Last week several students wondered whether it was fair to compare randomized algorithms and deterministic algorithms.

<table>
<thead>
<tr>
<th>Deterministic algorithm-</th>
<th>Randomized algorithm-</th>
</tr>
</thead>
<tbody>
<tr>
<td>always outputs</td>
<td>may sometimes not</td>
</tr>
<tr>
<td>the right answer</td>
<td>output the right answer</td>
</tr>
<tr>
<td>always runs</td>
<td>may sometimes not</td>
</tr>
<tr>
<td>efficiently</td>
<td>run efficiently</td>
</tr>
</tbody>
</table>

Is this "FAIR"?? Of course not.

We demand less from randomized algorithms,

But, in return, we expect randomized algorithms to do more - be more efficient, be simpler.

A new question: is it **useful** to have randomized algorithm?

- It depends on the situation and the alternatives.
- It depends on the probability of error/inefficiency.

(Great examples from class discussion of use of randomized algo: Google search, Watson)
In this lecture:
A general tool for design and analysis of algorithms that are randomized, and

\[
P(\text{bad things happening}) \xrightarrow{n \to \infty} 0
\]

---

The Chernoff Bound

\[X \sim B(n, p) \quad \text{[each succeeds w.p. } p, \text{ } X \text{ is the number of successes]}
\]

\[\forall r > 0, \quad P(X \geq \mathbb{E}[X] + r) \leq e^{-\frac{r^2}{2n}}.
\]

---

Central limit theorem: for large \(n\), \(X\) normal with expectation \(\mathbb{E}[X]\), variance \(\mathbb{V}[X]\)

| prob. | \[\frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mathbb{E}[X])^2}{2\sigma^2}}\] | \(\mathbb{E}[X]\) |
Galton's Board

For each peg, the ball goes left w.p. \( \frac{1}{2} \) and right w.p. \( \frac{1}{2} \) independently.

If we interpret right as \( +1 \), small as \( -1 \), the limit to which the algorithm falls is \( \Sigma x_i \) for independent \( \pm 1 \) random variables.

If you repeat this experiment with many balls, you find an approximate normal distribution in the beans.
Remarks

* There are lots of versions of the Chernoff bound out there. Some with better parameters, some more general.

* In CLRS C.5, Thm C.8, there's the classic proof of Chernoff by considering the random variable $e^{a(X - \mathbb{E}(X))}$ ($a$ is a parameter) and applying Markov.

A proof by Impagliazzo and Kabanets, 2010:

$$X = X_1 + X_2 + \cdots + X_n$$

where $X_n - X_n$ are independent.

We'll pick a sub-sum by taking each $k \leq n$ to the sub-sum with probability $q$, where $q$ is a parameter that will be set later.

\begin{align*}
(4) \Pr(\bigwedge_{i \text{ picked}} X_i = 1) &= \Pr(\bigwedge_{i \text{ picked}} X_i = 1 \mid \sum_{i \text{ picked}} X_i \geq \mathbb{E}[X] + r) \\
&\quad \cdot \Pr(\sum_{i \text{ picked}} X_i \geq \mathbb{E}[X] + r) + \\
&\quad \cdot \Pr(\sum_{i \text{ picked}} X_i < \mathbb{E}[X] + r) \\
&\geq \Pr(\bigwedge_{i \text{ picked}} X_i = 1 \mid \sum_{i \text{ picked}} X_i \geq \mathbb{E}[X] + r) \cdot \Pr(\sum_{i \text{ picked}} X_i \geq \mathbb{E}[X] + r)
\end{align*}
$$P\left( \bigwedge_{i \text{ picked}} X_i = 1 \mid \sum X_i \geq E[x] + r \right) \geq (1-q)^{n-(E[x]+r)}$$

(There are \( n-(E[x]+r) \) zero's among \( X_1, \ldots, X_n \) and we shouldn't pick any of them)

$$\Rightarrow \quad P\left( \bigwedge_{i \text{ picked}} X_i = 1 \right) \geq (1-q)^{n-(E[x]+r)} \cdot P\left( \sum X_i \geq E[x] + r \right)$$

(2) $$P\left( \bigwedge_{i \text{ picked}} X_i = 1 \right) \leq \sum_{k, \text{ the number of } i's\text{ picked } k \in 0, \ldots, n} \binom{n}{k} q^k \cdot (1-q)^{n-k} \cdot p^k$$

**Binomial formula**

\[
(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}
\]

Using binom,

$$P\left( \bigwedge_{i \text{ picked}} X_i = 1 \right) \leq \left( q \frac{p}{1-q} + 1 - q \right)^n$$

Combining the conclusions of (1) and (2),

$$\left(1-q\right)^{n-(E[x]+r)} P\left( \sum X_i \geq E[x] + r \right) \leq \left( q \frac{p}{1-q} + 1 - q \right)^n.$$ 

Theorem follows for

$$q = q = \frac{r}{(E[x]+r)(1-\frac{E[x]}{n})}.$$
Applications of Chernoff in this lecture

- Tighter analysis of Quick-Sort.
- Monte-Carlo sampling.
- Amplification for randomized algorithms.

Quick-Sort (A)

0. If n = 0, stop.
1. Pick uniformly at random i.e. \( i \in \{1, \ldots, n\} \)
2. Partition A around \( A[i] \).
3. Sort the elements \( \leq A[i] \) recursively.
4. Sort the elements \( \geq A[i] \) recursively.

```
3 4 1 8 6 9
```

```
3 4

3

5 8 6 9

5 8

6

A

∅
```

∅
In recitation: Quick-Sort has $\Theta(n \log n)$ expected running time.

We'll show: The probability that Quick-Sort runs in time $\geq cn \log n$ is at most $\frac{1}{n}$ for some constant $c \geq 1$.

Define indicator random variables: For an element $a$ in the array and a depth $t$ of the Quick-Sort run,

$$X_{a,t} = \begin{cases} 0 & \text{in } a's \text{ array, a middle pivot was chosen} \\ 1 & \text{o/w} \end{cases}$$

(An element is in the middle if its order is between $\frac{n}{10}$ and $\frac{9n}{10}$)

* For technical reasons, even after $a$'s sub-array is gone, we'll assume pivots are still being picked and the probability to pick a middle pivot is $0.8$

Example
$$P(X_{a,t} = 1) = 0.2$$

$$\forall a \quad X_{a,0}, X_{a,1}, X_{a,2} \ldots \text{ are independent.}$$

(In contrast, \( \{X_{a,t}\}_{t=1}^T \text{ for a fixed } t \text{ are dependent})$$

Let \( T = 100 \ln n \).

By Chernoff, \( \forall a \)

$$P\left( \sum_{t=0}^{T} X_{a,t} > E\left[ \sum_{t=0}^{T} a_{t} \right] + 0.2T \right) \leq e^{-\frac{(0.2T)^2}{2T}} = \frac{1}{n^2}$$

Union Bound For events \( E_1 \ldots E_n \), \( P(V \cap E_i) \leq \sum_{i=1}^{n} P(E_i) \)

$$\Rightarrow P\left( \bigvee_{a} \sum_{t=0}^{T} X_{a,t} > 0.4T \right) \leq n \cdot \frac{1}{n^2} = \frac{1}{n}$$

If \( \forall a \sum_{t=0}^{T} X_{a,t} \leq 0.4T \), then the size of a's subarray shrunk up to 0.6T times.

$$\left( \frac{9}{10} \right)^{0.6T} \cdot n < 1$$

$$\Rightarrow$$ All of the arrays shrunk to \( \emptyset \).

$$\Rightarrow$$ Array is sorted within depth \( T = O(\ln n) \).

$$\Rightarrow$$ Running time = \( O(n \ln n) \). \( \square \)
How to keep on track when arguing about randomized algo?

[The big problem: we're very used to]
deterministic algo

* Identify the random choices of the algo.
* After fixing the random choices, we have a deterministic algo.

**Example** For Quick sort,

random choices = pivots choices

we analyze variables = depth of recursion tree for elements a₁—aₙ

<table>
<thead>
<tr>
<th>random choice 1</th>
<th>depth a₁</th>
<th>depth a₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>n</td>
<td>1gn</td>
</tr>
<tr>
<td>random choice 2</td>
<td>½gn</td>
<td>31gn</td>
</tr>
<tr>
<td>M</td>
<td>½gn</td>
<td>31gn</td>
</tr>
</tbody>
</table>

We're checking in what fraction of the rows you have a depth that is more than T, where T was chosen so T = 0(n)
Monte-Carlo Sampling

There is a large population \( U \).

In some of the population occurs a phenomenon \( S \subseteq U \).

(e.g. \( S = \{ x \in U \mid x \text{ votes republican} \} \).

\( S = \{ x \in U \mid x \text{ has cancer} \} \).

Given \( x \in U \) can test whether \( x \in S \) or not.

How to estimate \( \frac{|S|}{|U|} \) efficiently?

Solution

- Pick u.a.r. \( x_1, \ldots, x_k \in U \).
- Output \( \frac{1}{k} \sum_{i=1}^{k} I_{x_i \in S} \).

By Chernoff, (can similarly get \( \hat{s} \leq \frac{|S|}{|U|} - r \))

\[
P( \hat{s} \geq \frac{|S|}{|U|} + r ) \leq e^{-\frac{-2r^2}{k}}
\]

By linearity of expectation \( \mathbb{E}[\hat{s}] = \mathbb{E} \left[ \sum_{i=1}^{k} I_{x_i \in S} \right] \)

\[
= \frac{1}{k} \sum_{i=1}^{k} \mathbb{E}[I_{x_i \in S}]
= \frac{1}{k} \sum_{i=1}^{k} \frac{|S|}{|U|}
= \frac{|S|}{|U|}
\]
Amplification

Suppose we are given a randomized algorithm that always runs in time $T$, but outputs an incorrect answer with prob. $\frac{1}{3}$.

Then we can get a new randomized algorithm that always runs in time $O(T\log \frac{1}{\varepsilon})$ and outputs an incorrect answer with prob. $\leq \varepsilon$.

How?

The new algorithm will run the original algorithm $k = O(\log \frac{1}{\varepsilon})$.

It will output the answer that repeats in the largest number of runs.

$$P(\text{new algo } \text{ incorrect}) \leq P\left( \sum I_j \geq \frac{k}{2} \right) \leq e^{-\frac{\left(\frac{k}{2} - \frac{k}{3} - \frac{k}{6}\right)^2}{2\frac{\left(\frac{k}{2} - \frac{k}{3} - \frac{k}{6}\right)^2}{2}} $$

Where

$I_j$: indicator random variable: $1 \leq j \leq k$

is the $j$-th run incorrect?

$$\forall j \ E[I_j] = \frac{1}{3} \rightarrow E[\sum I_j] = \sum E[I_j] = \frac{2k}{3}$$