Abhimanyu M A **(11111002)** , Sumesh T A **(11111065)**
Department of Computer Science and Engineering
IIT Kanpur

September 26,2011

ASSIGNMENT 2

Problem 1. **How to use random sampling ?**

We shall be doing the following.

Let our original graph be $\boldsymbol{G = (V, E)}$, we now need to find a subgraph $\boldsymbol{G_S = (V, E_S)}$ which is a 3-approximate distance preserver, that is the distance between $\boldsymbol{(u, v)}$ in $\boldsymbol{G_S}$ is atmost 3 times the distance between $\boldsymbol{(u, v)}$ in the original graph $\boldsymbol{G}$.

Algorithm

We assume that the original graph is maintained as an adjacency list, from this we will construct an augmented adjacency list where each entry $\boldsymbol{u}$ in the list of $\boldsymbol{v}$ also has a pointer to entry $\boldsymbol{v}$ in the list of $\boldsymbol{u}$. This is done as follows.

For each entry $\boldsymbol{i}$ from $\boldsymbol{1}$ to $\boldsymbol{n}$. If $\boldsymbol{(i, j)}$ is there in the original adjacency list, and $\boldsymbol{j > i}$ we add $\boldsymbol{(i, j)}$ to list of $\boldsymbol{i}$ and $\boldsymbol{(j, i)}$ to the list of $\boldsymbol{j}$. Since they are added at the same time we will can have pointers from each to the other. Hence we get an augmented adjacency list, the reason we augmented the adjacency list is to support delete operation in $\boldsymbol{O(1)}$ time.

(a) Let $\boldsymbol{E_S = \phi}$.

(b) With a probability $\boldsymbol{p}$, we shall select each vertex and add it to $\boldsymbol{A}$. We shall fix on the value of $\boldsymbol{p}$ later. We shall also introduce the concept of clusters initially each of the vertex $\boldsymbol{u}$ will be a cluster by itself, and will be referred to as the center of the cluster.

(c) For each vertex $\boldsymbol{u \notin A}$ we shall do the following

   i. If for $\boldsymbol{u}$ there is no edge $\boldsymbol{(u, v)}$ where $\boldsymbol{v \in A}$ we shall add all the edges of the vertex to $\boldsymbol{E_S}$.

   ii. If $\boldsymbol{u}$ has an edge $\boldsymbol{(u, v_i)}$ where $\boldsymbol{v_i \in A}$, then take the edge $\boldsymbol{(u, v_i)}$ of minimum weight add it to $\boldsymbol{E_S}$ and also add all edges $\boldsymbol{(u, w_j)}$ where $\boldsymbol{weight(u, w_j) < weight(u, v_i)}$ where $\boldsymbol{w \notin A}$, we will keep all other edges since in $\boldsymbol{G}$ since we might later need to add them. In short add the minimum weight edge of u that links it to $\boldsymbol{A}$, and all edges of weight less than the minimum weight. We shall also add $\boldsymbol{u}$ to the the cluster of $\boldsymbol{v_i}$

(d) For all edges $\boldsymbol{(u, v)}$ belonging to the same cluster, that has not been added. we remove the edge from the set. This completes our building of the cluster.

(e) What now exists is the edges between the clusters, for each vertex $\boldsymbol{v \in C_i}$ where $\boldsymbol{C_i}$ is a cluster, for all $\boldsymbol{j \neq i}$ we add the minimum weight edge $\boldsymbol{(u, v)}$ where $\boldsymbol{u \in C_j}$.

(f) Remove all other edges.

Proof of Correctness

We shall now prove that the algorithm does indeed give a correct result.

To prove that the distance is atmost 3 times the actual distance all we need to prove that is that each edge (u, v) can be replaced by $(u, x), (x, y)(y, v)$ where $weight(u, x) < (u, v)$, $weight(x, y) < (u, v)$ and $weight(y, v) < (u, v)$.

(a) Now if either u or v doesn't belong to any cluster then we have added all the edges and hence we will not have a problem.

(b) Now if both u and v belong to the same cluster with cluster centre w,

   i. If $u == w$ or $v == w$ we are done every node in the cluster is directly linked to the centre.

   ii. If $weight(u, v) < weight(u, w)$ or $weight(u, v) < weight(v, w)$ we have already added the edge when we added u or v to the cluster.

   iii. Else we know that both $weight(u, w) < weight(u, v)$ and $weight(v, w) < weight(u, v)$ hence $weight(u, w)+weight(v, w) < 2*weight(u, v)$. Hence the construction satisfies the condition necessary for a 3-approximate distance preserving subgraph.

(c) Now we consider the case that u and v belong to different clusters, let there be an edge (u, w) where $w, v \in C_i$ and $weight(u, w) < weight(u, x) \forall x \in C_i$ where C_i is centred on x Now consider the following cases.

   i. If $w == v$ we are done since we have added the edge (u, v).

   ii. Else consider the set of edges $(u, w), (w, x), (x, v)$. We already know that (u, w) is the least weight edge hence $weight(u, w) < weight(u, v)$, we also know that $weight(x, v) < weight(u, v)$ since otherwise we would have added the edge (u, v) when we added v to the cluster i, now we can also see that $weight(w, x) < weight(u, w) < weight(u, v)$ since (w, x) was added in the first phase and (u, w) in the second phase, hence by the proximity constraint. Hence we can replace (u, v) can be replaced by $(u, x), (x, y)(y, v)$ where $weight(u, x) < (u, v)$, $weight(x, y) < (u, v)$ and $weight(y, v) < (u, v)$.

Bounding expected number of edges

Before we go into the proof of that we find that $E[|A|]$ Since each element is added with probability p, $E[|A|] = np$ Now let $np = q$.

(a) In the second step we add the edge (u, v) which is the minimum for all $v \in A$, and all edges lesser than that, since each vertex is selected to the cluster with probability p, Assume that all the edges are ordered, then the edges incident on the vertices forming the cluster form a partition of the set of edges, since each vertex is taken uniformly and randomly the expected number of elements selected for cluster $= \delta p$ where δ is the degree of the vertex. Now since we consider that they are ordered, the set of all edges with weight lower than edge (u, v) is nothing but the expected size of the first partition, which can be easily seen as $\frac{\delta}{\delta p} = \frac{1}{p}$. Now if we consider each vertex, then we can apply linearity of expectation and get the following result.

$$\text{Expected contribution of a single vertex} = \frac{1}{p} \tag{1-1}$$

$$\text{Total contribution by Linearity of Expectation} \leq \sum_{\forall v \in V} \frac{1}{p} \tag{1-2}$$

$$\leq \frac{n}{p} \tag{1-3}$$

The inequality comes due to the fact that the same edge may be selected more than once We know that this has to be $O(n^{3/2})$ Hence we set $p = \frac{1}{\sqrt{n}}$. We can then see that it will fit the other contributions also into the limit.

(b) Now that we have fixed the probability p we can analyse the number of edges added by the vertices which does to belong to any cluster. Again we shall partition it into the number of edges added by each vertex.

$$\text{Expected number of edges added by single vertex} = \delta Pr[\text{Vertex is selected}] \quad (1\text{-}4)$$
$$E[X_i] = \delta Pr[\text{Vertex is selected}] \quad (1\text{-}5)$$

Where δ is the degree of the vertex.

$$Pr[\text{Vertex is selected}] = Pr[\text{None of it's neighbours belong to a cluster}] \quad (1\text{-}6)$$
$$= \left(1 - \frac{1}{\sqrt{n}}\right)^{\delta} \quad (1\text{-}7)$$

$$\text{Expected number of edges added by single vertex} = \delta \left(1 - \frac{1}{\sqrt{n}}\right)^{\delta} \quad (1\text{-}8)$$

We can have two cases $\delta \leq \sqrt{n}$ and $\delta > \sqrt{n}$. Now for $\delta \leq \sqrt{n}$

$$E[X_i] < \delta \quad (1\text{-}9)$$
$$< \sqrt{n} \quad (1\text{-}10)$$

This comes since the probability $\left(1 - \frac{1}{\sqrt{n}}\right)^{\delta}$ is atmost 1.
Now for $\delta > \sqrt{n}$. Let $\delta = k\sqrt{n}$ where $k > 1$.

$$E[X_i] = k\sqrt{n}\left(1 - \frac{1}{\sqrt{n}}\right)^{k\sqrt{n}}. \quad (1\text{-}11)$$

We know that asymptotic value of $\left(1 - \frac{1}{x}\right)^{x} = \frac{1}{e}$. Substituting that we get.

$$E[X_i] = k\sqrt{n}\left(\frac{1}{e}\right)^{k} \quad (1\text{-}12)$$
$$= \sqrt{n}\left(\frac{k}{e^k}\right) \quad (1\text{-}13)$$
$$< \sqrt{n} \cdot 1 \quad (1\text{-}14)$$

Hence for both cases we have $E[X_i] < \sqrt{n}$
So

$$E[X] \leq \sum_{\forall v \in V} \sqrt{n} \quad (1\text{-}15)$$
$$\leq n^{\frac{3}{2}} \quad (1\text{-}16)$$

The $\leq$ sign comes due to the fact that few edges may be selected more than once.

(c) Now we shall consider the edges added due to the inter cluster linkings. This is very much straight forward to see, each vertex adds a single edge for each cluster. If we consider a single vertex

$$\boldsymbol{E}[\boldsymbol{X_i}] = \boldsymbol{E}[|\boldsymbol{A}|] \tag{1-17}$$

We have already proved initially that, $\boldsymbol{E}[|\boldsymbol{A}|] = \boldsymbol{np}$. Substituting $\boldsymbol{p} = \sqrt{\boldsymbol{n}}$ we get

$$\boldsymbol{E}[\boldsymbol{X_i}] = \frac{\boldsymbol{n}}{\sqrt{\boldsymbol{n}}} \tag{1-18}$$

$$= \sqrt{\boldsymbol{n}}. \tag{1-19}$$

Again by linearity of expectation

$$\boldsymbol{E}[\boldsymbol{X}] \leq \sum_{\forall v \in V} \sqrt{\boldsymbol{n}} \tag{1-20}$$

$$\leq \boldsymbol{n}^{\frac{3}{2}} \tag{1-21}$$

The $\leq$ sign comes due to the fact that few edges may be selected more than once.

Hence total number of edges selected on expectation $= \boldsymbol{3n^{\frac{3}{2}}}$.

Since we have the total number of edges count at the end of the algorithm, we shall see if this is greater than $\boldsymbol{6n^{\frac{3}{2}}}$, if yes we shall drop all the result and run the algorithm again. We have the result by Markov Inequality

$$\boldsymbol{Pr}\left[|\boldsymbol{E_S}| > \boldsymbol{6n^{\frac{3}{2}}}\right] \leq \frac{\boldsymbol{3n^{\frac{3}{2}}}}{\boldsymbol{6n^{\frac{3}{2}}}} \tag{1-22}$$

$$\boldsymbol{Pr}\left[|\boldsymbol{E_S}| > \boldsymbol{6n^{\frac{3}{2}}}\right] \leq \frac{\boldsymbol{1}}{\boldsymbol{2}} \tag{1-23}$$

This now can be considered as a coin toss experiment with success probability $= \frac{\boldsymbol{1}}{\boldsymbol{2}}$. Then the expected number of trials before success $= \boldsymbol{2} = \boldsymbol{O(1)}$. Hence we get a Las Vegas algorithm by repeating the algorithm only $\boldsymbol{O(1)}$ time. This completes our answer.

$\longrightarrow$ Answer

Problem 2. **Random choices may work!**

(a) Deterministic algorithm will have a specific ordering for scanning the children. Without loss of generality let us assume that the order is from left to right. We can fix the deterministic alogrithm to fix the values of the node.

i. For the root set value as either **0** or **1**.
ii. Now for each of the children from left to right assign values as follows.
iii. leftChild.value = parent.value.
iv. middleChild.value = not (parent.value).
v. rightChild.value = parent.value

Proof. Now consider for the deterministic algorithm at all points of time when the left and middle children's values are not conclusive enough to decide the value of the parent hence we need to scan right child also. Hence we will need to scan all nodes of the tree. Now we need to prove the number of nodes in the tree. Thus tree with height 0 tree will have 1 node, tree with height 2 will have 3 leaves and tree with height 3 will have 9 leaves and so on. Thus tree with height $\boldsymbol{h}$ will have $\mathbf{3^h}$ leaf nodes. Hence a deterministic algorithm must must read all n = $\mathbf{3^h}$ leaves when the left and middle children of the parent node have contradicting values. □

(b) The algortihm can be stated as follows.

i. Randomly choose any two leaf nodes which has same parent.

ii. If they have same value then value of parent node is equal to that of value of these two children.

iii. Else scan the third child and set the value of the parent is equal to the value of this child.

Analysis: For cases where leaf nodes are having values 0,0,0 and 1,1,1. we can immediately set the value after scanning any two nodes. We will now look at worst case scenario where we have one value is different.

The expected number query = $\mathbf{(2 * 1)}$+ (1* Prob(first two values are different))

Now, Prob(first two values are different)) = $\frac{\mathbf{2}}{\mathbf{3}}$.

The expected number query $\leq \mathbf{(2 * 1)} + \frac{\mathbf{2}}{\mathbf{3}} = 2.67$

Now for the total tree of height $\boldsymbol{h}$,we can see that total number of expected query = $\mathbf{(2.67)^h}$

now we know that $\mathbf{3^h = n}$

Height $\boldsymbol{h = log_3 n}$

The expected number query = $\mathbf{(2.67)^{log_3 n}}$

$\leq \boldsymbol{n^{log_3 2.67}}$

$\leq \boldsymbol{n^{.8939}}$

$\longrightarrow$ Answer

Problem 3. **Two-dimensional Pattern Matching** We shall first show that the finger print function has the required properties.
Let the string $\boldsymbol{x = \{0, 1\}*}$

(a) We shall now give a deterministic algorithm to generate the finger print function. Let $\boldsymbol{I = I_2}$ the $\mathbf{2 \times 2}$ Identity Matrix. Now for a string $\boldsymbol{x}$ we shall prove that $\boldsymbol{M(x)}$ is defined by induction.

i. For strings of length **0&1** it is trivially true since $\boldsymbol{M(\epsilon), M(0), M(1)}$ are well defined.

ii. Now let us assume it is defined for all strings of length $\leq \boldsymbol{k}$

iii. Now let $\boldsymbol{x = \{0,1\}^{k+1}}$ be a string of length $\boldsymbol{k+1}$. Let $\boldsymbol{x = yz}$ such that $\boldsymbol{y = \{0,1\}^k}$ and $\boldsymbol{z = 0,1}$. We know that both of them are well defined since they are strings of length $\leq \boldsymbol{k}$. From the definition $\boldsymbol{M(x) = M(y) \times M(z)}$, since $\boldsymbol{M(y)}$ and $\boldsymbol{M(z)}$ are well defined, $\boldsymbol{M(x)}$ is also well defined.

(b) To prove that

$$\boldsymbol{M(x) = M(y) \Rightarrow x = y} \tag{3-1}$$

We shall define a deterministic algorithm to retrieve the string given the fingerprint function, proving that this can be done deterministically in effect proves that the function from string $\boldsymbol{x}$ to $\boldsymbol{M(x)}$ is bijective and hence $\boldsymbol{M(x) = M(y) \Rightarrow x = y}$. We shall once again use induction to prove our hypothesis.

i. For strings of length **0&1** it is true since $\boldsymbol{M(\epsilon), M(0), M(1)}$ are well defined, and distinct.

ii. Now let us assume that for all strings of length $\leq \boldsymbol{k}$ we will be able to find $\boldsymbol{x}$ given $\boldsymbol{M(x)}$.

iii. Now let $\boldsymbol{x = \{0,1\}^{k+1}}$ be a string of length $\boldsymbol{k+1}$. Let $\boldsymbol{x = yz}$ such that $\boldsymbol{y = \{0,1\}^k}$ and $\boldsymbol{z = 0,1}$.

$$\boldsymbol{Let\ M(y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix}} \tag{3-2}$$

If $\boldsymbol{z = 0}$.

$$\boldsymbol{M(x) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times [r]1011} \tag{3-3}$$

$$\boldsymbol{M(x) = \begin{bmatrix} a+b & b \\ c+d & d \end{bmatrix}} \tag{3-4}$$

If $\boldsymbol{z = 1}$.

$$\boldsymbol{M(x) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \times [r]1101} \tag{3-5}$$

$$\boldsymbol{M(x) = \begin{bmatrix} a & a+b \\ c & c+d \end{bmatrix}} \tag{3-6}$$

Since only for $\boldsymbol{b = 0}$ or $\boldsymbol{c = 0}$ we can see that if the last character was a **0** then the current matrix then atleast one entry in the left will be strictly greater than the value of the right, and vice versa if the last character was a **1**, hence given a matrix we can find out what the last character was deterministically, and since we have assumed that we can find out for the smaller matrix, inductively we can apply this until we reach the identity matrix. Hence we can define a bijective function from $\boldsymbol{M(x) \rightarrow x}$. Hence

$$\boldsymbol{M(x) = M(y) \Rightarrow x = y} \tag{3-7}$$

(c) Now to prove the bound of the fibonacci number we shall look again at the multiplication, as can be seen and was proved in the last example, each element in the matrix. We shall also prove this by induction.

i. For strings of length $\mathbf{1}$ it is true since we can see that all entries of $\boldsymbol{M(1), M(0)}$ are within 1, the $\mathbf{1}^{\boldsymbol{s}}\boldsymbol{t}$ fibonacci number.

ii. Now let us assume that for all strings of length $\leq \boldsymbol{k}$ we will be able to find $\boldsymbol{x}$ given $\boldsymbol{M(x)}$.

iii. Now let us assume that

$$\boldsymbol{M(x)} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{bmatrix} \tag{3-8}$$

Now given $\boldsymbol{z = 0, 1}$ we can have

$$\boldsymbol{M(x)} = \begin{bmatrix} \boldsymbol{a+b} & \boldsymbol{b} \\ \boldsymbol{c+d} & \boldsymbol{d} \end{bmatrix} \tag{3-9}$$

or

$$\boldsymbol{M(x)} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{a+b} \\ \boldsymbol{c} & \boldsymbol{c+d} \end{bmatrix} \tag{3-10}$$

In both these cases the numbers are the sum of previous elements, and we also have proved that either $\boldsymbol{a > b}$ and $\boldsymbol{c > d}$ or vice versa, holds, so we can see that $\boldsymbol{a, b}$ are upper bounded by consecutive fibonacci terms, and hence their sum is bounded by the next fibonacci term.

One Dimensional Pattern Matching

Consider we have two string $\boldsymbol{x}$ of length $\boldsymbol{n}$ which is the original string and $\boldsymbol{y}$ of length $\boldsymbol{m}$ which is the pattern which we need to look for in $\boldsymbol{x}$. We know that $\boldsymbol{n \geq m}$.

The deterministic algorithm is as follows.
First we compute $\boldsymbol{M(z)modp}$ of the first $\boldsymbol{m}$ characters of $\boldsymbol{x}$. Now for each new character we first divide by the M(0) or M(1) corresponsing to the character $\boldsymbol{m}$ steps before, and then multiply with the next character, and check this with $\boldsymbol{M(y)}$.

$$\boldsymbol{M(0)^{-1}} = \begin{bmatrix} \mathbf{1} & \mathbf{-1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3-11}$$

$$\boldsymbol{M(1)^{-1}} = \begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \mathbf{-1} & \mathbf{1} \end{bmatrix} \tag{3-12}$$

$$\boldsymbol{M(z) = (M[z-m]^{-1} * M(z-1) * M[z])modp} \tag{3-13}$$

Here $\boldsymbol{M(z)}$ is the fingerprint of the last $\boldsymbol{m}$ characters till $\boldsymbol{z^{th}}$ character. and $\boldsymbol{M[z]}$ is the fingerprint of the $\boldsymbol{z^{th}}$ charcter.

If we do not have $\boldsymbol{modp}$ the algorithm will always give the correct solution, and also we can see that it has very low incremental cost. Now if we select $\boldsymbol{p}$ from range $\boldsymbol{0..i}$ then we have the number of factors of $\boldsymbol{M(x)}$ is restricted by $\boldsymbol{m}$ where $\boldsymbol{m}$ is the length of the pattern hence .

Probability of false positive = Probabilitiy that $\boldsymbol{p}$ has a factor in $\boldsymbol{m}$ hence

$$\boldsymbol{Pr[Error] = (mlogi)/i} \tag{3-14}$$

If $i = cmlogm$ we have

$$Pr[Error] = 1/c \tag{3-15}$$

Two Dimensional Pattern Matching

Here for the first $m \times m$ entries we compute it deterministically, now for each of the other entries

$$A(i,j) = (_{k=j-m}j(A[z-m,j]^{-1} * M(i-1,j) * M[i,j])modp) \tag{3-16}$$

This problem is infact no different from matching a m^2 length string, from the point of view of the error probability. Again the probability holds except that here the length is $= m^2$

$$Pr[Error] = (m^2logi)/i \tag{3-17}$$

If $i = 2cm^2logm$ we have

$$Pr[Error] = 1/c \tag{3-18}$$

$\longrightarrow$ Answer

Problem 4. **Approximate Ham Sandwich Cut Algorithm**

(a) Randomly and uniformly sample p red point and p blue points.

(b) We shall now find the ham sandwich cut of these points using brute force, that is for all pairs of points we will check if the line passing through them is the ham sandwich cut or not, since if any other line which is a ham sandwich cut can be moved and rotated such that it passes through two of the points without disturbing any other points.

(c) Hence for each pair of $\binom{2p}{2}$ points find the line, and check if it is the ham sandwich cut. Each point can be verified in $O(1)$ time, and hence for each line we have $O(p)$ time, hence total time taken to find ham sandwich cut of $2p$ points $= O(p^3)$.

(d) Now we shall say that this is the approximate ham sandwich cut of the whole set of points, and based on the value of ϵ we shall fix the value of p

Since the complexity is $O(p^3)$ we shall select upto $O(n^{\frac{1}{3}})$. So we set $p = clogn$

For the proof we shall draw parallels between this problem and that of finding the approximate median of a given sample, Now let the line found by the equation be $y = mx + c$ we shall now rotate all the point about the origin such that the line is now given by $x = C$, now we can easily see that C has to be the approximate median of the x co-ordinates of both red and blue points.
We can also note that after the rotation we need not bother about the y co-ordinates since they are partitioned only by the x co-ordinate.

We shall now consider the red points alone, we know that $\boldsymbol{C}$ is the median of the $\boldsymbol{x}$ co-ordinates of $\boldsymbol{p}$ red points sampled.Now we shall look at the probability that this is a $\boldsymbol{(1+\epsilon)}$ approximation of the median of $\boldsymbol{x}$ co-ordinates of the points. Let $\boldsymbol{R}$ be the set of Red points and $\boldsymbol{B}$ be the set of blue points. Let $\boldsymbol{R_S}$ be the set of sampled points of $\boldsymbol{R}$ and similarly $\boldsymbol{B_S}$ We shall now define the random variable

$$\boldsymbol{X} \quad = \quad \text{The no. of elements in } \boldsymbol{R_S} \text{ which have rank} \leq \boldsymbol{(1-\epsilon)\frac{n}{2}} \tag{4-1}$$

Since we can see that each point is independently chosen without considering the other choices we have

$$\begin{array}{rl} \boldsymbol{X_i = 1} & \text{if the } \boldsymbol{i^{th}} \text{ sampled element has rank} \leq \boldsymbol{(1-\epsilon)\frac{n}{2}} \\ \boldsymbol{0} & \boldsymbol{otherwise} \end{array} \tag{4-2}$$

(4-3)

$$\boldsymbol{X} \quad = \quad \sum_{i=1}^{p} \boldsymbol{X_i} \tag{4-4}$$

$$\boldsymbol{E[X]} \quad = \quad \sum_{i=1}^{p} \boldsymbol{E[X_i]} \tag{4-5}$$

$$= \quad \sum_{i=1}^{p} \frac{\boldsymbol{(1-\epsilon)\frac{p}{2}}}{\boldsymbol{p}} \tag{4-6}$$

$$= \quad \boldsymbol{(1-\epsilon)\frac{p}{2}} \tag{4-7}$$

Since the median is a very symmetric condition we can see that

$$\boldsymbol{P}[\text{Algorithm is wrong}] \quad = \quad \boldsymbol{2P[X > \frac{p}{2}]} \tag{4-8}$$

We shall later bound this to a suitable value to get a very high probability.

$$\boldsymbol{P[X > \frac{p}{2}]} \quad = \quad \sum_{i=\frac{p}{2}}^{p} \binom{\boldsymbol{p}}{\boldsymbol{i}} \left(\frac{\boldsymbol{1-\epsilon}}{\boldsymbol{2}}\right)^{i} (1 - \frac{\boldsymbol{1-\epsilon}}{\boldsymbol{2}})^{p-i} \tag{4-9}$$

The following conversion is possible since $\boldsymbol{(\frac{1}{2})^p}$ is the maximum value for the product.

$$\boldsymbol{P[X > \frac{p}{2}]} \quad \leq \quad \sum_{i=\frac{p}{2}}^{p} \binom{\boldsymbol{p}}{\boldsymbol{i}} \left(\frac{\boldsymbol{1}}{\boldsymbol{2}}\right)^{p} \tag{4-10}$$

Applying Sterlings approximation and the fact that $\boldsymbol{\frac{p}{i} \leq 2}$ we get.

$$\boldsymbol{P[X > \frac{p}{2}]} \quad \leq \quad \sum_{i=\frac{p}{2}}^{p} \frac{\boldsymbol{e}^{\boldsymbol{p}}}{\boldsymbol{4}} \tag{4-11}$$

$$\boldsymbol{P[X > \frac{p}{2}]} \quad \leq \quad \frac{\boldsymbol{e}^{\boldsymbol{p}}}{\boldsymbol{4}} \tag{4-12}$$

Setting $\boldsymbol{p = clogn}$

$$\boldsymbol{P[X > \frac{p}{2}]} \leq \boldsymbol{C'n^{-clog4/e}} \tag{4-13}$$

$$\boldsymbol{P[ErrorRed]} = \boldsymbol{2 * P[X > \frac{p}{2}] = 2 \times C'n^{-clog4/e}} \tag{4-14}$$

Asymptotically this value approaches zero hence we have very high probability bound, Similarly we can get the same bound for the set of Blue points hence total error probability

$$\boldsymbol{P[Error]} \leq \boldsymbol{P[ErrorRed] + P[ErrorBlue]} \tag{4-15}$$

By Union Theorem

$$\boldsymbol{P[Error]} = \boldsymbol{4 \times C'n^{-clog4/e}} \tag{4-16}$$

Hence the running time of our algorithm is $\boldsymbol{O(log^3 n)}$

Since we will anyway use $\boldsymbol{O(n)}$ time to read the input, if we set $\boldsymbol{p = n^{\frac{1}{3}}}$ then we shall actually get much better bounds without losing out on the time complexity

$\longrightarrow$ Answer

Submitted by Abhimanyu M A ***(11111002)*** *, Sumesh T A* ***(11111065)*** *on September 26,2011.*