Binomial distribution

$X_n = \text{numbers of heads when flipping a coin } n \text{ times.}$

**Claim**

$$\Pr[X_n = i] = \binom{n}{i} 2^{-n}.$$  

Where:  
$$\binom{n}{k} = \frac{n!}{(n-k)!k!}.$$  

Indeed, $\binom{n}{i}$ is the number of ways to choose $i$ elements out of $n$ elements (i.e., pick which $i$ coin flip come up heads).

Each specific such possibility (say 010010...) had probability $1/2^n$.  

---

### Part II

**Understanding the binomial distribution**
Massive randomness.. Is not that random.

Consider flipping a fair coin \( n \) times independently, head given 1, tail gives zero. How many heads? ...we get a binomial distribution.

This is known as **concentration of mass**. This is a very special case of the **law of large numbers**.

**Side note...**

**Law of large numbers (weakest form)...**

**Informal statement of law of large numbers**

For \( n \) large enough, the middle portion of the binomial distribution looks like (converges to) the normal/Gaussian distribution.

**Intuitive conclusion**

Randomized algorithm are unpredictable in the tactical level, but very predictable in the strategic level.
Part III
QuickSort with high probability

Show that QuickSort running time is $O(n \log n)$

- Probability of failure is
  $\alpha = Pr[C_{QS} \geq 32n \ln n] \leq Pr[\bigcup_i E_i] \leq \sum_{i=1}^{n} Pr[E_i]$.

- **Union bound**: for any two events $A$ and $B$:
  $Pr[A \cup B] \leq Pr[A] + Pr[B]$.
  Assume: $Pr[E_i] \leq 1/n^3$.

- Bad probability... $\alpha \leq \sum_{i=1}^{n} Pr[E_i] \leq \sum_{i=1}^{n} 1/n^3 = 1/n^2$.

- $\implies$ QuickSort performs $\leq 32n \ln n$ comparisons, w.h.p.

- $\implies$ QuickSort runs in $O(n \log n)$ time, with high probability.

Proving that an element...

- ... participates in small number of rounds.

- $n$: number of elements in input for QuickSort.

- $x$: Arbitrary element $x$ in input.

- $S_1$: Input.

- $S_i$: input to $i$th level recursive call that include $x$.

- $x$ lucky in $j$th iteration, if balanced split...

  $|S_{j+1}| \leq (3/4) |S_j|$ and $|S_j \setminus S_{j+1}| \leq (3/4) |S_j|$

- $Y_j = 1 \iff x$ lucky in $j$th iteration.

- $Pr[Y_j] = 1/2$.

- **Observation**: $Y_1, Y_2, \ldots, Y_m$ are independent variables.

- $x$ can participate $\leq \rho = \log_{4/3} n \leq 3.5 \ln n$ rounds.

- ...since $|S_j| \leq n(3/4)^{\rho} \not\in$ of lucky iteration in $1 \ldots j$.

- If $\rho$ lucky rounds in first $k$ rounds $\implies |S_k| \leq (3/4)^{\rho} n \leq 1$. 

Show that QuickSort running time is $O(n \log n)$

- QuickSort picks a pivot, splits into two subproblems, and continues recursively.

- Track single element in input.

- Game ends, when this element is alone in subproblem.

- Show every element in input, participates $\leq 32 \ln n$ rounds (with high enough probability).

- $E_i$: event $i$th element participates $> 32 \ln n$ rounds.

- $C_{QS}$: number of comparisons performed by QuickSort.

- Running time $O(C_{QS})$.

- Probability of failure is $\alpha = Pr[C_{QS} \geq 32n \ln n] \leq Pr[\bigcup_i E_i] \leq \sum_{i=1}^{n} Pr[E_i]$.

- $\implies$ by the union bound.
Proving that an element...  
... participates in small number of rounds.

- Brain reset!
- Q: How many rounds \( x \) participates in = how many coin flips till one gets \( \rho \) heads?
- A: In expectation, \( 2\rho \) times.

Lemma

In \( M \) coin flips: \( \Pr[\# \text{ heads } \leq M/4] \leq \exp(-M/8) \).

- Set \( M = 32 \ln n \geq 8\rho \).
- \( \Pr[Y_j = 0] = \Pr[Y_j = 1] = 1/2 \).
- \( Y_1, Y_2, \ldots, Y_M \) are independent.
- \( \implies \) probability \( \leq \rho \leq M/4 \) ones in \( Y_1, \ldots, Y_M \) is
  \[ \leq \exp\left(-\frac{M}{8}\right) \leq \exp(-\rho) \leq \frac{1}{n^3}. \]
- \( \implies \) probability \( x \) participates in \( M \) recursive calls of \( \text{QuickSort} \) \( \leq 1/n^3 \).

Proving that an element...  
... participates in small number of rounds.

- Assume the following:

Theorem

With high probability (i.e., \( 1 - 1/n^2 \)) the depth of the recursion of \( \text{QuickSort} \) is \( \leq 32 \ln n \). Thus, with high probability, the running time of \( \text{QuickSort} \) is \( O(n \log n) \).

- Same result holds for \( \text{MatchNutsAndBolts} \).

Proving that an element...  
... participants in small number of rounds.

\( n \) input elements. Probability depth of recursion in \( \text{QuickSort} \) \( > 32 \ln n \) is \( \leq (1/n^2) \times n = 1/n^2 \).

Result:

Alternative proof of high probability of \( \text{QuickSort} \)

- \( T \): \( n \) items to be sorted.
- \( t \in T \): element.
- \( X_i \): the size of subproblem in \( i \)th level of recursion containing \( t \).
- \( X_0 = n \), and \( \mathbb{E}[X_i | X_{i-1}] \leq \frac{13}{8} X_{i-1} + \frac{1}{2} X_{i-1} \leq \frac{7}{8} X_{i-1} \).
- \( \forall \) random variables \( \mathbb{E}[X] = \mathbb{E}_y[\mathbb{E}[X | Y = y]] \).
- \( \mathbb{E}[X_i] = \mathbb{E}_y[\mathbb{E}[X_i | X_{i-1} = y]] \leq \mathbb{E}_{X_{i-1}=y}[\frac{7}{8} y] = \frac{7}{8} \mathbb{E}[X_{i-1}] \leq \left(\frac{7}{8}\right)^i \mathbb{E}[X_0] = \left(\frac{7}{8}\right)^n n. \)
Alternative proof of high probability of *QuickSort*

- $M = 8 \log_{8/7} n$: $\mu = \mathbb{E}[X_M] \leq \left(\frac{7}{8}\right)^M n \leq \frac{1}{n^7} n = \frac{1}{n^7}$.

- **Markov’s Inequality:** For a non-negative variable $X$, and $t > 0$, we have:
  $$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$ 

- By Markov’s inequality:
  $$\Pr[t \text{ participates} > M \text{ recursive calls}] \leq \Pr[X_M \geq 1] \leq \frac{\mathbb{E}[X_M]}{1} \leq \frac{1}{n^7}.$$ 

- Probability any element of input participates $> M$ recursive calls $\leq n(1/n^7) \leq 1/n^6$.

---

**Preliminaries**

- **$X$, $Y$:** Random variables are *independent* if $\forall x, y$:
  $$\Pr[(X = x) \cap (Y = y)] = \Pr[X = x] \cdot \Pr[Y = y].$$

- The following is easy to prove:

**Claim**

*If $X$ and $Y$ are independent

- $\implies \mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.
- $\implies Z = e^X$ and $W = e^Y$ are independent.*
Proof of Chernoff inequality

Fix arbitrary \( t > 0 \):

\[
\Pr[Y \geq \Delta] = \Pr[tY \geq t\Delta] = \Pr[\exp(tY) \geq \exp(t\Delta)] \leq \frac{E[\exp(tY)]}{\exp(t\Delta)},
\]

Chernoff inequality...

...what it really says

By theorem:

\[
\Pr[Y \geq \Delta] = \sum_{i=\Delta}^{n} \Pr[Y = i] = \sum_{i=n/2+\Delta/2}^{n} \left(\begin{array}{c} n \\ i \end{array}\right) \leq \exp\left(-\frac{\Delta^2}{2n}\right),
\]

Proof of Chernoff inequality

Continued...

\[
E[\exp(tX_i)] = \frac{1}{2}e^t + \frac{1}{2}e^{-t} = \frac{e^t + e^{-t}}{2}
\]
\[
= \frac{1}{2}\left(1 + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots\right) + \frac{1}{2}\left(1 - \frac{t}{1!} + \frac{t^2}{2!} - \frac{t^3}{3!} + \cdots\right)
\]
\[
= 1 + \frac{t^2}{2!} + + \cdots + \frac{t^{2k}}{(2k)!} + \cdots.
\]

However: \( (2k)! = k!(k + 1)(k + 2) \cdots 2k \geq k!2^k \).

\[
E[\exp(tY)] = E\left[\exp\left(\sum_{i=1}^{n} tX_i\right)\right] = E\left[\prod_{i} \exp(tX_i)\right] = \prod_{i} E[\exp(tX_i)] \leq \exp\left(\frac{t^2}{2}\right).
\]

Corollary

Let \( X_1, \ldots, X_n \) be \( n \) independent random variables, such that \( \Pr[X_i = 1] = \Pr[X_i = -1] = \frac{1}{2} \), for \( i = 1, \ldots, n \). Let \( Y = \sum_{i=1}^{n} X_i \). Then, for any \( \Delta > 0 \), we have

\[
\Pr[|Y| \geq \Delta] \leq 2\exp\left(-\frac{\Delta^2}{2n}\right).
\]
Chernoff inequality for coin flips

Let $X_1, \ldots, X_n$ be $n$ independent coin flips, such that $\Pr[X_i = 1] = \Pr[X_i = 0] = \frac{1}{2}$, for $i = 1, \ldots, n$. Let $Y = \sum_{i=1}^n X_i$. Then, for any $\Delta > 0$, we have

$$\Pr\left[\frac{n}{2} - Y \geq \Delta\right] \leq \exp\left(-\frac{2\Delta^2}{n}\right)$$

and

$$\Pr\left[Y - \frac{n}{2} \geq \Delta\right] \leq \exp\left(-\frac{2\Delta^2}{n}\right).$$

In particular, we have

$$\Pr\left[Y - \frac{n}{2} \geq \Delta\right] \leq 2 \exp\left(-\frac{2\Delta^2}{n}\right).$$

The special case we needed

**Lemma**

In a sequence of $M$ coin flips, the probability that the number of ones is smaller than $L \leq M/4$ is at most $\exp(-M/8)$.

**Proof.**

Let $Y = \sum_{i=1}^M X_i$ the sum of the $M$ coin flips. By the above corollary, we have:

$$\Pr[Y \leq L] = \Pr\left[\frac{M}{2} - Y \geq \frac{M}{2} - L\right] = \Pr\left[\frac{M}{2} - Y \geq \Delta\right],$$

where $\Delta = M/2 - L \geq M/4$. Using the above Chernoff inequality, we get $\Pr[Y \leq L] \leq \exp(-\frac{2\Delta^2}{M}) \leq \exp(-M/8)$. □

The Chernoff Bound

**The general problem**

Let $X_1, \ldots, X_n$ be $n$ independent Bernoulli trials, where $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$, and let denote

$$Y = \sum_{i} X_i \quad \mu = \mathbb{E}[Y].$$

**Question:** what is the probability that $Y \geq (1 + \delta)\mu$.
The Chernoff Bound
The general case

Theorem (Chernoff inequality)

For any $\delta > 0$,

$$\Pr[Y > (1 + \delta)\mu] < \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}}\right)^\mu.$$  

Or in a more simplified form, for any $\delta \leq 2e - 1$,

$$\Pr[Y > (1 + \delta)\mu] < \exp\left(-\mu\delta^2/4\right),$$

and

$$\Pr[Y > (1 + \delta)\mu] < 2^{-\mu(1+\delta)},$$

for $\delta \geq 2e - 1$.

---

Part VI
Treaps

Balanced binary search trees...

- Work usually by storing additional information.
- Idea: For every element $x$ inserted randomly choose priority $p(x) \in [0, 1]$.
- $X = \{x_1, \ldots, x_n\}$ priorities: $p(x_1), \ldots, p(x_n)$.
- $x_k$: lowest priority in $X$.
- Make $x_k$ the root.
- partition $X$ in the natural way:
  (A) $L$: set of all the numbers smaller than $x_k$ in $X$, and
  (B) $R$: set of all the numbers larger than $x_k$ in $X$.  

---
Continuing recursively, we have:

(A) $L$: set of all the numbers smaller than $x_k$ in $X$, and
(B) $R$: set of all the numbers larger than $x_k$ in $X$.

**Definition**

Resulting tree a *treap*. Tree over the elements, and a heap over the priorities; that is, $TREAP = TREE + HEAP$.

**Treaps - implementation**

**Observation**

*Given* $n$ distinct elements, and their (distinct) priorities, the treap storing them is uniquely defined.

**Treaps continued**

**Lemma**

$S$: $n$ elements.

*Expected depth of treap $T$ for $S$ is $O(\log(n))$.*

*Depth of treap $T$ for $S$ is $O(\log(n))$ w.h.p.*

**Proof.**

QuickSort...
**Treaps – insertion**
- \( x \): an element to insert.
- Insert it into \( T \) as a regular binary tree.
- Takes \( O(\text{height}(T)) \).
- \( x \) is a leaf in the treap.
- Pick priority \( p(x) \in [0, 1] \).
- Valid search tree... but priority heap is broken at \( x \).
- Fix priority heap around \( x \).

**Fix treap for a leaf \( x \)...**

```plaintext
RotateUp(x)
y ← parent(x)
while \( p(y) > p(x) \) do
  if \( y \).left child = x then
    RotateRight(y)
  else
    RotateLeft(y)
y ← parent(x)
```

Insertion takes \( O(\text{height}(T)) \).

**Treaps – deletion**
- Deletion is just an insertion done in reverse.
- \( x \): element to delete.
- Set \( p(x) \leftarrow +\infty \).
- rotate \( x \) down till its a leaf.
- Rotate so that child with lower priority becomes new parent.
- \( x \) is now leaf – deleting is easy...

**Split**
- \( x \): element stored in treap \( T \).
- split \( T \) into two treaps – one treap \( T_{\leq x} \) and treap \( T_{> x} \) for all the elements larger than \( x \).
- Set \( p(x) \leftarrow -\infty \),
- fix priorities by rotation.
- \( x \) item is now the root.
- Splitting is now easy....
- Restore \( x \) to its original priority. Fix by rotations.
Meld

- $T_L$ and $T_R$: treaps.
- all elements in $T_L$ | all elements in $T_R$.
- Want to merge them into a single treap...

Treap – summary

**Theorem**

Let $T$ be an empty treap, after a sequence of $m = n^c$ insertions, where $c$ is some constant.

$d$: arbitrary constant.

The probability depth $T$ ever exceed $d \log n$ is $\leq 1/n^{O(1)}$.

A treap can handle insertion/deletion in $O(\log n)$ time with high probability.

Proof

Proof.

- $T_1, \ldots, T_m$: sequence of treaps.
- $T_i$ is treap after $i$th operation.
- $\alpha_i = \Pr[\text{depth}(T_i) > tc' \log n] = \Pr[\text{depth}(T_i) > c't \left( \frac{\log n}{\log |T_i|} \right) \cdot \log |T_i|] \leq \frac{1}{n^{d \log n}}$.
- Use union bound...

Bibliographical Notes

- Chernoff inequality was a rediscovery of Bernstein inequality.
- ...published in 1924 by Sergei Bernstein.
- Treaps were invented by Siedel and Aragon Seidel and Aragon [1996].
- Experimental evidence suggests that Treaps performs reasonably well in practice see Cho and Sahni [2000].
- Old implementation of treaps I wrote in C is available here: http://valis.cs.uiuc.edu/blog/?p=6060.