

A Study of Group Testing Algorithms.


A Dissertation submitted in partial fulfillment of the
requirement for the M. Tech (Computer Science) degree
at the Indian Statistical Institute.

By
M.SATISH

Under the guidance of
Dr. Bimal Kumar Roy,
The Computer Science Unit,
Indian Statistical Institute,
203, Barrackpore Trunk Road,
Calcutta - 700 035.
India.

CERTIFICATE

This is to certify that the work described in the dissertation entitled *A Study of Group Testing Algorithms* has been undertaken by Mandavilli Satish under my guidance and supervision. The dissertation is found worthy of acceptance for the award of Degree of Master of Technology in Computer Science.


(Dr. Bim Kumar Roy)

Indian Statistical Institute,
Calcutta,
Date :- July 14, 1995.

Acknowledgements

I am deeply grateful to Dr. Bimal Kumar Roy for his guidance and advice in this dissertation, without which this could not have been possible. I am also deeply indebted to my colleagues and friends for sharing their views regarding this subject. I also thank the Dean of Studies, the CSSC and the Computer Science Unit for providing their facilities. I am grateful to my brother, Swamy, for being a constant source of inspiration.

Calcutta,
July, 1995.

(M.SATISH)

Contents

1	INTRODUCTION	2
2	SATISH—ALGORITHM	4
3	AFIS—ALGORITHM	8
4	SAFIS—ALGORITHM	15
5	DIHA—ALGORITHM	16
6	IMPLEMENTATION AND CONCLUSIONS	20
	BIBLIOGRAPHY	21

Chapter 1

INTRODUCTION

The GROUP TESTING problem may be described as follows. Consider a set of n items each of which being either defective or good. The objective is to identify all the defective items by a minimal no., of tests. Each test is performed on a subset of the n items and can have two possible outcomes either positive or negative. A positive outcome indicates that the subset under test is contaminated, i.e., it contains at least one defective item. A negative outcome indicates that the subset under test is pure, i.e., it contains only good items.

Group Testing was introduced by Dorfman in 1943. In subsequent years a large no., of papers were published on this subject. There are two basic models in the literature for this problem- a stochastic model and a deterministic model. In the stochastic model it is assumed that each item is assumed with some probability, and the objective is to minimize the expected no., of tests required to identify all the defective items. In deterministic approach it is assumed that the no., of defective items (or an upper bound on this number) is known in advance, and the objective is to minimize the maximum no., of tests required to identify all the defective items. In practical applications, the assumption of the deterministic model that there is an a priori information on the no., of defective items is not realistic.

APPLICATIONS :- It is useful in high speed network, medical examination, quantity searching and statistics.

To solve this we will study four algorithms. First one is developed by myself, Second

one is developed by A.B.Noy, Frank, Kessler and S.Kuttan. Third one is an improvement over the second. Fourth one is developed by Ding-Zhu and Haesun Park. These Algorithms are described in the following chapters.

Chapter 2

SATISH—ALGORITHM

This is a simple straight forward divide and conquer algorithm. Starting from the input set of items, the algorithm tests a set and bisects it when the set is found to be contaminated. Continue to do this until every item is either in a pure set or in a contaminated singleton. This algorithm is developed by me.

COMPLEXITY :-

Let $T(n, d)$ be the no., of tests required by the algorithm to identify among n items all d defective items. We will assume that $n = 2^l$.

We will prove that the worst case complexity of this algorithm is

$$2d(l - p) + 2^{p+2} - 2d - 1,$$

where $2^p < d < 2^{p+1}$, $n = 2^l$ for $d \neq 0$ and for $d = 0$, it is 1. To find the worst case complexity of this algorithm, we first solve the following equation.

$$T'(n, d) = 1 + T'(n, \lfloor \frac{d}{2} \rfloor) + T'(n, \lceil \frac{d}{2} \rceil),$$

$T'(n, 1) = 2l + 1$ and $T'(n, 0) = 1$, where $n = 2^l$.

To solve this equation, we first try to estimate the solution of this equation and prove that estimated value is the correct value. For this we try to calculate the values of this recursive equation for few values of d .

After some simplifications, we get,

$$\begin{aligned}
T'(n, 1) &= 2l + 1, & T'(n, 2) &= 4l - 1, & T'(n, 3) &= 6l - 5, \\
T'(n, 4) &= 8l - 9, & T'(n, 5) &= 10l - 15, & T'(n, 6) &= 12l - 21, \\
T'(n, 7) &= 14l - 21, & T'(n, 8) &= 16l - 33,
\end{aligned}$$

If we observe carefully we can find an order in the values of $T'(n, d)$.

$T'(n, d)$ value is $2ld + x$, where x value is as follows.

If d is in the interval $(2^p, 2^{p+1}]$ then x value is previous x value of $T(n, d)$, i.e., of $T(n, d-1)$ minus $2(p+1)$.

After some simple calculations, we see that the estimated value of $T'(n, d)$ is $2d(l-p) + 2^{p+2} - 2d - 1$ if d is $d \neq 0$ and if $d = 0$ then it is 1.

LEMMA :- If $d \neq 0$ then $T'(n, d) = 2d(l-p) + 2^{p+2} - 2d - 1$,

where $2^p < d \leq 2^{p+1}$,

and $T'(n, 0) = 1$.

PROOF :- Let us fix n and we will prove it by induction on d .

From the simple calculations we can prove that the result is true for $d = 1$.

Let us assume that the result is true for $d = 1, 2, 3, \dots, k-1, \forall n$

We will prove that the result is true for $d = k$. Let p be such that $2^p < k \leq 2^{p+1}$.

Then, $2^{p-1} < \lfloor \frac{k}{2} \rfloor, \lceil \frac{k}{2} \rceil \leq 2^p$ if $k \neq 2^p + 1$. Therefore,

$$\begin{aligned}
T'(n, k) &= 1 + 2\lfloor \frac{k}{2} \rfloor(l-1-p+1) + 2^{p+1} - 2\lfloor \frac{k}{2} \rfloor - 1 + 2\lceil \frac{k}{2} \rceil(l-1-p+1) + 2^{p+1} - 2\lceil \frac{k}{2} \rceil - 1 \\
&= 2k(l-p) + 2^{p+2} - 2k - 1,
\end{aligned}$$

as required.

Let $k = 2^p + 1$.

Then, $\lfloor \frac{k}{2} \rfloor = 2^{p-1}$ and $\lceil \frac{k}{2} \rceil = 2^{p-1} + 1$. Then,

$$T'(n, k) = \begin{cases} 1 + 2 * 2^{p-1}(l - 1 - p + 2) + 2^p \\ -2 * 2^{p-1} - 1 + 2(2^{p-1} + 1) \\ \times (l - 1 - p + 1) + 2^{p+1} - 2(2^{p-1} + 1) - 1 \end{cases}$$

$= 2k(l - p) + 2^{p+2} - 2k - 1$, (After some simplifications) as required. Therefore, for every fixed n , we have proved the lemma by induction on d Hence the lemma.

THEOREM :- If $2^p < k \leq 2^{p+1}$, $2^m < q \leq 2^{m+1}$ and $p > m+1$ then $\lceil \log(k-q) \rceil = p$ or $p - 1$.

PROOF :- From the given inequalities we can that $2^p - 2^{m+1} < k - q < 2^{p+1} - 2^m$

After some simplifications we can prove that,

$$\lceil \log(k - q) \rceil = p \text{ or } p + 1.$$

LEMMA :- If $2^p < d \leq 2^{p+1}$, then

$$T'(n, d) = 1 + T'(\frac{n}{2}, \lfloor \frac{d}{2} \rfloor) + T'(\frac{n}{2}, \lceil \frac{d}{2} \rceil),$$

$T'(n, 0) = 1$ and $T'(n, 1) = 2l + 1$, is the worst case complexity of this algorithm.

PROOF :- We will prove the lemma by induction on d for every fixed n such that $d \leq n$. For $d = 1$ we will prove as follows: $T(n, 1) = 1 + T(\frac{n}{2}, 1) + T(\frac{n}{2}, 0) = 2 + T(\frac{n}{2}, 1) = 2 + T(\frac{n}{4}, 0) + T(\frac{n}{4}, 1) + 1 = 2l + 1$, as required. Assume the result upto $k - 1$, where $k \geq 2$. Suppose, $T(n, k) = 1 + T(\frac{n}{2}, q) + T(\frac{n}{2}, k - q)$, $k > q \geq 1$, at worst case. Let p and m be such that, $2^p < k \leq 2^{p+1}$, and $2^m < q \leq 2^{m+1}$. w.l.g., we can assume that $p > m$. From the above theorem,

$$2^{p-1} < k - q \leq 2^p \text{ or } 2^p < k - q \leq 2^{p+1}$$

Case 1 :- $2^{p-1} < k - q \leq 2^p$.

Then, at the worst case

$$\begin{aligned} T(n, k) &= 1 + 2q(l-1-m) + 2^{m+2} - 2q - 1 + 2(k-q)(l-1-p-1) + 2^{p+1} - 2(k-q) - 1 \\ &= 2k(l-p) + 2^{p+2} - 2k - 1 + 2q(p-m-1) + 2^{m+2} - 2^{p+1} \end{aligned}$$

$$\text{Claim :- } 2q(p-m-1) + 2^{m+2} - 2^{p+1} \leq 0.$$

Maximum value of q is 2^{m+1} . Therefore, it is enough to show that,

$$2 * 2^{m+1}(p-m-1) + 2^{m+2} - 2^{p+1} \leq 0 \quad (\text{Since, } p \geq m+1)$$

i.e., $p-m \leq 2^{p-m-1}$ i.e., $p-m-1 < 2^{p-m-1}$, which is always true. Hence the claim.

At $m = p-1$, $T(n, \lfloor \frac{k}{2} \rfloor) = T'(n, \lfloor \frac{k}{2} \rfloor)$. Therefore, $T(n, \lfloor \frac{k}{2} \rfloor)$ is the maximum at $m = p-1$. Therefore, $2^{p-1} < q \leq 2^p$ and $2^{p-1} < k-q \leq 2^p$, then $T(n, \lfloor \frac{k}{2} \rfloor)$ is the maximum, in particular, at $q = \lfloor \frac{k}{2} \rfloor$ or $q = \lceil \frac{k}{2} \rceil$.

Therefore, $T'(n, d)$ is the maximum.

$$\text{Case 2:- } 2^p < k-q \leq 2^{p+1}$$

Therefore, at the worst case,

$$\begin{aligned} T(n, d) &= 1 + 2q(l-1-m) + 2^{l-1-m} + 2^{m+2} - 2q - 1 + 2(k-q)(l-1-p) + 2^{p+2} - 2(k-q) - 1 \\ &= 2k(l-p) + 2^{p+2} - 2k - 1 + 2pq - 2mq + 2^{m+2} - 2k. \end{aligned}$$

$$\text{Claim:- } 2pq - 2mq + 2^{m+2} - 2k \leq 0$$

$2pq - 2mq + 2^{m+2} - 2k \leq 0$ iff $q(p-m) + 2^{m+1} - k \leq 0$ iff $p-m < 2^{p-m-1}$, for $p-2 \geq m$ which is always true. Using the geometry we can easily see that the claim is true for the other values of m .

Hence the claim.

Hence $T(n, \lfloor \frac{k}{2} \rfloor)$ is at most $T'(n, \lfloor \frac{k}{2} \rfloor)$. In particular at $q = \lfloor \frac{k}{2} \rfloor$ or $q = \lceil \frac{k}{2} \rceil$, $T(n, \lfloor \frac{k}{2} \rfloor) = T'(n, \lfloor \frac{k}{2} \rfloor)$. Hence, $T'(n, d) = 1 + T'(\frac{n}{2}, \lfloor \frac{d}{2} \rfloor) + T'(\frac{n}{2}, \lceil \frac{d}{2} \rceil)$ is the worst case complexity of this algorithm.

Chapter 3

AFIS—ALGORITHM

The basic idea of the algorithm is as follows. Since the no., of defective items is unknown, the algorithm tries to estimate the value of d . It tests disjoint sets of items of sizes, $1, 2, \dots, 2^i$ until the first time a contaminated set is found. Namely, the answer for the first i tests was negative and last answer was positive. At this stage, the algorithm detected $1 + 2 + \dots + 2^{i-1} = 2^i - 1$, good items and a contaminated set of size 2^i . Using binary search this item can be detected by performing i additional tests. Since prior to the binary search the algorithm performed $i + 1$ tests, it follows that for the price of $2 \cdot i + 1$ tests the algorithm learned about 2^i items. In other words, the status of a new item is known by performing $2 \log a + 1$ tests.

The above-described strategy, called the doubling process, is the heart of the algorithm DOUBLE that is shown in this chapter. However, this strategy by itself can be modified. When d is small, the algorithm may perform many unnecessary tests on a large pure set. To overcome this difficulty, the algorithm can test all the items before it starts doubling. One more trick is as follows. The algorithm tests a set of size three. If the set is pure then algorithm can test the rest of the items and if the set is contaminated then the algorithm begins the doubling process. By testing three items instead of two sets one of two items and one of one item, the algorithm saved a test to spend on testing the rest of the items. If the set containing the three items is contaminated, then by two additional tests either two defective items are detected or a good item and a defective item is detected.

To implement the above idea we have used the following sets. U : the set of unknown

items D : the set of defective items, and G : the set of good items.

Initially, $U = \{0, 1, 2, \dots, n - 1\}$ is the set of all items and D and G are empty. These sets are updated accordingly when a defective item is detected or a good item is detected. Testing a set of items is done by procedure TEST that returns a positive answer for a contaminated set and a negative answer for a pure set.

Procedure DOUBLE :- This is the main procedure. In this the whole idea described above is implemented. This calls the following procedures.

Procedure THREE-TEST :- The input for this procedure is a contaminated set of three items. The procedure tests two items one at a time and finds either two defective items or one good item and one defective item are detected.

Procedure BINARY-TEST :- The input for this is a contaminated set of $0 \leq k \leq 2^i$ items, for $i \geq 2$. The procedure performs at most i tests and find one defective item.

Procedure FINAL-TEST :- This procedure tests the remaining items in the set U . The maximum size of U at this stage is 2.

Procedure DOUBLE :

```
U := {0, 1, 2, ...n - 1};
D :=  $\phi$ ;
G :=  $\phi$ ;
while( |U|  $\geq$  3 ) do
    {x, y, z} := arbitrary items from U;
    TEST( {x, y, z} );
    if positive THREE-TEST( {x, y, z} );
    if negative
        U := U \ {x, y, z};
        TEST(U);
        if negative
            G := G  $\cup$  U;
```

```

        U :=  $\phi$ ;
    if positive
        k := 4;
        repeat
            C := k arbitrary items from U or U if  $k \geq |U|$ 
            TEST(C);
            if positive
                x := BINARY-EST(C);
                D := D  $\cup$  {x};
                U := U \ {x};
                abort-repeat;
            if negative
                k := 2k;
                G := G  $\cup$  C;
                U := U \ C;
        end-repeat;
    end-while;
    FINAL-TEST;

```

end-lgorithm.

Procedure **THREE-TEST**({X, Y, Z})

```

    TEST({x});
    if positive
        D := D  $\cup$  {x};
        TEST({y});
        if positive D := D  $\cup$  {y};
        if negative G := G  $\cup$  {y};
        U := U \ {x, y};
    if negative

```

```

G := G ∪ {x};
TEST( {y} );
if positive
    D := D ∪ {y};
    U := U \ {x, y};
if negative
    D := D ∪ {z};
    U := U \ {x, z};

```

end-of-procedure.

Procedure **BINARY-TEST(C)**

```

k := |C|;
repeat
    C' := ⌊ $\frac{k}{2}$ ⌋ arbitrary items from C;
    TEST( C' );
    if positive C := C';
    if negative C := C \ C';
    k := |C|;
until k=1;
x := the only item in C;
return( x );

```

end-of-procedure.

Procedure **FINAL-TEST**

```

while( U ≠ ∅ ) do

```

```

    x := an arbitrary item from U;
    TEST( {x} );
    if positive D := D ∪ {x};
    if negative G := G ∪ {x};
    U := U \ {x};
end-while;

```

end-of-procedure.

COMPLEXITY :-

Consider the while loop in the DOUBLE algo. There are four possible flows of this loop.

Flow 1 :- A subset of three items is tested and found to be pure, and then the rest of the yet untested items are tested and found to be contaminated. Thereafter, $i - 2$ ($i \geq 2$) disjoint sets of sizes $4, 8, \dots, 2^{i-1}$ are tested and found to be pure and a subset of size 2^i is tested and found to be contaminated. Finally, BINARY-TEST is invoked and detects one defective item.

Flow 2 :- A subset of three items is tested and found to be contaminated. Then procedure THREE-TEST detects two defective items.

Flow 3 :- A subset of three items is tested and found to be contaminated. Then procedure THREE-TEST detects one defective item and one good item.

Flow 4 :- A subset of three items is tested and found to be pure, and the rest of the yet untested items is tested and found to be pure.

Clearly, defective items are detected only in flows 1-3 and by procedure FINAL-TEST. Let d_i be the total no., of defective items that are detected in all occurrences of Flow i , throughout the algo. Let d be the total no., of defective items. Then, $d - 2 \leq d_1 + d_2 + d_3 \leq d$.

LEMMA :- The total no., of tests performed in all occurrences of flows 2 and 3 during the run of the algorithm is $1.5d_2 + 3d_3$, and the total no., of items that are identified by these tests is $d_2 + 2d_3$.

LEMMA :- In each of the occurrences of flow 1, the algorithm performs $2 \log a + 1$ and detects one defective item and $a - 1$ good items, where $a = 2^i$ for some $i \geq 2$.

Let A_1, A_2, \dots, A_{d_1} be the d_1 subsets each of which contains $|A_j| - 1$ good items and one defective item, that are detected in the corresponding d_1 occurrences of Flow 1. Let $|A_j| = a_j$. Let $T(n, d)$ be the no., of tests required by the algo DOUBLE to identify among n items all d defective items.

LEMMA :- (A) $\sum_{j=1}^{d_1} a_j \leq n - d_2 - 2d_3$.

(B) $T(n, d) \leq \sum_{j=1}^{d_1} (2 \log a_j + 1) + 1.5d_2 + 3d_3 + 2$.

LEMMA :- For $0 < d \leq n/2$,

$T(n, d) \leq 2d \log(n/d) + d + 2$.

PROOF :- If $d_1 = 0$, then we have

$$T(n, d) \leq 1.5d_2 + 3d_3 + 2 \leq 2d \log(n/d) + d + 2$$

from the above lemma and the fact that $d \leq n/2$ gives the above inequality.

Suppose now that $d_1 > 0$. $T(n, d) \leq \sum_{j=1}^{d_1+d_2} (2 \log c_j + 1) + 1.5d_2 + 2$, where $c_j = a_j$ for $1 \leq j \leq d_1$ and $c_j = 2$ for $d_1 < j \leq d_1 + d_3$. Then after some simplifications we get,

$$\sum_{j=1}^{d_1+d_2} \log c_j \leq (d_1 + d_3) \log \frac{n - d_2}{d_1 + d_3}$$

Using this we obtain that $T(n, d) \leq 2(d_1 + d_3) \log \frac{n - d_2}{d_1 + d_3} + (d_1 + d_3) + 1.5d_2 + 2 = 2(d_1 + d_3) \log \frac{(n - d') + (d_1 + d_3)}{d_1 + d_3} - \frac{d_1 + d_2}{2} + \frac{3}{2}d' + 2 \cong G(d_1 + d_3)$, where $d' = d_1 + d_2 + d_3$

since $G(x)$ is an increasing of x in the interval $0 < x \leq n - d'$, it follows that $T(n, d) \leq G(d_1 + d_3) \leq G(d') = 2d' \log \frac{n}{d'} + d' + 2$. Since $d - 2 \leq d' \leq d$, this bound implies that

$$T(n, d) \leq \max_{0 \leq i \leq 2} \left\{ 2(d - i) \log \frac{n}{d - i} + (d - i) + 2 \right\} = 2d \log \frac{n}{d} + d + 2,$$

where the equality follows from the fact that $d/n \leq 1/2 < \sqrt{2}/e$.

LEMMA :- For any d , $T(n, d) \leq 2n$.

PROOF :-

CLAIM :- The no., of tests performed so far is at most twice the no., items that are identified so far.

PROOF OF CLAIM :- The claim is proved by induction. Initially, the claim is true. For flow 1 the claim is true since $2i + 1 \leq 2 \cdot 2^i$ for any positive integer i . In both flow 2 and flow 3, three tests are performed and two items are identified. In flow 4 two tests are performed and at least three items are identified. Finally, FINAL-TESTS performs exactly one test per item.

The Lemma follows from the above claim.

Chapter 4

SAFIS—ALGORITHM

This is basically an improved version of the AFIS Algorithm. This is improved by myself.

(1) In **THREE-TEST**, the procedure tests two items. If both the tests are negative then the third one is a defective one. Therefore we can remove these three items from the unknown set. In the original algorithm, only first and last items are removed from the unknown set.

(2) In the **BINARY-TEST**, the procedure finds a defective item. To find this, the algorithm uses divide and conquer technique. At any stage, if a test is performed on a set in this procedure and gives a negative answer then the entire set can be removed from the set of unknown items. In the original algorithm the set is not removed even if the set is pure.

Chapter 5

DIHA—ALGORITHM

This is basically, an extension to the bisecting algorithm, described in the in the satish-algo., Starting from the input set of items, the algorithm tests a set and bisects it when the set is found to be contaminated. Continue to do this until every item is either in a pure set or in a contaminated singleton. A lot of tests are wasted in the bisecting algorithm in the following situation: When X_1 and X_2 both are contaminated sets with X_2 is a subset of X_1 , then the test on X_2 would render the test on X_1 useless. From this observation, we make the following improvement: Once a contaminated set is found, a defective item is eliminated from the set.

ALGORITHM-DIHA :-

```
input S;  
G :=  $\phi$ ;  
D :=  $\phi$ ;  
Q := {S};  
repeat  
    pop X from queue Q;  
    TEST( X );
```

```

    if  $X$  is pure then  $G := G \cup X$ ;
    else
        ELIMINATE( $X$ );
        if  $|X| = 1$  then push  $X$  into queue  $Q$ ;
        if  $|X| \geq 2$  then bisect  $X$  into  $X_1$  and  $X_2$  such that  $|X_1| = 2^{\lceil \log |X| \rceil - 1}$ 
        and  $X_2 = X \setminus X_1$  and push  $X_1$  and  $X_2$  into queue  $Q$ ;
    until  $Q$  is  $\phi$ ;

```

end-of-algorithm.

Procedure **ELIMINATE**(X)

```

     $Y := X$ ;
    while(  $|Y| \geq 2$  ) do
        begin
             $Y_1 := \lceil Y/2 \rceil$  items from  $Y$ ;
            TEST(  $Y_1$  );
            if  $Y_1$  is contaminated then  $Y := Y_1$ ;
            else
                 $Y := Y \setminus Y_1$ ;
                 $G := G \cup Y_1$ ;
                 $X := X \setminus Y_1$ ;
        end-while;
     $X := X \setminus Y$ ;
     $D := D \cup Y$ ;

```

end-of-procedure.

COMPLEXITY :-

LEMMA :- Let n be a power of 2. Then for $0 \leq d \leq n$,

$$T(n, d) \leq d \log \frac{n}{d+1} + 4d - \log(d+1) + 1.$$

PROOF :- Let $n = 2^u$, $v = \lfloor \log(d+1) \rfloor$ and $v' = \log(d+1) - v$. Note that detecting a defective item from a set of size s by procedure ELIMINATE needs $\lfloor \log s \rfloor$ TESTS. It is clear that ELIMINATE is applied to at most one set of size n , two sets of sizes between $1 + n/4$ and $n/2$, ..., and in general at most 2^i sets of sizes between $1 + n/2^{i+1}$ and $n/2^i$ for $i < v$. Thus, the no., of tests required by procedure is at most

$$d \log \frac{n}{d+1} + 2d - \log(d+1).$$

Now consider the tree T^* which is built up by the bisecting process as follows: The node set of T^* consists of S and all sets X' and X'' appearing in the computation. A node X is the father of two sons X' and X'' if and only if X' and X'' are obtained by bisecting X after eliminating a defective item. Clearly, every internal node is contaminated set from which a defective item is eliminated. Thus T^* has at most d internal nodes. It follows that the total number of nodes of T^* is at most $2d + 1$. Therefore,

$$T(n, d) \leq d \log \frac{n}{d+1} + 4d - \log(d+1) + 1.$$

LEMMA :- Let $d = d_1 + d_2$ and $n = n_1 + n_2$, where $d_1 \geq 0, d_2 \geq 0, n_1 > 0$ and $n_2 > 0$. Then,

$$d_1 \log \frac{n_1}{d_1} + d_2 \log \frac{n_2}{d_2} \leq d \log \frac{n}{d}.$$

LEMMA :- For $1 \leq d \leq n$,

$$T(n, d) \leq d(\log \frac{n}{d} + 4).$$

PROOF :- We prove it by induction on d . For $d = 1$, the algorithm finds the only defective item with $\lfloor \log n \rfloor + 1$ TESTS which is clearly bounded by $\log n + 4$. For $d > 1$, the algorithm eliminates the first defective item with $\lfloor \log n \rfloor + 1$ Tests's and bisects the remaining $n - 1$ items into two sets S' and S'' of sizes n_1 and n_2 , respectively, where $n_1 = 2^{u-1}$ with $u = \lfloor \log(n-1) \rfloor$. Suppose that S' and S'' contain d_1 and d_2 respectively. Then the no., of tests Tests's for identifying items in S' is at most

$$d_1(\log \frac{n_1}{d_1+1} + 4) - \log(d_1+1) + 1.$$

Case 1 :- $d_2 > 0$. By the induction hypothesis, all defective items in S'' can be identified in at most $d_2(\log(n_2/d_2) + 4)$ tests. Thus the total no., of tests is at most $d(\log \frac{n}{d} + 4)$, using the above lemma and after some simple calculations.

Case 2 :- $n \neq 2^u + 1$ and $d_2 = 0$. In this case $u = \lceil \log(n-1) \rceil = \lceil \log n \rceil$ and the algo., uses one test to detect S'' . Thus the total no., of tests is at most $d(\log \frac{n}{d} + 3)$.

Case 3 :- $n = 2^u + 1$ and $d_2 = 0$. In this case, $u+1 = \lceil \log n \rceil$ and $n_1 = n_2 = 2^{u-1}$ and the total no., tests is at most $d(\log \frac{n}{d} + 4)$.

It is an immediate consequence that $T(n, d) \leq d \log \frac{n}{d} + 4d$, for $1 \leq d \leq n$

Hence the lemma.

Chapter 6

IMPLEMENTATION AND CONCLUSIONS

Arrays have been used for sets. End of array is -1 . For, queue also array has been used.

According to the worst case complexity 1st algorithm is the worst. But if the algorithms are tested using random inputs, 1st algorithm is the best in the following case : If the ratio between the no., of defectives and the total no., of items is high then 1st algorithm is giving the best results. SAFIS algo., also giving very good results, in fact it is giving the second best. Otherwise, DIHA algo., is giving the best results. The reason why DIHA algo., giving the best result is : it tries to delete a defective item whenever it finds a contaminated set. Therefore, if the ratio is less then DIHA algo., giving the best results. In all cases AFIS algo., giving the worst results.

BIBLIOGRAPHY

1. Ding-Zhu Du and Haesun Park, On competitive Group Testing, *Siam.J.*, Vol. 23, No. 5 pp.1019-1025, October 1994.
2. A.B.Noy, F.K.Hwang, I.Kessler, S.Kutten, A New Competitive Algorithm For Group Testing, *Discrete applied Mathematics*, vol.52(1994) 29-38.
3. Robert Dorfman, The Detection Of Defective Members Of Large Populations, *Ann. Math. Statist.* 14(1943) pp. 436-440.
4. Ding Zhu Du and F.K.Hwang, Minimizing A Combinatorial Function, *Siam J. Disc. Math.* Vol. 3, No., 4, December 1982, pp 523-528.
5. Some combinatorial aspects of designs useful in Group Testing by G.M.Saha.
6. *Concrete Mathematics* by Ronald L. Graham, Donald E. Knuth, Oren Patashnik