***======== logic ========***

**conjunction**: P and Q; **disjunction**: P or Q; **negation**: not P; **implication**: P => Q logically equivalent to (not P) or Q; **contrapositive**: not Q => not P; **converse**: Q => P;

**Direct proof, proof by contraposition, proof by contradiction, proof by cases, induction, strong induction, well ordering principle**,

**Induction**: Base case + Induction hypothesis (for some k >= 1, assume …) + Induction step (by the induction hypothesis).

***======== number theory ========***

**gcd** = (x, y) -> y == 0 ? x : gcd(y, x mod y) (assume that x >= y >= 0 and x > 0)

def **extended-gcd** (x, y): return (x, 1, 0) if y == 0

(d, a, b) = extended-gcd (y, x mod y); return (d, b, (a - int(x / y) \* b))

**RSA**: Pick N = pq (p, q primes), e such that gcd(e, (p-1)(q-1)) = 1. **public key**: (N, e); **private key**: d. Property: (x^e)^d = 1 mod N

**Fermat's Little Theorem**: for any prime p and integer a in {1,2,…,p-1}, we have a^(p-1) = 1 mod p. => k^m = k^(m mod (p-1)) mod p

***======== stable marriage ========***

**nonexistence of stable match of roommates**

[A | B C D], [B | C A D], [C | A B D], [D | - - -]

{(A, B), (C, D)} => {(A, D), (B, C)} => {(A, B), (C, D)} => ...

**Stable Marriage Algorithm: Every Morning**: Each man goes to the first woman on his list not yet crossed off and proposes to her. **Every Afternoon**: Each woman says maybe to the man she likes best among the proposals (she now has him on a string) and never to all the rest. **Every Evening**: Each of the rejected suitors crosses off the woman from his list.

**Termination Lemma**: guaranteed to terminate in n^2 days.

**Improvement Lemma**: If W has M on a string on the kth day, then on every subsequent day she has someone on a string whom she likes at least as much as M.

**optimal woman:** highest woman on a man’s list whom he is paired with in any stable pairing.

**Male optimal pairing:** a stable pairing s.t. each man is paired with his optimal woman.

**Theorem**: The pairing output by the traditional propose & reject algorithm is male optimal.

**Theorem**: If a pairing is male optimal, then it is also female pessimal.

***======== error correction ========***

**Lagrange interpolation**: delta\_i (x) = prod\_{j!=i} (x-xj) / prod\_{j!=i} (xi-xj)

**Secret sharing**: at least k people => pick random polynomial with deg k-1 s.t. P(0) = secret. Give P(1) to 1st, ..., P(n) to nth.

**Erasure errors**: n packets, <=k packets lost => deg=n-1. send n+k packets. Prime q>=n+k

**General errors**: n packets, <=k packets lost => deg=n-1. send n+2k packets. Prime q>=n+2k

**Error locator polynomial** E(x) = (x-e1)…(x-ek); P(i)E(i)=R(i)E(i). Let P(x)E(x)=Q(x). Solve Q(x)=R(x)E(x). deg n+k-1 + deg k. But leading coefficient of E is 1.

Q(x) = a\_{n+k-1}x^{n+k-1} + … + a0; E(x) = x^k + b\_{k-1}x^{k-1} + … + b0

***======== graph theory ========***

**Euler's Theorem**: An undirected graph G = (V, E ) has an Eulerian tour if and only if the graph is connected (except possibly for isolated vertices) and every vertex has even degree. (for directed graph: for every vertex v, in-degree = out-degree)

**de Bruijn sequence**: a 2^n-bit **circular** sequence such that every string of length n occurs as a contiguous substring of the sequence exactly once.

**de Bruijn graph**: V = {0, 1}^(n-1), E = for each a1a2...an-1, (a1a2...an-1, a2a3...a{n-1}0), (a1a2...an-1, a2a3...a{n-1}1), (0a1a2...an-2, a1a2...a{n-1}), (1a1a2...an-2, a1a2...a{n-1})

**n-dimensional hypercube graph**: |E| = n2^(n-1). Let **E\_{S, V-S}** denote the set of edges connecting vertices in S to vertices in V-S. Then |E\_{S, V-S}| >= min(|S|, |V-S|) (prove by induction on n, consider S0, S1 both <= 2^(n-1)/2; or S0 > 2^(n-1)/2)

***======== counting ========***

**place k balls into n bins with replacement**, order does not matter: Represent each of the balls by a 0 and the separations between boxes by 1 => k \* 0 + (n-1) \* 1 => n+k-1 choose k

***======== probability ========***

**Union bound**: Pr[union] <= sum of Pr

**E(X)** = (definition) sum\_{a in A} a \* Pr[X=a] = sum\_{w in Omega} X(w) \* Pr(w)

**variance**: Var(X) = E((X - E(X)))^2 (definition) = E(X^2) - (E(X))^2

**standard deviation**: sigma(X) = sqrt(Var(X))

**covariance** of X, Y: E(XY) - E(X)E(Y)

**X,Y independent** => E(XY) = E(X)E(Y)

**X,Y independent** => Var(X+Y) = Var(X) + Var(Y)

**Theorem**: Let X be a random variable that takes on only non-negative integer values. Then E[X] = sum\_{i=1}^infinity Pr[X>=i].

***======== distribution ========***

**binomial distribution**: Pr[X=i] = (n choose i)p^i(1-p)^(n-i). X ~ Bin(n, p). **E[X]** = np. E(X^2) = n^2p^2 + np(1-p). **Var** = np(1-p).

**geometric distribution**: X ~ Geom(p). Pr[X=i] = (1-p)^(i-1)p. **E[X]** = 1/p. Pr[X>=i] = (1-p)^(i-1). **Var** = (1-p)/p^2.

**Poisson distribution**: X ~ Poiss(lambda). Pr[X=i] = lambda^i/i! \* e^(-lambda). **E[X]**=lambda. E(X^2)=lambda\*(lambda+1), **Var** = lambda.

**fixed points in random permutation: E(X)** = 1, E(X^2) = 2, **Var** = 1

**random walk**: **E(X)** = 0, E(X^2) = n, **Var(X)** = n

***======== example of probabilities ========***

**error correction**: encode n packets into n+k packets s.t. the recipient can reconstruct the original n packets from any n packets received. packets lost with prob p. X (packets received) ~ Bin(n+k, 1-p)

**Hash table**: n locations. A = event: insert m keys with no collisions, assuming uniformly distributed hash function.

Pr[A] = (1 - 1/n) \* (1 - 2/n) \* ... \* (1 - (m-1)/n). Take ln: ln(1-x) ~ -x. ln(Pr[A]) ~ -m^2/2n => Pr[A] ~ e^(-m^2/2n)

union bound: Ai: pair i has collision. Pr[bar A] <= sum Pr[Ai] = m(m-1)/2n

**throw m balls into n bins**. Y = number of empty bins. E[Y] = n(1-1/n)^m

**collect at least one copy of each of n different cards**. X = boxes need to. Xi = #boxes buy while trying to get the ith new card.

E(Xi) = n / (n - i + 1) => E(X) = n sum\_{i=1}^n 1/i ~ n (ln n + lambda).

lambda = 0.5772... Euler constant

***======== approximation ========***

**Markov's inequality**: For a nonnegative r.v. X with E(X) = miu, and any alpha > 0, Pr[X >= alpha] <= E(X) / alpha

proof technique: note sum\_a a\*Pr[X=a] >= sum\_{a >= alpha} a\*Pr[X=a]

**Chebyshev's inequality**: For a random variable X with expectation E(X) = miu, and for any alpha > 0, Pr[|X-miu| >= alpha] <= Var(X) / alpha^2

proof technique: define r.v. Y := (X-miu)^2. Pr[|X-miu| >= alpha] = Pr[Y >= alpha^2].

corollary: r.v. X with E(X) = miu, sigma = sqrt(Var(X)), Pr[|X-miu| >= beta\*sigma] <= 1 / beta^2

**i.i.d**: independent, identically distributed

estimate a proportion p by taking a small sample

An = Sn / n = (X1 + ... + Xn) / n

Var(An) = Var(Xi) / n = p(1-p) / n

Pr[|An-p| >= epsilon\*p] <= Var(An) / (epsilon\*p)^2 <= delta

=> n >= (sigma^2 / miu^2) \* 1 / (epsilon^2 \* delta)

=> n >= (1-p)/p\*1/(epsilon^2 \* delta)

**Law of Large Numbers**: X1, ..., Xn i.i.d. r.v. with common E(Xi) = miu. Define An := 1/n sum\_i Xi. Then for any alpha > 0, we have Pr[|An - miu| >= alpha] -> 0 as n -> oo

***======== conditional distribution*** ***========***

**conditional distribution**: X given Y=b is collection of values {(a, Pr[X=a|Y=b]): a in A}

**conditional expectation**: E(X|Y=b) = sum\_{a in A} a \* Pr[X=a|Y=b]

**conditional independence**: A and B are said to be conditionally independent given C if Pr[A, B|C] = Pr[A|C] \* Pr[B|C]

**total expectation law**: E(X) = sum\_{b in B} Pr[Y=b] \* E(X|Y=b)

**prior distribution:** {(i, Pr[X=i]): i = 1, ..., n}

**posterior distribution**: {(i, Pr[X=i|Y1=H]): i = 1, ..., n}

**Pr[Y2=H|Y1=H] = sum\_{i=1}^n Pr[X=i|Y1=H] \* Pr[Y2=H|X=i, Y1=H]**

**MAP (maximum a posteriori) rule**: guess the value a\* for which the conditional probability of X=a\* given the observations is the largest

**error probability analysis:**

Pr[E] = Pr[sum\_i Z\_i > n/2] = sum\_{k=ceil(n/2)}^n {n choose k} p^k(1-p)^(n-k)

= Pr[S > n/2] < Pr[|S-np|>n(1/2-p)] <= Var(S)/(n^2(1/2-p)^2) = p(1-p)/(1/2-p)^2 \* (1/n)

***======== continuous probability ========***

**probability density function**: a function f:R->R s.t. Pr[a<=X<=b] = int\_a^b f(x)dx for all a <= b

condition: int\_-oo^oo f(x) dx = 1

**X discrete**, **Y = cX** => distribution of Y: Pr[X=a] = Pr[Y=a/c]

**X continuous, Y = cX** => pdf of Y: f\_Y(x) = 1/c f\_X(x/c)

**expectation**: E(X) = int\_-oo^oo xf(x) dx

**variance**: Var(X) = E((X - E(X))^2 = int\_-oo^oo x^2f(x) dx - (int\_-oo^oo xf(x) dx)^2

**joint density**: a function f:R^2->R s.t. Pr[a<=X<=b, c<=Y<=d] = int\_c^d int\_a^b f(x,y) dxdy for all a<=b, c<=d

**independence for continuous r.v.'s**: events a<=X<=b and c<=Y<=d are independent for all a,b,c,d. or f(x,y) = f1(x)f2(y)

**exponential distribution**: f(x) = lambda\*e^(-lambda\*x) if x>=0 else 0 (lambda > 0) (with parameter lambda)

E(X) = 1/lambda, E(X^2) = 2/lambda^2, Var(X) = 1/lambda^2

Pr[X>t] = int\_t^oo lambda\*e^(-lambda\*x) dx = e^(-lambda\*t)

memoryless property: exponential distribution & geometric distribution

Pr[X>s+t | X>t] = Pr[X>s] (s > 0, t > 0)

X1 exp with param l\_1, X2 exp with param l\_2 => Y = min{X1, X2} exp with param l\_1+l\_2

**Normal distribution**: X ~ N(miu, sigma^2). f(x) = 1/sqrt(2\*pi\*sigma^2) \* e^(-(x-miu)^2 / (2\*sigma^2)) (sigma > 0)

when miu=0 and sigma=1, standard normal distribution

in general, Y = (X-miu) / sigma has the standard normal distribution

**Central Limit Theorem**: Let X1,...,Xn be i.i.d.r.v. with common expectation miu=E(Xi) and variance sigma^2=Var(Xi) (both assumed to be <oo). Define A'n = (An-miu)/(sigma/sqrt(n)) = ((sum\_i Xi)-n\*miu) / (sigma\*sqrt(n)). Then as n->oo, the distribution of A'n approaches the standard normal distribution in the sense that, for any real alpha,

Pr[A'n<=alpha] -> 1/sqrt(2pi) int\_-oo^alpha e^(-x^2/2) dx as n->oo

***======== cardinality and intractability*** ***========***

**bijection from N to Z**: f(x) = x/2 if x is even else -(x+1)/2

**Cantor-Bernstein theorem**: if |A| <= |B| and |B| <= |A| then |A| = |B|

**injection from N to Q**: N -> Z\*Z -> Q. The latter by mapping rational number to 2D points, and go by spiral

**finite binary strings countable** (prove by listing strings in increasing order of length, and then in lexicographic order)

[0, 1] uncountable. prove by **diagonalization**, **add 2 to each digit**

**cantor set**: repeatedly remove 1/3 ~ 2/3 of interval. **uncountable set of measure 0**

represent each number as ternary strings. C = {x in [0,1]: x has ternary representation of only 0's and 2's}

divide each ternary string by 2 -> onto [0, 1]

**S countably infinite => |P(S)| > |S|**

**halting problem**: Turing(P): if TestHalt(P, P) = "yes" then loop forever; else halt. Turing(Turing) contradiction