① $a \in \mathbb{Z}_n$ has inverse iff $(a,n)=1$

② extended euclidean algo: $(\log x)$

$g=(x,y)$ 则可以用 euclidean algo find a,b s.t. $g=ax+by$

③ a category of cryptosystem: one-time pad, rely on a key

key 为 random string, 至少和原文一样长; 且只被用一次.

比如 Caesar Cipher: $E_s(m) = c_1c_2\ldots c_n$, $c_i \equiv m_i + s \pmod{26}$

④ A sol to key-change problem: Diffie-Hellman protocol

$g \in \mathbb{Z}_p$ is a generator of $\mathbb{Z}_p$ if $\forall x \in \mathbb{Z}_p, \exists i \in \mathbb{N}$ s.t. $g^i \equiv x \mod (p)$

For prime p, $\mathbb{Z}_p$ has 至少一个 generator

Diffie-Hellman protocol

双方给出 $a, b \in \mathbb{Z}_p^+$ 作为各自的 secret key,

with public parameter p 以及大 prime num q

双方的 public key 分别为 $A=g^a \in \mathbb{Z}_p$, $B=g^b \in \mathbb{Z}_p$

shared secret key 为 $k=AB=g^{ab}\in\mathbb{Z}_p$

Diffie-Hellman Assumption

Give p, g, a, b, 没有 efficient algo 可计算 $g^{ab} \in \mathbb{Z}_p$

Discrete Logarithm Assumption

Given $q, g, x \in \mathbb{Z}_q$, 没有 efficient algo 可计算 $i \in \mathbb{N}$ st. $g^i = x \in \mathbb{Z}_p$ $\quad \in [0, q-1]$

Fermat's Little Thm Let p be prime,

$\forall a \in \mathbb{Z}_p$, $a^{p-1}=1 \in \mathbb{Z}_p$. (或 $a^p = a$)

let p, q be primes, 取 $n=pq$

$\Rightarrow$ $\mathbb{Z}_n^+$ 中, 和 n coprime 的数有 $\varphi(pq)=(p-1)(q-1)$ 个 (显然)

且 $\forall k \in \mathbb{N}$, $\boxed{a^{1+k\varphi(n)} = a \in \mathbb{Z}_n}$ ☆

RSA Encryption Protocol

Bob 希望给 Alice 发消息: m

Alice 选择两个大 prime p, q st. $m < pq$

并选择 $e \in \mathbb{Z}_n^+$ s.t. $(e, \varphi(n)) = 1$

$\varphi(n) = (p-1)(q-1)$

Alice 计算 $d = e^{-1} \in \mathbb{Z}_{\varphi(n)}$ 作为她的 private key

最后把 n 和 e 发给 Bob

Bob 将 $c = m^e \in \mathbb{Z}_n$ 作为 ciphertext 发给 Alice.

(Alice 只需要计算 $m = c^d \in \mathbb{Z}_n$ 即可)

RSA Assumption

given n 作为两个 prime 的 product, e st $(e, \varphi(n))=1$

c st. $c=m^e$, 没有计算 m 的 efficient algo

Factorization Hardness Assumption 没有 prime factorize $n \in \mathbb{Z}$ 的 efficient algo

$$Pr[X=a_i] = \sum_{w\in\Omega, X(w)=a_i} Pr[w]$$

$$E[X] = \sum_i a_i \cdot Pr[X=a_i] = \sum_{w\in\Omega} X(w) Pr(w)$$

Averaging Argument 至少有一个 $w \in \Omega$ st. $X(w) \geq E[X]$ (S)

Markov's ineq Let $v>0$; $X[w] \geq 0$ for all w ☆

$$\Rightarrow \boxed{Pr[X \geq v] \leq \frac{E[X]}{v}}$$

reverse M ineq if we can bound X for b

$$\Rightarrow \boxed{Pr[X \geq v] \geq \frac{E[X]-v}{b-v}}$$

Quick sort

```
Quicksort(A[1,...,n])
  if n=1, return A
  p = random index
  (L,R) = partition(A,p)
  return Quicksort(L) ∪ A[p] ∪ Quicksort(R)
```

expected: $O(n \log n)$

$$Var(X) = E[(X-E[X])^2] = E[X^2] - E^2[X], \quad \sigma(X) = \sqrt{Var(X)}$$

if $X = \begin{cases} 0 & \text{for } 1-p \\ 1 & \text{for } p \end{cases} \Rightarrow E[X]=p$, $Var[X] = p(1-p)$

$Var(cX) = c^2 Var(X)$

For idd X_i, $E[\prod_i X_i] = \prod_i E[X_i]$, $Var(\sum_i X_i) = \sum_i Var(X_i)$ ☆

Chebyshev $P[|X - E[X]| \geq a] \leq \frac{Var(X)}{a^2}$

且 For i.d.d. X_i, $Pr[|\frac{1}{n}X - E[X_i]| \geq \varepsilon] \leq \frac{Var(X_i)}{\varepsilon^2 n}$ for each X_i

Protocol 245 (RSA Signature) Suppose Alice wishes to send a message m to Bob and allow Bob to verify that he receives the intended message. As with the *RSA encryption protocol* (page 252), Alice computes (e, n) as her public key and sends them to Bob, and she computes $d \equiv e^{-1} \pmod{\phi(n)}$ as her private key. Then:

- Alice computes
$$s \equiv m^d \pmod{n}$$
and sends m and s to Bob.
- Bob computes
$$m' \equiv s^e \pmod{n}$$
and verifies that the result is equal to m.

he correctness of this scheme follows from the correctness of RSA encryption; we have:

$$\begin{aligned} m' &\equiv s^e \pmod{n} \\ &\equiv m^{ed} \pmod{n} \\ &\equiv m^{1+k\phi(n)} \pmod{n} \\ &\equiv m \pmod{n} \end{aligned}$$

Hoeffding's For (i.d.d.) indicator random variables X_i,

令 $X = \sum X_i$

$\Pr[\frac{1}{n}X \geq p-\varepsilon], \Pr[\frac{1}{n}X \leq p-\varepsilon] \leq e^{-2\varepsilon^2 n}$

(so $\Pr[|\frac{1}{n}X - p| \geq \varepsilon] \leq 2e^{-2\varepsilon^2 n}$)

Polling Confidence level $1-\delta$, margin of error ε

令 X 代表 supporter, n 代表总人数

$X = \sum_i X_i$ where $X_i = \begin{cases} 1, \text{第i个人 support} \\ 0, \sim\text{不 support} \end{cases}$

$E(X_i) = np$

如果 $\Pr[|\frac{X}{n} - p| \leq \varepsilon] \geq 1-\delta$

则称 polling n 个人是可以得到 $\frac{X}{n}$ 和真实 p 的差距 $\leq \varepsilon$ 的 confidence level $\geq 1-\delta$

(目的是：求出 n 应该多大，$\frac{X}{n}$ 才接近真实民意)

By Chernoff bound, $n \geq \frac{1}{2\varepsilon^2}\ln(\frac{2}{\delta})$

For multicandidate:

Theorem 211 (Union Bound) *Let $E_1, E_2, \ldots, E_m$ be a sequence of events over the same sample space. Then*

$$\Pr\left[\bigcup_{i=1}^{m} E_i\right] \leq \sum_{i=1}^{m} \Pr[E_i]$$

In other words, the probability of the union is no more than the sum of the probabilities of the individual events.

Theorem 210 (Sampling Theorem) *Suppose n people are polled to ask which candidate they support, out of m possible candidates. Let $X^{(j)}$ be the number of people who state that they support candidate j, and let p_j be the true level of support for that candidate. We wish to obtain*

$$\Pr\left[\bigcap_j \left(\left|\frac{X^{(j)}}{n} - p_j\right| \leq \varepsilon\right)\right] \geq 1-\gamma$$

In other words, we desire a confidence level $1-\gamma$ that all estimates $X^{(j)}/n$ are within margin of error $\pm\varepsilon$ of the true values p_j. We obtain this when the number of samples n is

$$n \geq \frac{1}{2\varepsilon^2}\ln(\frac{2m}{\gamma})$$

Theorem:
A language is in NP if and only if some nondeterministic polynomial time Turing machine decides it.

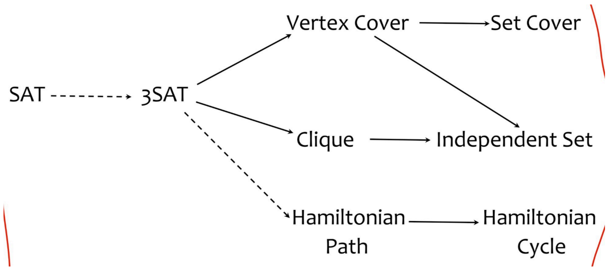
Formal definition: A problem **L** is **NP-hard** if:
for EVERY problem **X** in NP, **X** $\leq_p$ **L.**

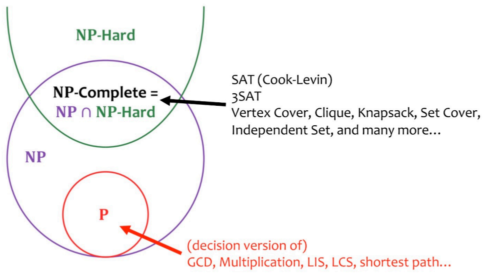
A problem **L** is NP-complete if
- **L ∈ NP**
- **L** is **NP**-hard

Defn: A $\leq_p$ **B** if there is a poly-time-computable function **f** where

x is a yes-instance of **A** ⇔ **f(x)** is a yes-instance of **B.**

A **Boolean formula** Φ is made up of:
- "literals": variables and their negations (e.g. x, y, z, ¬x, ¬y, ¬z)
- OR: ∨
- AND: ∧



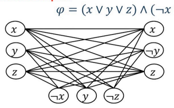


Construction of G_φ:
- For each clause, make a vertex for each literal
- Add an edge between two literals in *different clauses* only if they're "compatible"
  - They refer to different variables (e.g. x and ¬y) **or**
  - They are the same (e.g. z and z)

Set k_φ to m --- number of clauses

Concrete example:

$$\varphi = (x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z)$$



Observations:
1) Any clique can have at most one vertex from a clause ($k_\varphi \leq 3$)
2) A clique can have several x or $\neg x$, but not both

3SAT $\leq_p$ VC

- φ unsat ⇒ G_φ has no k_φ-VC

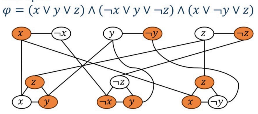
- **Construction of G_φ:**
  - **add** *variable gadgets* and *clause gadgets (for every variable and clause)*
  - **add** edge $\{u, v\}$ if
    - u is in a *variable gadget* and
    - v is in a *clause gadget and*
    - u and v are *labeled the same*
- **Set k_φ** to $n + 2m$ (n – number of variables, m – number of clauses)

- Concrete example:

$$\varphi = (x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z)$$

variable gadgets

clause gadgets



Algorithm

Step 1: Find an **MST** (in polynomial time)

Step 2: Walk around the perimeter of the **MST** to form **"tree-tour"**

tree-tour is not a legitimate TSP tour!

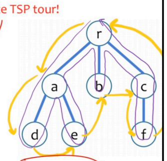
走两遍(来回)

Step 3: "Shortcut" tree-tour

- Repeatedly visit
  the next unvisited vertex in tree-tour

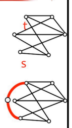
Why shortcut tree-tour ≤ **tree-tour** ?

- **Triangle inequality!**



Details

Step 1: describe the mapping
- Given an instance $G = (V, E)$ for HP
- An instance G' for HC is obtained by
  - Adding a path (s, x, t) into G



Step 2: prove correctness
- "Yes"-HP instance ⇒ "Yes"-HC instance
  - Suppose there is a (s, t)-HP P in G, how to construct an HC in G'?
  - Just add (s, x, t) into P to get a HC.
- "Yes"-HC instance ⇒ "Yes"-HP instance
  - Suppose there is an HC C in G', how to construct an HP in G.
  - Observe that $(s, x, t) \subseteq C$. So $P = C \setminus (s, x, t)$ is an (s, t)-HP.

Graph Cuts

def①

- A **cut** of a graph is a *partition of* its vertices $(S, \bar{S})$

def②

- An edge **crosses** the cut $(S, \bar{S})$ if one of its endpoints is in S and the other is in $\bar{S}$.

def③

- The **size** of a cut $(S, \bar{S})$ is the number of edges crossing it.