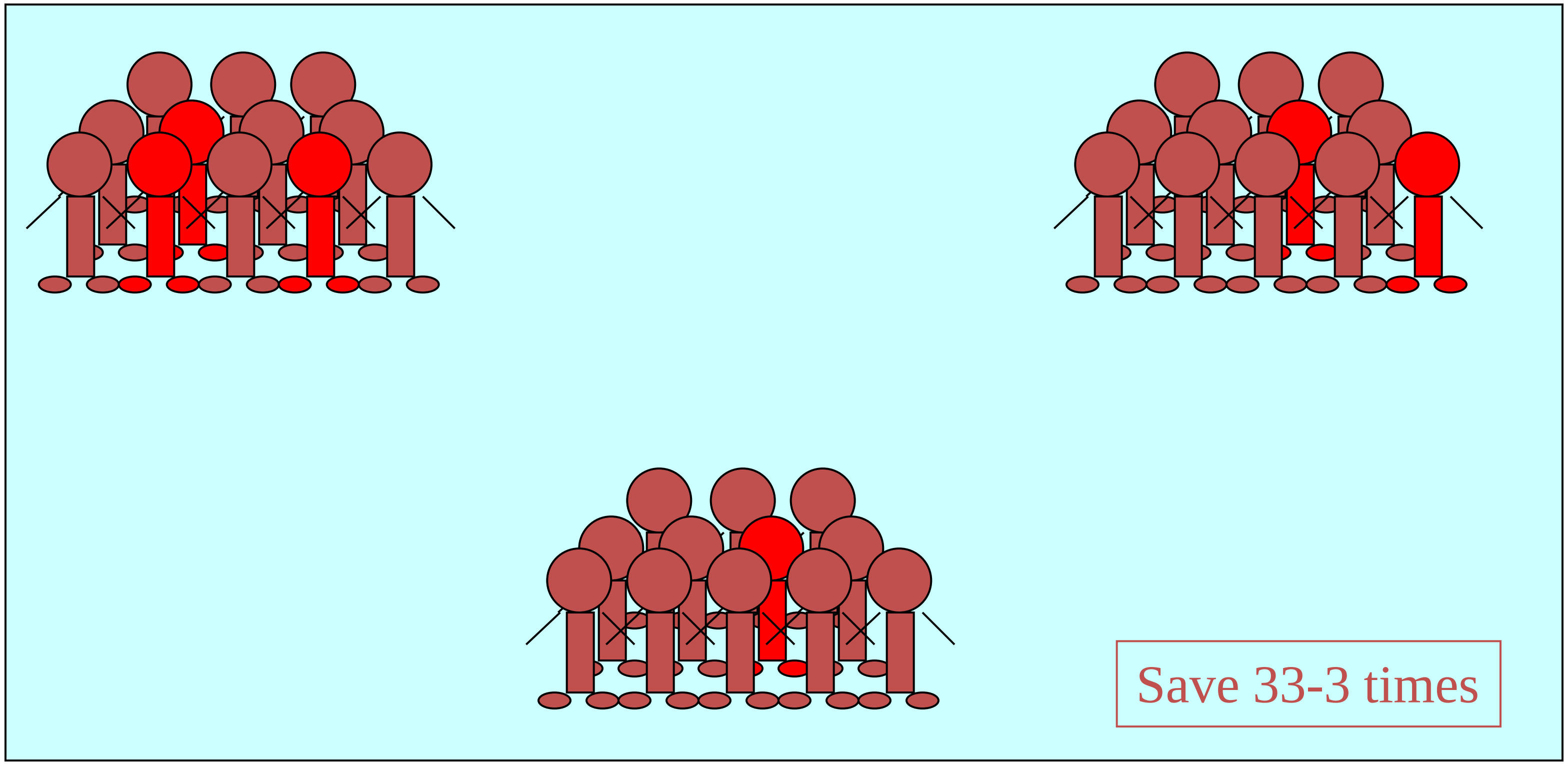
Some group is negative!



They are done!



Save a lot of Money! (Adaptive Idea)



What's next?

Save Money or Time

- An adaptive (sequential) algorithm conducts the tests *one by one* and the outcomes of all previous tests can be used to set up the later test. (*Save money!?)*
- A non-adaptive algorithm specifies a set of tests in advance so that they can be conducted *simultaneously*; thus forbidding using the information of previous tests. (*Save time!)*
- Usually, the first one takes less tests.

Non-adaptive Algorithm

We can use a matrix to describe a non-adaptive algorithm.

Items are indexed by columns and the tests are indexed by rows. Therefore, the (i, j) entry is 1 if the item j is included in the *pool* i (for test), and 0 otherwise.

An example (12 items and 9 tests)

1			1			1			1		
1				1			1			1	
1					1			1			1
	1		1					1		1	
	1			1		1					1
	1				1		1		1		
		1	1				1				1
		1		1				1	1		
		1			1	1				1	

The blanks are zeros.

Relation with designs

- Let $M = [m_{i,j}]$ be a txn matrix mentioned above. Then we can use n sets (ordered) S_i's to represent the matrix where

  $S_k = \{i : m_{i,k} = 1, i = 1, 2, \ldots, t\}, k = 1, 2, \ldots, n.$
- The following sets represent the above (0,1)-matrix:

  {1,2,3}, {4,5,6}, {7,8,9}, {1,4,7}, {2,5,8}, {3,6,9}, {1,5,9}, {2,6,7}, {3,4,8}, {1,6,8}, {2,4,9}, {3,5,7}. (Have you seen this collection of sets before?)

Set notation

- It is easier to apply combinatorial structures to construct pooling designs, therefore, we use sets (or codewords) for columns in non-adaptive algorithms.
- We shall use set notation most of the time.
- The set corresponding to a codeword (binary vector) is called the ***support*** of the codeword.

Can we find positives from the above matrix?

- Yes, we can if the number of positives is not too many, say at most 2, by running the 9 tests simultaneously corresponding to rows.
- The reason is that the union of (at most) 2 columns can not contain any other distinct column. (?)

Decoding Idea

- Since there are 12 items and the number of positives is at most 2, we have 1 + 12 + 66 possible inputs. (?)
- Let each input be a 12-dim column vector **x** and A be the above 9x12 matrix. Then the outcome (0,1)-vector **y** is obtained by A**x** such that its ith component is 1 if the ith component of A**x** is positive and 0 otherwise.
- If A is 1-1, then we are able to find the positives, see it?

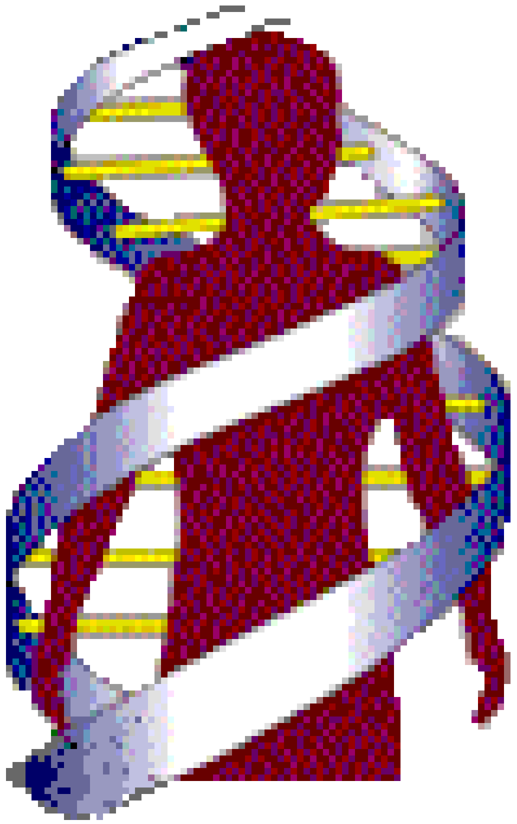
Outcome Vectors

- The vector **y** is an *outcome vector* which is corresponding to an input **x**.
- If A is 1-1, then we can decode **x** as long as we know its outcome vector.
- In order to get the job done with lower decoding complexity, extra properties for A is needed.
- Many of them!

Various Models

- *Errors* occurred.
- There are *inhibitors*.
- There is a *threshold*.
- Defective items are sets: *complex* model.
- *Competitive* model: the number of defectives is unknown.
- The defectives are *mutually obscuring*.

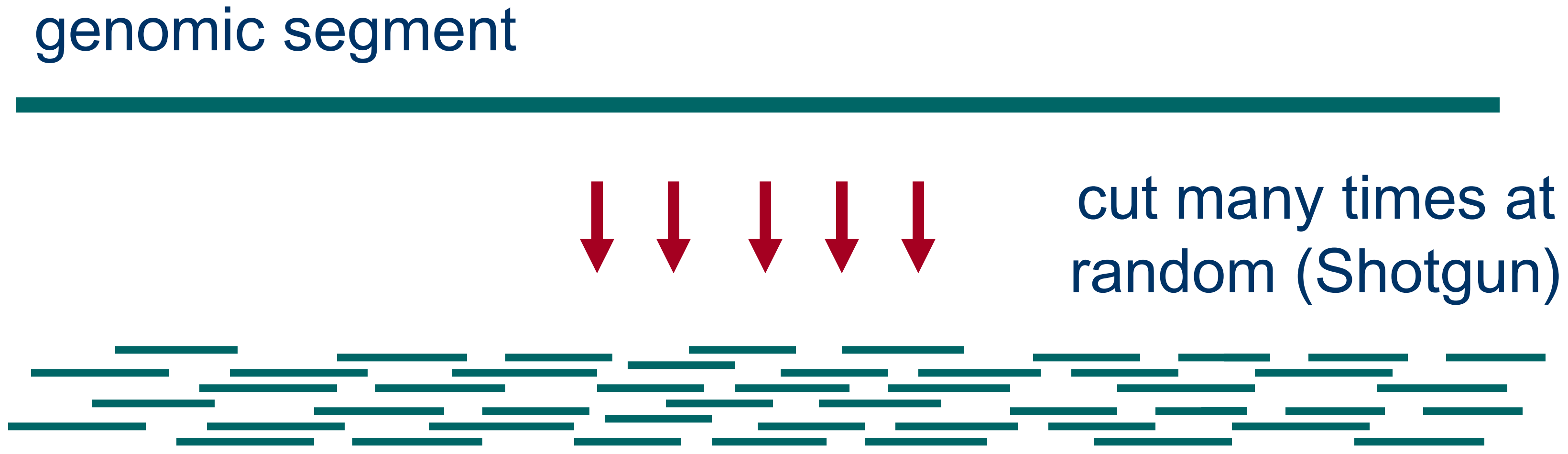
Human Genome Project



Shotgun Sequencing

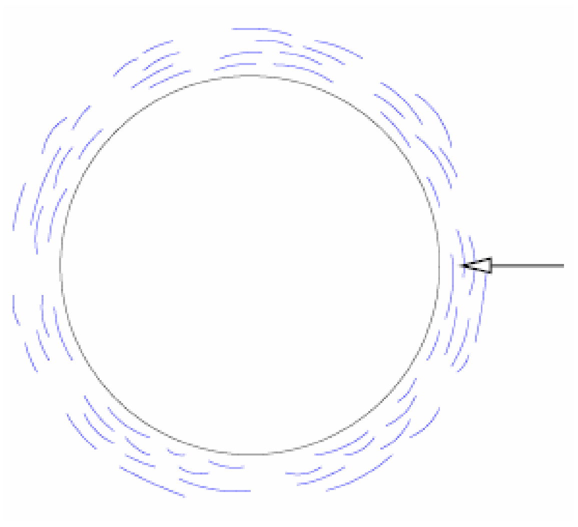
- Shotgun sequencing is a throughput technique resulting in the sequencing of a large number of bacterial genomes, mouse genomes and the celebrated human genomes. In all such projectss, we are left with a collection of contigs that for special reasons cannot be assembled with general assembly algorithms. Continued …

Random shotgun approach

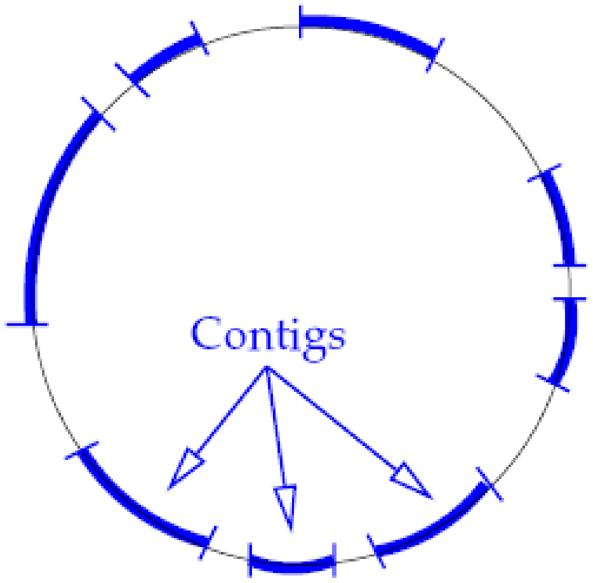


Whole-genome shotgun sequencing

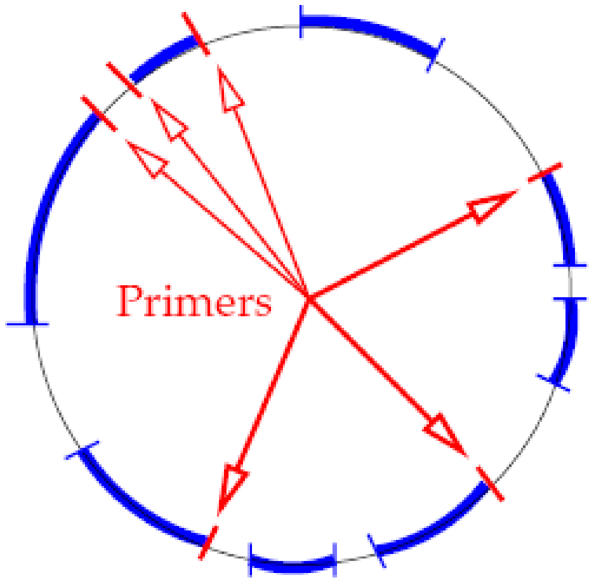
– Short reads are obtained and covering the genome with redundancy and possible gaps.



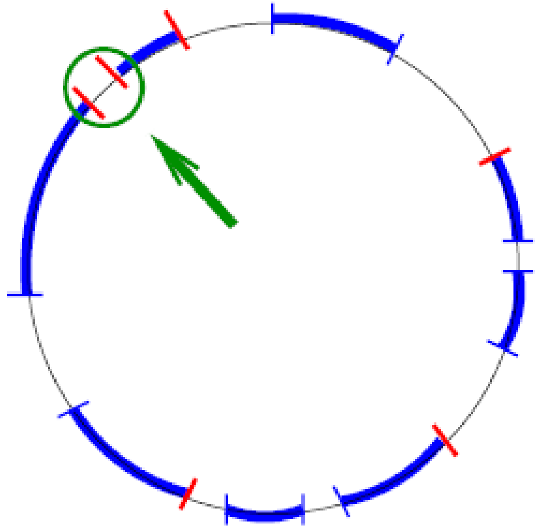
–Reads are assembled into contigs with unknown relative placement.

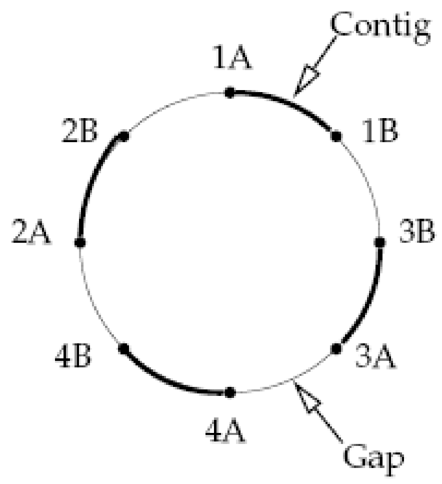


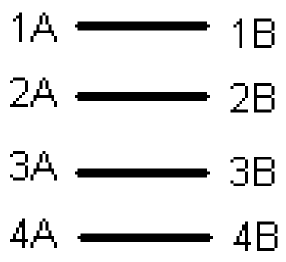
– Primers : (short) fragments of DNA characterizing ends of contigs.



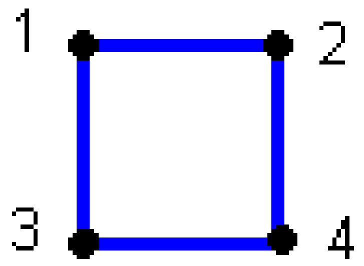
- A PCR (Polymerase Chain Reaction) reaction reveals if two primers are proximate (adjacent to the same gap).
- Multiplex PCR can treat multiple primers simultaneously and outputs if there is a pair of adjacent primers in the input set and even sometimes the number of such pairs.

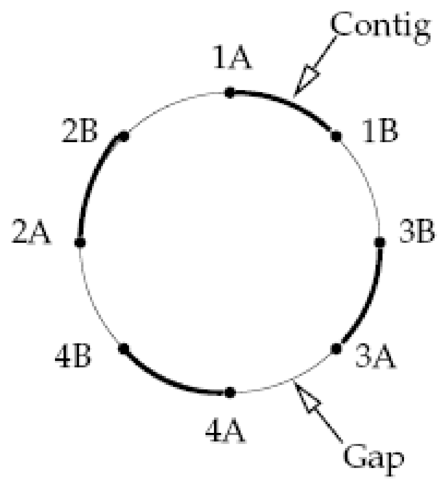




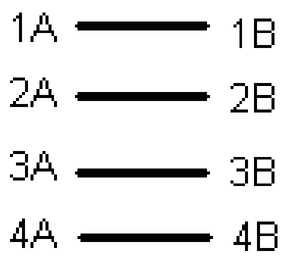


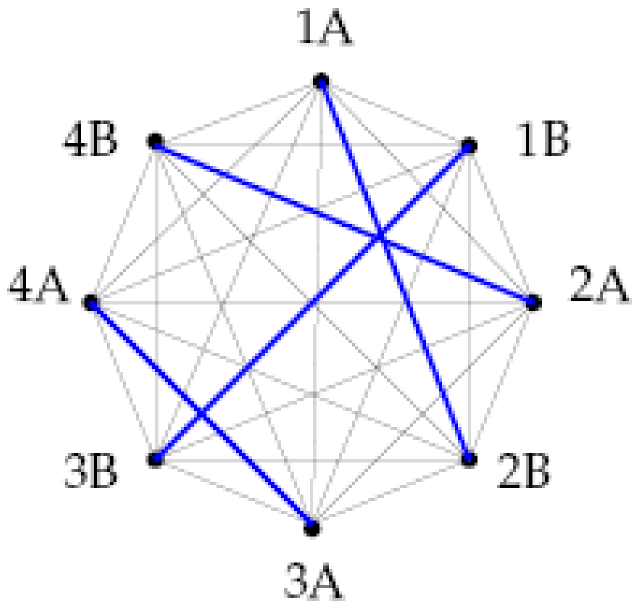
- Two primers of each contig are “mixed together”
  - Find a Hamiltonian cycle by PCRs!





- Primers are treated independently.
  - Find a perfect matching by PCRs.





Goal

- Our goal is to provide an experimental protocol that identifies all pairs of adjacent primers with as few PCRs (queries) (or multiplex PCRs respectively) as possible.

Mathematical Models

Hidden Graphs

(Reconstructed)

- Topology-known graphs, e.g. Hamiltonian cycle, matching, star, clique, bipartite graph, …, etc.

- Graphs of bounded degree
- Hypergraphs
- Graphs of known number of edges

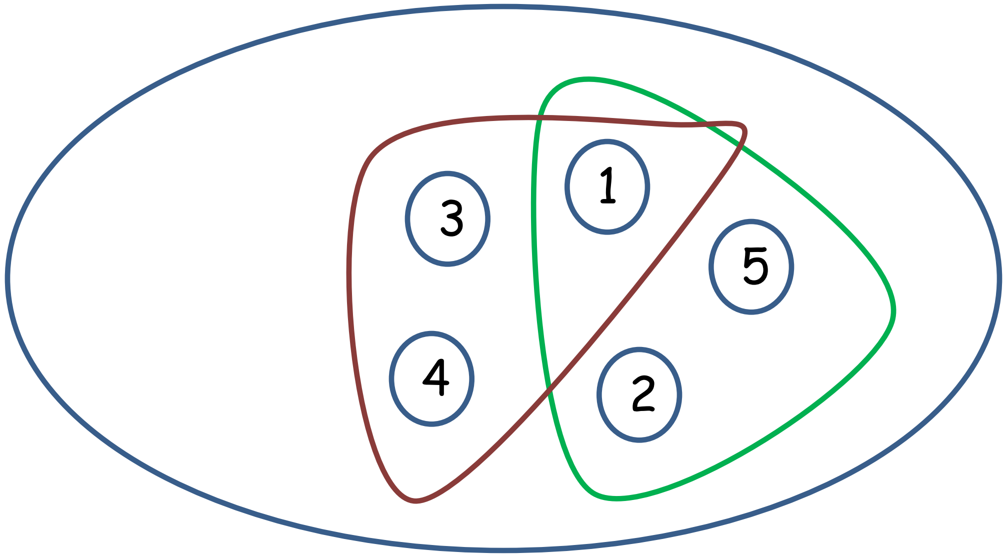
Idea

- Edge-detecting queries: $Q_G(S)$
- $Q_G(S) = 1$ if the induced subgraph of the vertex subset S of V(G) contains at least one edge of G and $Q_G(S) = 0$ otherwise.
- For convenience, we assume G is defined on $[1,n] = \{1, 2, \ldots, n\}$.

Hidden 3-uniform hypergraphs

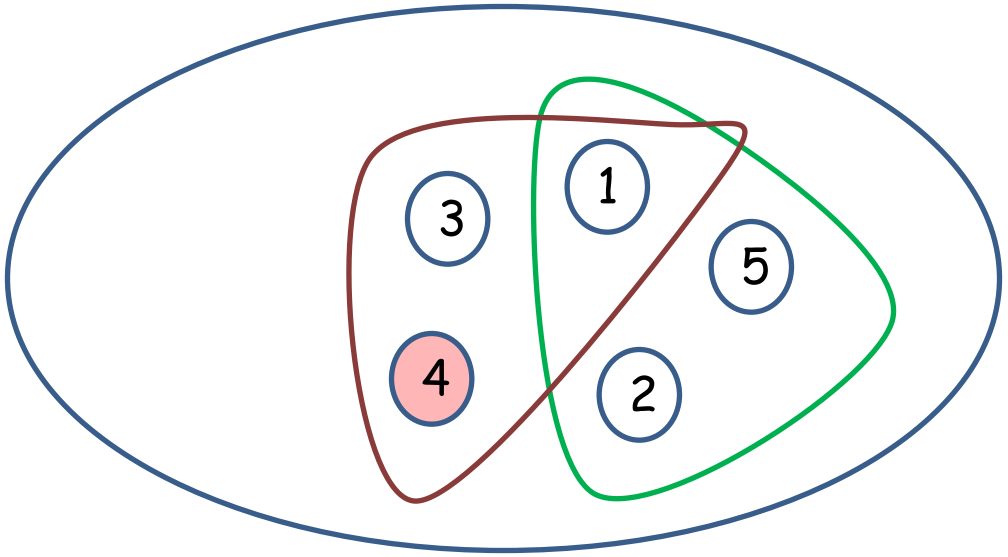
- The following example shows an idea of finding a hidden 3-uniform hypergraph.

Find a hyperedge



{1,2,3,4,5}	
Q({1,2,3,4,5})	Yes
Q({1,2,3})	No
Q({1,2,3,4})	Yes

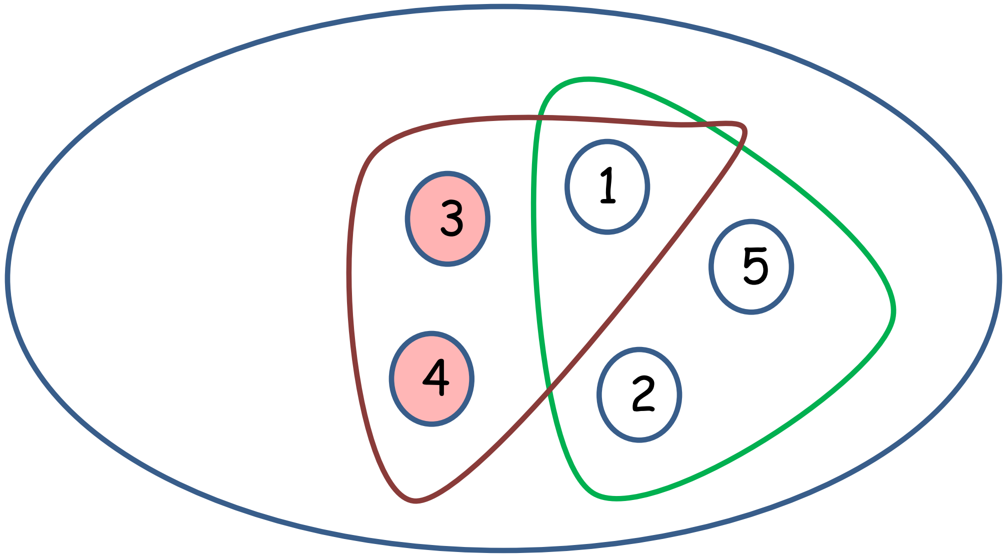
Find a hyperedge



Algorithm

{1,2,3,4,5}	
Q({1,2,3,4,5})	Yes
Q({1,2,3})	No
Q({1,2,3,4})	Yes
{4,1,2,3}	
Q({4,1,2})	No

Find a hyperedge



Algorithm

{1,2,3,4,5}	
Q({1,2,3,4,5})	Yes
Q({1,2,3})	No
Q({1,2,3,4})	Yes
{4,1,2,3}	
Q({4,1,2})	No
{4,3,1,2}	
Q({4,3,1})	Yes

Conclusion

- We can find a hidden r-uniform hypergraph G with $m \cdot r \cdot \log_2 n + (2^{r+2})^{1/2} r^r m^{r/2}$ edge-detecting queries where m is the number of edges in G.
- Note that if r = 2, then we can find a hidden graph with at most $m \cdot 2 \cdot \log_2 n + 9m$ queries. (with Huilan Chang and Chih-Huai Shih, Optimization Letters 2014)

Problems

- We believe that the best answer should be around ($mr\log_2 n$) queries, but not able to prove this fact.
- We have no ***good*** idea about *non-adaptive algorithm* for optimal result at this moment.

Further Remarks

- We can approach this study via different topics such as *combinatorial design, algebraic combinatorics, coding theory and graph theory*.
- Group testing does play an important role in applications such as *computational molecular biology, network security, image compression*, …, etc.

References

- A new construction of 3-bar-*separable* matrices via improved decoding of Macula construction (with F. K. Hwang), **Discrete Optim.**, 5(2008), 700-704.
- An upper bound of the number of tests in pooling designs for the *error-tolerant complex model* (with Hong-Bin. Chen and F. K. Hwang), **Optimization Letters**, 2(2008), no. 3, 425-431.
- The minimum number of e-vertex-cover among *hypergraphs* with e edges of given rank (with F. H. Chang, F. K. Hwang and B. C. Lin), **Discrete Applied Math**., 157(2009), 164-169.
- Non-adaptive algorithms for *threshold* group testing (with Hong-Bin Chen), **Discrete Applied Math.**, 157(2009), 1581-1585.

Continued

- Identification and classification problem on pooling designs for *inhibitor* models (with Huilan Chang and Hong-Bin Chen), **J. Computational Biology**, 17(2010), No. 7, 927-941.
- Reconstruction of *hidden graphs and threshold* group testing (with Huilan Chang, Hong-Bin Chen and Chi-Huai Shih), **J. Combin. Optimization**, 22(2011), no. 2, 270 – 281.
- Group testing with multiple *mutually-obscuring positives* (with Hong-Bin Chen), **Lecture Notes Computer Science,** 7777(2013), 557 – 568.
- *Threshold* group testing on *inhibitor* model (Huilan Chang and Chie-Huai Shih), **J. Computational Biology**, Vol. 20, No. 6, 2013, 1 – 7.

Continued

- New bound on *2-bar-separable* codes of length 2 (with Minquan Cheng, J. Jiang, Yuan-Hsun Lo and Ying Miao), **Des. Code Cryptography**, to appear.
- Learning a *hidden graph* (with Huilan Chang and Chie-Huai Shih), **Optimization Letters**, 8(2014), no. 8, 2341 – 2348.
- *Codes with identifiable parent property in multimedia finger printing* (with Minquan Cheng, J. Jiang, Yuan-Hsun Lo and Ying Miao), **Des. Code Cryptography**, to appear.

Keep Moving Forward!