Chapter 13
Randomized Algorithms

Randomization

Algorithmic design patterns.
- Greedy.
- Divide-and-conquer.
- Dynamic programming.
- Network flow.
- Randomization.

Randomization. Allow fair coin flip in unit time.

Why randomize? Can lead to simplest, fastest, or only known algorithm for a particular problem.

Ex. Symmetry breaking protocols, graph algorithms, quicksort, hashing, load balancing, Monte Carlo integration, cryptography.

13.1 Contention Resolution

Contestation Resolution in a Distributed System

Contention resolution. Given n processes $P_1, \ldots, P_n$, each competing for access to a shared database. If two or more processes access the database simultaneously, all processes are locked out. Devise protocol to ensure all processes get through on a regular basis.

Restriction. Processes can’t communicate.

Challenge. Need symmetry-breaking paradigm.
**Protocol.** Each process requests access to the database at time \( t \) with probability \( p = \frac{1}{n} \).

**Claim.** Let \( S[i, t] = \) event that process \( i \) succeeds in accessing the database at time \( t \). Then \( \frac{1}{e \cdot n} \leq \Pr[S(i, t)] \leq \frac{1}{2n} \).

**Pf.** By independence, \( \Pr[S(i, t)] = p \cdot (1-p)^{n-1} \).

- Setting \( p = \frac{1}{n} \), we have \( \Pr[S(i, t)] = \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \).

**Useful facts from calculus.** As \( n \) increases from 2, the function:
  - \( (1 - 1/n)^n \) converges monotonically from \( 1/4 \) up to \( 1/e \)
  - \( (1 - 1/n)^{n-1} \) converges monotonically from \( 1/2 \) down to \( 1/e \).

**Claim.** The probability that process \( i \) fails to access the database in \( e \cdot n \) rounds is at most \( 1/e \). After \( e \cdot n(c \ln n) \) rounds, the probability is at most \( n^{-c} \).

**Pf.** Let \( F[i, t] = \) event that process \( i \) fails to access database in rounds \( 1 \) through \( t \). By independence and previous claim, we have \( \Pr[F(i, t)] \leq (1 - \frac{1}{en})^t \).

- Choose \( t = \lceil e \cdot n \rceil \): \( \Pr[F(i, t)] \leq (1 - \frac{1}{en})^\lceil en \rceil \leq \frac{1}{e} \)
- Choose \( t = \lceil e \cdot n \rceil \cdot \lceil c \ln n \rceil \): \( \Pr[F(i, t)] \leq \left(\frac{1}{e}\right)^{c \ln n} = n^{-c} \)

**Claim.** The probability that all processes succeed within \( 2e \cdot n \ln n \) rounds is at least \( 1 - 1/n \).

**Pf.** Let \( F[t] = \) event that at least one of the \( n \) processes fails to access database in any of the rounds \( 1 \) through \( t \).

\[
\Pr[F[t]] = \Pr\left[ \bigcup_{i=1}^{n} F[i, t] \right] \leq \sum_{i=1}^{n} \Pr[F[i, t]] \leq n \left(1 - \frac{1}{en}\right)^t
\]

- Choosing \( t = 2 \lceil \ln n \rceil \lceil c \ln n \rceil \) yields \( \Pr[F[t]] \leq n \cdot n^2 = 1/n \).

**Union bound.** Given events \( E_1, \ldots, E_n \), \( \Pr\left[ \bigcup_{i=1}^{n} E_i \right] \leq \sum_{i=1}^{n} \Pr[E_i] \)
Global Minimum Cut

**Global min cut.** Given a connected, undirected graph $G = (V, E)$ find a cut $(A, B)$ of minimum cardinality.

**Applications.** Partitioning items in a database, identify clusters of related documents, network reliability, network design, circuit design, TSP solvers.

**Network flow solution.**
- Replace every edge $(u, v)$ with two antiparallel edges $(u, v)$ and $(v, u)$.
- Pick some vertex $s$ and compute min $s$-$v$ cut separating $s$ from each other vertex $v \in V$.

**False intuition.** Global min-cut is harder than min $s$-$t$ cut.

**Contraction Algorithm**

**Claim.** The contraction algorithm returns a min cut with prob $\geq 2/n^2$.

**Pf.** Consider a global min-cut $(A^*, B^*)$ of $G$. Let $F^*$ be edges with one endpoint in $A^*$ and the other in $B^*$. Let $k = |F^*|$ = size of min cut.
- In first step, algorithm contracts an edge in $F^*$ probability $k / |E|$.
- Every node has degree $\geq k$ since otherwise $(A^*, B^*)$ would not be min-cut. $\Rightarrow |E| \geq \frac{1}{2}kn$.
- Thus, algorithm contracts an edge in $F^*$ with probability $\leq 2/n$.

$$\Pr\left[ E_1 \cap E_2 \cdots \cap E_{n-2} \right] = \Pr[E_1] \times \Pr[E_2 | E_1] \times \cdots \times \Pr[E_{n-2} | E_1 \cap E_2 \cdots \cap E_{n-3}]$$
$$\geq \left(1 - \frac{1}{2}\right) \left(1 - \frac{1}{2}\right) \cdots \left(1 - \frac{1}{2}\right)$$
$$= \left(\frac{n-2}{2}\right) \left(\frac{n-4}{2}\right) \cdots \left(\frac{1}{2}\right) \left(\frac{1}{2}\right)$$
$$= \frac{2}{n(n-1)} \cdots \left(\frac{1}{2}\right) \left(\frac{1}{2}\right)$$
$$\geq \frac{2}{n^2}$$
Contraction Algorithm

**Amplification.** To amplify the probability of success, run the contraction algorithm many times.

**Claim.** If we repeat the contraction algorithm \( n^2 \ln n \) times with independent random choices, the probability of failing to find the global min-cut is at most \( 1/n^2 \).

**Pf.** By independence, the probability of failure is at most

\[
\left(1 - \frac{2}{n^2}\right)^{n^2 \ln n} \leq \left(e^{-2}\right)^{2 \ln n} = \frac{1}{n^2}
\]

\((1 - 1/x)^x \leq 1/e\)

Global Min Cut: Context

**Remark.** Overall running time is slow since we perform \( \Theta(n^2 \log n) \) iterations and each takes \( \Omega(m) \) time.

**Improvement.** [Karger-Stein 1996] \( O(n^2 \log^3 n) \).
- Early iterations are less risky than later ones: probability of contracting an edge in min cut hits 50% when \( n / \sqrt{2} \) nodes remain.
- Run contraction algorithm until \( n / \sqrt{2} \) nodes remain.
- Run contraction algorithm twice on resulting graph, and return best of two cuts.

**Extensions.** Naturally generalizes to handle positive weights.

**Best known.** [Karger 2000] \( O(m \log^3 n) \).

13.3 Linearity of Expectation

**Expectation.** Given a discrete random variables \( X \), its expectation \( E[X] \) is defined by:

\[
E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j]
\]

**Waiting for a first success.** Coin is heads with probability \( p \) and tails with probability \( 1-p \). How many independent flips \( X \) until first heads?

\[
E[X] = \sum_{j=0}^{\infty} j \cdot \Pr[X = j] = \sum_{j=0}^{\infty} j \cdot (1-p)^{j-1} p = \frac{p}{1-p} \sum_{j=0}^{\infty} (1-p)^{j} = \frac{p}{1-p} \cdot \frac{1-p}{p} = \frac{1}{p}
\]
Expectation: Two Properties

**Useful property.** If $X$ is a 0/1 random variable, $E[X] = Pr[X = 1]$.

**Pf.**

$$ E[X] = \sum_{j=0}^{1} j \cdot Pr[X = j] = \sum_{j=0}^{1} j \cdot Pr[X = j] = Pr[X = 1] $$

**Linearity of expectation.** Given two random variables $X$ and $Y$ defined over the same probability space, $E[X + Y] = E[X] + E[Y]$.

Decouples a complex calculation into simpler pieces.

---

**Guessing Cards**

**Game.** Shuffle a deck of $n$ cards; turn them over one at a time; try to guess each card.

**Memoryless guessing.** No psychic abilities; can’t even remember what’s been turned over already. Guess a card from full deck uniformly at random.

**Claim.** The expected number of correct guesses is 1.

**Pf.** (surprisingly effortless using linearity of expectation)

- Let $X_i = 1$ if $i$th prediction is correct and 0 otherwise.
- Let $X = \text{number of correct guesses} = X_1 + ... + X_n$.
- $E[X_i] = Pr[X_i = 1] = 1/n$.
- $E[X] = E[X_1] + ... + E[X_n] = 1/n + ... + 1/n = 1$.

**Claim.** The expected number of steps is $\Theta(n \log n)$.

**Pf.**

- Phase $j = \text{time between } j \text{ and } j+1 \text{ distinct coupons}$. 
- Let $X_j = \text{number of steps you spend in phase } j$.
- Let $X = \text{number of steps in total} = X_0 + X_1 + ... + X_n$.

$$ E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = n H(n) $$

$\Rightarrow$ expected waiting time $= n/(n-j)$

---

**Coupon Collector**

**Coupon collector.** Each box of cereal contains a coupon. There are $n$ different types of coupons. Assuming all boxes are equally likely to contain each coupon, how many boxes before you have $\geq 1$ coupon of each type?

**Claim.** The expected number of steps is $\Theta(n \log n)$.

**Pf.**

- Phase $j = \text{time between } j \text{ and } j+1 \text{ distinct coupons}$. 
- Let $X_j = \text{number of steps you spend in phase } j$.
- Let $X = \text{number of steps in total} = X_0 + X_1 + ... + X_n$.

$$ E[X] = \sum_{j=0}^{n-1} E[X_j] = \sum_{j=0}^{n-1} \frac{n}{n-j} = n \sum_{i=1}^{n} \frac{1}{i} = n H(n) $$

$\Rightarrow$ expected waiting time $= n/(n-j)$
13.4 MAX 3-SAT

Maximum 3-Satisfiability

MAX-3SAT. Given 3-SAT formula, find a truth assignment that satisfies as many clauses as possible.

\[ C_1 = x_2 \lor \overline{x_3} \lor \overline{x_4} \]
\[ C_2 = x_2 \lor x_3 \lor \overline{x_4} \]
\[ C_3 = \overline{x_1} \lor x_2 \lor x_4 \]
\[ C_4 = \overline{x_1} \lor \overline{x_2} \lor x_3 \]
\[ C_5 = x_1 \lor x_2 \lor x_4 \]

Remark. NP-hard search problem.

Simple idea. Flip a coin, and set each variable true with probability \( \frac{1}{2} \), independently for each variable.

Claim. Given a 3-SAT formula with \( k \) clauses, the expected number of clauses satisfied by a random assignment is \( \frac{7k}{8} \).

Pf. Consider random variable \( Z_j = \begin{cases} 1 & \text{if clause } C_j \text{ is satisfied} \\ 0 & \text{otherwise.} \end{cases} \)

- Let \( Z = \text{weight of clauses satisfied by assignment } Z_j \).

\[
E[Z] = \sum_{j=1}^{k} E[Z_j]
\]

\[
= \sum_{j=1}^{k} \Pr[\text{clause } C_j \text{ is satisfied}]
\]

\[
= \frac{1}{2} k
\]

Corollary. For any instance of 3-SAT, there exists a truth assignment that satisfies at least a \( \frac{7}{8} \) fraction of all clauses.

Pf. Random variable is at least its expectation some of the time.

The Probabilistic Method

Corollary. For any instance of 3-SAT, there exists a truth assignment that satisfies at least a \( \frac{7}{8} \) fraction of all clauses.

Pf. Random variable is at least its expectation some of the time.

Probabilistic method. We showed the existence of a non-obvious property of 3-SAT by showing that a random construction produces it with positive probability!
Q. Can we turn this idea into a $7/8$-approximation algorithm? In general, a random variable can almost always be below its mean.

**Lemma.** The probability that a random assignment satisfies $\geq 7k/8$ clauses is at least $1/(8k)$.

**Pf.** Let $p_j$ be probability that exactly $j$ clauses are satisfied; let $p$ be probability that $\geq 7k/8$ clauses are satisfied.

\[
\frac{7k}{8} = E[Z] = \sum_{j=0}^{\infty} j p_j
\]

\[
= \sum_{j \leq 7k/8} j p_j + \sum_{j > 7k/8} j p_j
\]

\[
\leq \left(\frac{7k}{8} - \frac{1}{8}\right) \sum_{j \leq 7k/8} p_j + k \sum_{j > 7k/8} p_j
\]

\[
\leq \left(\frac{7k}{8} - \frac{1}{8}\right) \cdot 1 + k p
\]

Rearranging terms yields $p \geq 1/(8k)$.

**Johnson’s algorithm.** Repeatedly generate random truth assignments until one of them satisfies $\geq 7k/8$ clauses.

**Theorem.** Johnson’s algorithm is a $7/8$-approximation algorithm.

**Pf.** By previous lemma, each iteration succeeds with probability at least $1/(8k)$. By the waiting-time bound, the expected number of trials to find the satisfying assignment is at most $8k$.

**Maximum Satisfiability**

**Extensions.**
- Allow one, two, or more literals per clause.
- Find max weighted set of satisfied clauses.

**Theorem.** [Asano-Williamson 2000] There exists a $0.784$-approximation algorithm for MAX-SAT.

**Theorem.** [Karloff-Zwick 1997, Zwick+computer 2002] There exists a $7/8$-approximation algorithm for version of MAX-3SAT where each clause has at most 3 literals.

**Theorem.** [Håstad 1997] Unless $P = NP$, no $\rho$-approximation algorithm for MAX-3SAT (and hence MAX-SAT) for any $\rho > 7/8$ is very unlikely to improve over simple randomized algorithm for MAX-3SAT.

**Monte Carlo vs. Las Vegas Algorithms**

**Monte Carlo algorithm.** Guaranteed to run in poly-time, likely to find correct answer.

**Ex:** Contraction algorithm for global min cut.

**Las Vegas algorithm.** Guaranteed to find correct answer, likely to run in poly-time.

**Ex:** Randomized quicksort, Johnson’s MAX-3SAT algorithm.

**Remark.** Can always convert a Las Vegas algorithm into Monte Carlo, but no known method to convert the other way.

Stop algorithm after a certain point
RP and ZPP

RP. [Monte Carlo] Decision problems solvable with one-sided error in poly-time.

One-sided error.
- If the correct answer is no, always return no.
- If the correct answer is yes, return yes with probability $\geq \frac{1}{2}$.

ZPP. [Las Vegas] Decision problems solvable in expected poly-time.

Theorem. $P \subseteq ZPP \subseteq RP \subseteq NP$.

Fundamental open questions. To what extent does randomization help?
Does $P = ZPP$? Does $ZPP = RP$? Does $RP = NP$?

13.6 Universal Hashing

Hashing. Create an array $H$ of size $n$. When processing element $u$, access array element $H[h(u)]$.

Collision. When $h(u) = h(v)$ but $u \neq v$.
- A collision is expected after $\Theta(\sqrt{n})$ random insertions. This phenomenon is known as the "birthday paradox."
- Separate chaining: $H[i]$ stores linked list of elements $u$ with $h(u) = i$.

Dictionary Data Type

Dictionary. Given a universe $U$ of possible elements, maintain a subset $S \subseteq U$ so that inserting, deleting, and searching in $S$ is efficient.

Dictionary interface.
- Create(): Initialize a dictionary with $S = \phi$.
- Insert(u): Add element $u \in U$ to $S$.
- Delete(u): Delete $u$ from $S$, if $u$ is currently in $S$.
- Lookup(u): Determine whether $u$ is in $S$.

Challenge. Universe $U$ can be extremely large so defining an array of size $|U|$ is infeasible.

Applications. File systems, databases, Google, compilers, checksums P2P networks, associative arrays, cryptography, web caching, etc.
Ad Hoc Hash Function

Ad hoc hash function.

```
int h(String s, int n) {
    int hash = 0;
    for (int i = 0; i < s.length(); i++)
        hash = (31 * hash) + s[i];
    return hash % n;
}
```

Deterministic hashing. If |U| \geq n^2, then for any fixed hash function h, there is a subset S \subseteq U of n elements that all hash to same slot. Thus, \Theta(n) time per search in worst-case.

Q. But isn’t ad hoc hash function good enough in practice?

Algorithmic Complexity Attacks

When can’t we live with ad hoc hash function?
- Obvious situations: aircraft control, nuclear reactors.
- Surprising situations: denial-of-service attacks.

Real world exploits. [Crosby-Wallach 2003]
- Bro server: send carefully chosen packets to DOS the server, using less bandwidth than a dial-up modem
- Perl 5.8.0: insert carefully chosen strings into associative array.
- Linux 2.4.20 kernel: save files with carefully chosen names.

Hashing Performance

Idealistic hash function. Maps m elements uniformly at random to n hash slots.
- Running time depends on length of chains.
- Average length of chain = \alpha = m / n.
- Choose n = m \Rightarrow on average O(1) per insert, lookup, or delete.

Challenge. Achieve idealized randomized guarantees, but with a hash function where you can easily find items where you put them.

Approach. Use randomization in the choice of h.

Universal Hashing

Universal class of hash functions. [Carter-Wegman 1980s]
- For any pair of elements u, v \in U, Pr_{h \in H}[h(u) = h(v)] \leq 1/n
- Can select random h efficiently.
- Can compute h(u) efficiently.

Ex. U = \{ a, b, c, d, e, f \}, n = 2.
Universal Hashing

**Universal hashing property.** Let $H$ be a universal class of hash functions; let $h \in H$ be chosen uniformly at random from $H$; and let $u \in U$. For any subset $S \subseteq U$ of size at most $n$, the expected number of items in $S$ that collide with $u$ is at most 1.

**Pf.** For any element $s \in S$, define indicator random variable $X_s = 1$ if $h(s) = h(u)$ and 0 otherwise. Let $X$ be a random variable counting the total number of collisions with $u$.

$$E_{h \in H}[X] = E[\sum_{s \in S} X_s] = \sum_{s \in S} E[X_s] = \sum_{s \in S} \Pr[X_s = 1] \leq \sum_{s \in S} \frac{1}{n} = 1 \frac{1}{n} = 1$$

(linearity of expectation $X_s$ is a 0-1 random variable universal (assumes $u \notin S$)

Designing a Universal Class of Hash Functions

**Theorem.** $H = \{ h_a : a \in A \}$ is a universal class of hash functions.

**Pf.** Let $x = (x_1, x_2, \ldots, x_r)$ and $y = (y_1, y_2, \ldots, y_r)$ be two distinct elements of $U$. We need to show that $\Pr[h_a(x) = h_a(y)] \leq 1/n$.

- Since $x \neq y$, there exists an integer $j$ such that $x_j \neq y_j$.
- We have $h_a(x) = h_a(y)$ iff
  $$a_j (y_j - x_j) \equiv \sum_{j \neq i} a_i (x_i - y_i) \mod p$$

- Can assume $a$ was chosen uniformly at random by first selecting all coordinates $a_i$ where $i \neq j$, then selecting $a_j$ at random. Thus, we can assume $a_j$ is fixed for all coordinates $i \neq j$.

- Since $p$ is prime, $a_j z \equiv m \mod p$ has at most one solution among $p$ possibilities. \( \text{--- see lemma on next slide} \)

- Thus $\Pr[h_a(x) = h_a(y)] = 1/p \leq 1/n$. \( \text{---} \)

Number Theory Facts

**Fact.** Let $p$ be prime, and let $z \equiv 0 \mod p$. Then $\alpha z \equiv m \mod p$ has at most one solution $0 \leq \alpha < p$.

**Pf.**
- Suppose $\alpha$ and $\beta$ are two different solutions.
- Then $(\alpha - \beta)z \equiv 0 \mod p$; hence $(\alpha - \beta)z$ is divisible by $p$.
- Since $z \equiv 0 \mod p$, we know that $z$ is not divisible by $p$; it follows that $(\alpha - \beta)$ is divisible by $p$.
- This implies $\alpha = \beta$. \( \text{---} \)

**Bonus fact.** Can replace "at most one" with "exactly one" in above fact.

**Pf idea.** Euclid’s algorithm.
13.9 Chernoff Bounds

**Theorem.** Suppose $X_1, \ldots, X_n$ are independent 0-1 random variables. Let $X = X_1 + \ldots + X_n$. Then for any $\mu \geq \mathbb{E}[X]$ and for any $\delta > 0$, we have

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

**Pf.** We apply a number of simple transformations.

- For any $t > 0$,
  $$
  \Pr[X > (1 + \delta)\mu] = \Pr\left[ e^{tX} > e^{t(1 + \delta)\mu} \right] \leq e^{-t(1 + \delta)\mu} \cdot \mathbb{E}[e^{tX}]
  $$
  Function $f(x) = e^{tx}$ is monotone in $x$.

- Now
  $$
  \mathbb{E}[e^{tX}] = \mathbb{E}\left[ e^{t \sum_i X_i} \right] = \prod_i \mathbb{E}[e^{tX_i}]
  $$
  Definition of $X$, independence.

**Chernoff Bounds (above mean)**

**Pf.** (cont)

- Let $p_i = \Pr[X_i = 1]$. Then,
  $$
  \mathbb{E}[e^{tX}] = p_i e^t + (1 - p_i)e^0 = 1 + p_i(e^t - 1) \leq e^{p_i(e^t - 1)}
  $$
  for any $\alpha \geq 0$, $1 + \alpha = e^{\alpha}$

- Combining everything:

  $$
  \Pr[X > (1 + \delta)\mu] \leq e^{-t(1 + \delta)\mu} \prod_i \mathbb{E}[e^{tX_i}] \leq e^{-t(1 + \delta)\mu} \prod_i e^{p_i(e^t - 1)} \leq e^{-t(1 + \delta)\mu} e^{\mu(e^t - 1)}
  $$

- Finally, choose $t = \ln(1 + \delta)$.

**Remark.** Not quite symmetric since only makes sense to consider $\delta < 1$.

**Theorem.** Suppose $X_1, \ldots, X_n$ are independent 0-1 random variables. Let $X = X_1 + \ldots + X_n$. Then for any $\mu \leq \mathbb{E}[X]$ and for any $0 < \delta < 1$, we have

$$
\Pr[X < (1 - \delta)\mu] < e^{-\delta^2 \mu / 2}
$$

**Pf idea.** Similar.

**Chernoff Bounds (below mean)**

**Remark.** Not quite symmetric since only makes sense to consider $\delta < 1$. 

**Chernoff Bounds (above mean)**

- Let $p_i = \Pr[X_I = 1]$. Then,
  $$
  \mathbb{E}[e^{tX}] = p_i e^t + (1 - p_i)e^0 = 1 + p_i(e^t - 1) \leq e^{p_i(e^t - 1)}
  $$
  for any $\alpha \geq 0$, $1 + \alpha = e^{\alpha}$

- Combining everything:

  $$
  \Pr[X > (1 + \delta)\mu] \leq e^{-t(1 + \delta)\mu} \prod_i \mathbb{E}[e^{tX_i}] \leq e^{-t(1 + \delta)\mu} \prod_i e^{p_i(e^t - 1)} \leq e^{-t(1 + \delta)\mu} e^{\mu(e^t - 1)}
  $$

- Finally, choose $t = \ln(1 + \delta)$.

**Remark.** Not quite symmetric since only makes sense to consider $\delta < 1$. 

**Chernoff Bounds (below mean)**

- Let $p_i = \Pr[X_I = 1]$. Then,
  $$
  \mathbb{E}[e^{tX}] = p_i e^t + (1 - p_i)e^0 = 1 + p_i(e^t - 1) \leq e^{p_i(e^t - 1)}
  $$
  for any $\alpha \geq 0$, $1 + \alpha = e^{\alpha}$

- Combining everything:

  $$
  \Pr[X > (1 + \delta)\mu] \leq e^{-t(1 + \delta)\mu} \prod_i \mathbb{E}[e^{tX_i}] \leq e^{-t(1 + \delta)\mu} \prod_i e^{p_i(e^t - 1)} \leq e^{-t(1 + \delta)\mu} e^{\mu(e^t - 1)}
  $$

- Finally, choose $t = \ln(1 + \delta)$.

**Remark.** Not quite symmetric since only makes sense to consider $\delta < 1$. 

**Chernoff Bounds (below mean)**

- Let $p_i = \Pr[X_I = 1]$. Then,
  $$
  \mathbb{E}[e^{tX}] = p_i e^t + (1 - p_i)e^0 = 1 + p_i(e^t - 1) \leq e^{p_i(e^t - 1)}
  $$
  for any $\alpha \geq 0$, $1 + \alpha = e^{\alpha}$

- Combining everything:

  $$
  \Pr[X > (1 + \delta)\mu] \leq e^{-t(1 + \delta)\mu} \prod_i \mathbb{E}[e^{tX_i}] \leq e^{-t(1 + \delta)\mu} \prod_i e^{p_i(e^t - 1)} \leq e^{-t(1 + \delta)\mu} e^{\mu(e^t - 1)}
  $$

- Finally, choose $t = \ln(1 + \delta)$.
13.10 Load Balancing

Load Balancing

System in which m jobs arrive in a stream and need to be processed immediately on n identical processors. Find an assignment that balances the workload across processors.

Centralized controller. Assign jobs in round-robin manner. Each processor receives at most \( \lceil m/n \rceil \) jobs.

Decentralized controller. Assign jobs to processors uniformly at random. How likely is it that some processor is assigned "too many" jobs?

Analysis.

- Let \( X_i \) = number of jobs assigned to processor \( i \).
- Let \( Y_{ij} = 1 \) if job \( j \) assigned to processor \( i \), and 0 otherwise.
- We have \( E[Y_{ij}] = 1/n \)
- Thus, \( X_i = \sum_j Y_{ij} \), and \( \mu = E[X_i] = 1 \).
- Applying Chernoff bounds with \( \delta = c - 1 \) yields
  \[
  \Pr[X_i > c] \leq \frac{e^{c-1}}{e^c} \leq \left( \frac{e}{c} \right)^c \leq \left( \frac{1}{\gamma(n)} \right)^2 \gamma(n) = \frac{1}{n^2}
  \]
- Let \( \gamma(n) \) be number \( x \) such that \( x^c = n \), and choose \( c = e \gamma(n) \).
  
  \[
  \Pr[X_i > e^c] \leq \left( \frac{e}{c} \right)^e \leq \left( \frac{1}{\gamma(n)} \right)^e \leq \left( \frac{1}{e} \right)^{\ln n} = \frac{1}{n^2}
  \]
- Union bound \( \Rightarrow \) with probability \( \geq 1 - 1/n \) no processor receives more than \( e \gamma(n) = \Theta(\log n / \log \log n) \) jobs.

Fact: this bound is asymptotically tight; with high probability, some processor receives \( \Theta(\log \log n) \) jobs.

Load Balancing: Many Jobs

Theorem. Suppose the number of jobs \( m = 16n \ln n \). Then on average, each of the \( n \) processors handles \( \mu = 16 \ln n \) jobs. With high probability every processor will have between half and twice the average load.

Pf.

- Let \( X_i, Y_{ij} \) be as before.
- Applying Chernoff bounds with \( \delta = 1 \) yields
  \[
  \Pr[X_i > 2\mu] \leq \left( \frac{e}{4} \right)^{16n \ln n} \leq \left( \frac{1}{e} \right)^{\ln n} = \frac{1}{n^2}
  \]
- Union bound \( \Rightarrow \) every processor has load between half and twice the average with probability \( \geq 1 - 2/n \).
### 13.5 Randomized Divide-and-Conquer

**Quicksort**

**Sorting.** Given a set of $n$ distinct elements $S$, rearrange them in ascending order.

```java
RandomizedQuicksort(S) {
    if |S| = 0 return
    choose a splitter $a_i \in S$ uniformly at random
    foreach (a \in S) {
        if (a < $a_i$) put a in $S^-$
        else if (a > $a_i$) put a in $S^+$
    }
    RandomizedQuicksort($S^-$)
    output $a_i$
    RandomizedQuicksort($S^+$)
}
```

**Remark.** Can implement in-place.

---

**Running time.**

- **[Best case.]** Select the median element as the splitter: quicksort makes $\Theta(n \log n)$ comparisons.
- **[Worst case.]** Select the smallest element as the splitter: quicksort makes $\Theta(n^2)$ comparisons.

**Randomize.** Protect against worst case by choosing splitter at random.

**Intuition.** If we always select an element that is bigger than 25% of the elements and smaller than 25% of the elements, then quicksort makes $\Theta(n \log n)$ comparisons.

**Notation.** Label elements so that $x_1 < x_2 < ... < x_n$. 

---

**O(log n) extra space**
**Quicksort: BST Representation of Splitters**

**BST representation.** Draw recursive BST of splitters.

- First splitter, chosen uniformly at random

**Observation.** Element only compared with its ancestors and descendants.
- $x_2$ and $x_7$ are compared if their LCA = $x_2$ or $x_7$.
- $x_2$ and $x_7$ are not compared if their LCA = $x_3$ or $x_4$ or $x_5$ or $x_6$.

**Claim.** $\Pr[x_i \text{ and } x_j \text{ are compared}] = \frac{2}{|j - i + 1|}$.

**Theorem.** Expected # of comparisons is $O(n \log n)$.

**Pf.**

\[
\sum_{1 \leq i < j \leq n} \frac{2}{j - i + 1} = 2 \sum_{j=1}^{n} \frac{1}{j} \leq 2n \sum_{j=1}^{n} \frac{1}{j} < 2n \int_{1}^{n} \frac{1}{x} \, dx = 2n \ln n
\]

- Probability that $i$ and $j$ are compared

**Theorem.** [Knuth 1973] Stddev of number of comparisons is $\sim 0.65N$.

**Ex.** If $n = 1$ million, the probability that randomized quicksort takes less than $4n \ln n$ comparisons is at least 99.94%.

**Chebyshev's inequality.** $\Pr[|X - \mu| \geq k\delta] \leq \frac{1}{k^2}$. 