CS 70 Discrete Mathmatics and Probability Theory

Lecture 0. Set

Set { } the objects are called elements of the set

$A = \{2, 3, 5\} \Rightarrow 2 \in A,\ 4 \notin A$

The order and repetition of elements don't matter the set. Two sets A and B are said to be equal as $A = B$.

Also, the set can be written as a different notation $Q = \{\frac{a}{b} \mid a, b \text{ are intergers}, b \neq 0\}$.

Cardinality 基数 (the size of a set)

$A = \{1, 2, 3, 4\} \Rightarrow |A| = 4$

$|A \cup B| = |A| + |B| - |A \cap B|$

$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$

A set can also have infinite numbers like all intergers, prime numbers(素数).

$\emptyset$ empty set, $A = \emptyset \Rightarrow |A| = 0$

Proof: $A \Delta B = (A \cup B) \cap \neg(A \cap B)$

$\Leftrightarrow A \Delta B = (A \cup B) \cap (\neg A \cup \neg B)$

$= (\neg A \cup \neg B) \cap (A \cup B)$

$= [(\neg A \cup \neg B) \cap A] \cup [(\neg A \cup \neg B) \cap B]$

$= [(A \cap \neg A) \cup (A \cap \neg B)] \cup [(B \cap \neg A) \cup (B \cap \neg B)]$ ($\emptyset$)

$= \{x \mid (x \in A \text{ and } x \notin B) \text{ or } (x \in B \text{ and } x \notin A)\}$

Subsets and Proper Subsets

A is a subset of B, written $A \subseteq B$.

B is a subset of A, written $B \subseteq A$.

A is a proper subset of B, written $A \subset B$. (meaning A excludes at least one element of B).

B is a proper subset of A, written $B \subset A$.

* $A \subseteq A$, $\emptyset \subset A$, $\emptyset \subseteq A$ (A is not a empty set).

De Morgan's laws

[The algebra of set 3]

$\neg(A \cup B) = \neg A \cap \neg B$

$\neg(A \cap B) = \neg A \cup \neg B$

[The algebra of set 1]

$(A \cap B) \cap C = A \cap (B \cap C)$

$(A \cup B) \cup C = A \cup (B \cup C)$

Intersections and Unions

The intersections of A and B, written $A \cap B$ (Two sets are disjoint if $A \cap B = \emptyset$)

The union of a set A and B, written $A \cup B$

$A \cup B = B \cup A$, $A \cup \emptyset = A$, $A \cap \emptyset = \emptyset$, $A \cap B = B \cap A$,

Distributive laws

[The algebra of set 2] *"$A \cup B \cap C$" 不成立 必须带括号否则结果不一致

$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

$A \Delta B = \{x \mid (x \in A \text{ and } x \notin B) \text{ or } (x \in B \text{ and } x \notin A)\}$

Complements (补集).

the relative complements of A in B., written as "B-A" or "$B \setminus A$", meaning $\{x \in B \mid x \notin A\}$

$A = \{3, 4, 5\}$ $B = \{1, 2, 3\} \Rightarrow B \setminus A = \{1, 2\}$

$A \setminus A = \emptyset$, $A \setminus \emptyset = A$, $\emptyset \setminus A = \emptyset$

Significant Sets

$\mathbb{N}$ natural numbers $\{0, 1, 2, \cdots\}$.

$\mathbb{Z}$ intergen numbers $\mathbb{Z}^+, \mathbb{Z}^-$

$\mathbb{Q}$ rational numbers (有理数) $\{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\}$

$\mathbb{R}$ real numbers

$\mathbb{C}$ complex numbers (复数 $a+bi$)

补充：Perrin 数列 定义如下：

$P(0)=3$, $P(1)=0$, $P(2)=2$, $\forall n \geq 3$ and $n \in \mathbb{Z}^+$,

$P(n) = P(n-2) + P(n-3)$

在学界普如此定义：$\forall n > 1$, $n \in \mathbb{Z}^+$, n is a Perrin Number if $P(n)$ is divisible by n

1899年, if Perrin number = Prime numbers ⇒ Every prime number is a perrin number.

1995年 找到不是 prime 的 Perrin numbers : 271441

Products and Power Sets

$A_1 \times A_2 \times \cdots A_n = \{(a_1, a_2, \cdots a_n) \mid a_1 \in A_1, \cdots a_n \in A_n\}$ if $A_1 = A_2 = \cdots A_n$ ⇒ written as A^n

Cartesian product / cross product of two sets A and B, written $A \times B$.

$A \times B = \{(a,b) \mid a \in A, b \in B\}$. $A=\{1,2,3\}$, $B=\{1,2\}$ ⇒ $A \times B = \{(1,1), (1,2), (2,1), (2,2), (3,1), (3,2)\}$

The power set of S; denoted by $\mathcal{P}(S)$ is the set of all subsets of S

$\mathcal{P}(S) = \{T \mid T \subseteq S\}$, if $|S| = k$ then $|\mathcal{P}(S)| = 2^k$.

$S = \{1,2,3\}$ $\mathcal{P}(S) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{3,1\}, \{1,2,3\}\}$

specially (1) R^2 (R real numbers) is the Cartesian plane.

(2) B^n are called bit strings of length n, $B=\{0,1\}$ (to represent the subsets of set) for example, we have a subset $A \subseteq S$, the characteristic of A is $(b_1, \cdots b_n) \in B^n$ where $b_i = \begin{cases} 1, & s_i \in A \\ 0, & s_i \notin A \end{cases}$

$A=\{1,3,5\}$, $B=\{3,4\}$, $S=\{1,2,3,4,5\}$

⇒ the characteristic of vector of A is (1,0,1,0,1) 不要写成{}这是向量

Theroem:

1. $\mathcal{P}(A \cap B) = \mathcal{P}(A) \wedge \mathcal{P}(B)$
2. $\mathcal{P}(A \cup B) \neq \mathcal{P}(A) \cup \mathcal{P}(B)$

eg: $A=\{1,2,3\}$ $B=\{4,5,6\}$.

$2^3 + 2^3 = 16 \neq 2^6$.

Sum and Products

$1+2+\cdots+n = \sum_{i=1}^{n} i$

$f(m) + \cdots f(n) = \sum_{i=m}^{n} f(i)$.

$f(m) f(m+1) \cdots f(n) = \prod_{i=m}^{n} f(i)$.

Universal and Existential Quantifiers

"$\forall$" "$\exists$"

Propositional Logic

proposition: a statement which is either true or false.

The statements that are propositions: $1+1=5$, $\sqrt{3}$ is irrational, Julia ate 2 eggs on his 10th birthday

the statements that are not propositions: $2+2$, $x^2+3x=5$ (Reason: what is x?), Andrew often eats eggs (Reason: often?), Harry was unpopular (unpopular?).

⇒ proposition should not include fuzzy terms. * propositional variables are called atomic formulas

for example, P, Q, R are variables representing proposition

$P \wedge Q$ (P and Q) True only both are true [Conjunction]

$P \vee Q$ (P or Q) True at loast one is true [Disjunction] } Propositional forms.

$\neg P$ (not P) True when P is false [Negation] 2.

law of excluded middle (排中律) either P or $\neg P$ is True $\Rightarrow$ $P \vee \neg P$ is always True

Tautology: A propositional form that is always true regardless of the truth values of its variables

$P \wedge \neg P$ is always False, is called Contradiction.

A useful tool for describing the possible values of a propositional form is a truth table.

P	Q	$P \wedge Q$
T	T	T
T	F	F
F	T	F
F	F	F

P	Q	$P \vee Q$
T	T	T
T	F	T
F	T	T
F	F	F

P	$\neg P$
T	F
F	T

$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

$P \vee (Q \vee R) \equiv (P \vee Q) \vee R$

$P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$

[Implication] is also a propositional form and it is most important.

$P \Rightarrow Q$, meaning "If P, then Q" or "Q if P" or "P only if Q" be true if Q"

P is called hypothesis (假设) and Q is the conclusion.

P	Q	$P \Rightarrow Q$	$\neg P \vee Q$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

$P \Rightarrow Q$ is always True if P is false, like "If pigs can fly, then horses can read.", this statement is true, called Vacuously True if hypothesis is false and statement is true.

From truth table, we can see $P \Rightarrow Q$ is logically equivalent to $\neg P \vee Q$, written $(P \Rightarrow Q) \equiv (\neg P \vee Q)$.

Proof: $(P \Rightarrow Q) \equiv (\neg P \vee Q) \equiv (Q \vee \neg P) \equiv (\neg(\neg Q) \vee \neg P)$

$\equiv (\neg Q \Rightarrow \neg P)$

"Logical equivalent"

if $P \Rightarrow Q$ and $Q \Rightarrow P$ then we say "P if and only if Q" or "P iff Q" we write $P \Leftrightarrow Q$.

P	Q	$\neg P$	$\neg Q$	$P \Rightarrow Q$	$Q \Rightarrow P$	$P \Leftrightarrow Q$	$\neg Q \Rightarrow \neg P$
T	T	F	F	T	T	T	T
T	F	F	T	F	T	F	F
F	T	T	F	T	F	F	T
F	F	T	T	T	T	T	T

From truth table, we can conclude that:

(1) $P \Leftrightarrow Q$ is true only "P is True and Q is True" or "P is False and Q is False"

(2) $(P \Rightarrow Q) \equiv (\neg Q \Rightarrow \neg P)$ $\quad Q \Rightarrow P$.

Contrapositive 逆否命题. Converse 逆命题.

Quantifiers 数量词

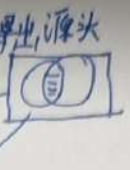
To express statements mathematically, we need two quantifiers: the universal quantifier "$\forall$" and the existential quantifier "$\exists$"

We can express propositions without quantifiers, using disjunctions and conjunctions.

For example, $(\exists x \in U)(P(x)) \equiv (P(1) \vee P(2) \vee P(3) \vee P(4))$ and $(\forall x \in U)(P(x)) \equiv P(1) \wedge P(2) \wedge P(3) \wedge P(4)$

comparison: $\forall x \in \mathbb{Z}, \exists y \in \mathbb{Z}, x<y$ (✓) $\exists y \in \mathbb{Z}, \forall x \in \mathbb{Z}, \ldots$

Much Ado About Negation



De Morgan's Law: $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$

$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$

$\neg(\forall x\, P(x)) \equiv \exists x\, \neg P(x)$

$\neg(\exists x\, P(x)) \equiv \forall x\, \neg P(x)$

eg: $\neg(\forall x \exists y P(x,y)) \equiv [\exists x \neg(\exists y P(x,y))] \equiv (\exists x \forall y \neg P(x,y))$

eg: 1. there are at most three distinct intergers x that satisfy $P(x)$

1. $\exists x \exists y \exists z\, \forall d\, (P(d) \Rightarrow d=x \vee d=y \vee d=z)$ 注: 这种写法为限制了个数 ≤ 3 个.

2. $\forall x \forall y \forall z \forall v\, (x \neq y \wedge x \neq z \wedge x \neq v \wedge y \neq z \wedge y \neq v \wedge z \neq v) \Leftrightarrow \neg(P(x) \wedge P(y) \wedge P(z) \wedge P(v))$

3. there are at least three distinct intergers x that satisfy $P(x)$.

$\exists x \exists y \exists z\, (x \neq y \wedge x \neq z \wedge y \neq z \wedge P(x) \wedge P(y) \wedge P(z))$ 注: 这种写法其实是说至少有3个.

(比如对 1,2,3,4, 其实把 $(x,y,z)=(1,2,3)$ 或 $(2,3,4)$ 都成立 同时这其实可以写成 $\exists x \exists y \exists z (x \neq y \wedge y \neq z \wedge z \neq x) \wedge \forall d\, (d=x \vee d=y \vee d=z \Rightarrow P(d))$ 形式.

4. there are exactly three distinct intergers x that satisfy $P(x)$.

$\exists x \exists y \exists z (x \neq y \wedge y \neq z \wedge z \neq x) \wedge \forall d\, (d=x \vee d=y \vee d=z \Leftrightarrow P(d))$.

Predicate 断言: Relation R holds between object x and object y: $R(x,y)$ — object x has property P: $P(x)$ — Student(x) = "x is a student"

~~"$\sum_{i=1}^{n} i(i+1)/2$"~~ "$n = \sum_{i=1}^{n} \frac{i(i+1)}{2}$"

"$x>z$"

"n is even and the sum of two primes"

} have a free variable, we call them predicates. which is same as boolean valued functions. like.

$P(n) =$ ~~"$n = \sum_{i=1}^{n} \frac{i(i+1)}{2}$"~~ $\Rightarrow P(n) = \begin{cases} \text{True}, & n \in A \\ \text{False}, & n \notin B \end{cases}$

Auxiliary symbols 辅助符号

Signatures consist of { predicate symbols → with parameters; function symbols → with parameters; constant symbols }

we call predicate/function symbols { unary if arity/parameters = 1; binary ~ 2; k-ary ~ k }

e 1.

ofs : Notation C符号) and basic facts

We use the notation ":=" to indicate a definition C赋值操作) 有别于 "="

For example, $q := 6$ defines variable q as having value 6 / d. the sets of intergers and natural members are closed under addition and multiplication 乘法

Direct ~~Discrete~~ Proof

For example about theorem : For any $a, b, c \in \mathbb{Z}$, if $a|b$, ~~$\in \mathbb{Z}$~~ and $a|c$ then $a|(b+c)$

$$(\forall a,b,c \in \mathbb{Z})((P(a,b) \wedge P(a,c)) \Rightarrow P(a, b+c))$$

solution : Assume that $a|b$ and $a|c$, i.e. there exists intergers q_1 and q_2 such that $b = aq_1$, $c = aq_2$. Then $b+c = (q_1+q_2)a$. Since the $\mathbb{Z}$ is closed under addition we conclude that $(q_1+q_2) \in \mathbb{Z}$ and so $a|(b+c)$

For another example : Let $0 < n < 1000$ be an interger if the sum of digits M(n) of n is divisible by 9, then n is divisible by 9 $\Leftrightarrow (\forall n \in \mathbb{Z}, 0<n<1000)(M(n) \Rightarrow P(n))$ P(n)

Proof : $n = \overline{abc} = 100a + 10b + c$ Assume the sum of digits of n is divisible by 9, then $\exists k \in \mathbb{Z}^+$, $a+b+c = 9k$. Since $99a + 9b = 9(11a+b)$ that $n = 9(k+11a+b)$. We conclude that n is divisible by 9.

反证法：适用于一个命题推导另一个命题

Proof by Contraposition : To prove $P \Rightarrow Q$ Assume $\neg Q \Rightarrow \cdots \Rightarrow \neg P$ which is equivalent to $P \Rightarrow Q$

Example 1. Let n be a positive integer and let d divide n. If n is odd then d is odd.

Assume that d is even, then $d = 2k$. Because $d|n$ then $n = d\ell = 2k\ell$ then n is even $\Rightarrow$ If n is odd then d is odd.

适用于单个命题，目的找反例(Counter-example)

~~Example 2~~

矛盾法 Sometimes we need lemma C引理)

Proof by Contradiction : To prove P Assume $\neg P \Rightarrow \cdots R \Rightarrow \cdots \neg R$ then $\neg P \Rightarrow R \wedge \neg R \Rightarrow P$

Example : There are infinitly many prime numbers.

Suppose the statment is false then there are finitly numbers, assumed as $p_1, p_2, \cdots p_k$, $k \in \mathbb{Z}^+$

Define number $q = p_1 p_2 \cdots p_k + 1$ can not be divided by any prime numbers then q is a prime number and $q > \max\{p_1, \cdots p_k\}$ Therefore the statement is True

Proof by Cases (分类讨论)

1. There exists irrational numbers x, y such that x^y is rational.

Let x, y both are $\sqrt{2}$, if $\sqrt{2}^{\sqrt{2}}$ is rational then the statment is true. Otherwise, $\sqrt{2}^{\sqrt{2}}$ is irrational, then let $m = \sqrt{2}^{\sqrt{2}}$ $n = \sqrt{2}$ $m^n = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = (\sqrt{2})^2 = 2$ is rational.

* This is a non-constructive proof (非构造性证明): We prove x exists but without explictly revealing what x is

Common Errors When writing proofs.

1. Don't assume the claim you aim to Prove: Like proof: $P \Rightarrow Q$ Assume: $P \Rightarrow Q \Rightarrow \cdots$ $\cdots \Rightarrow P \Rightarrow Q$

2. Never forget to consider the case where your variables corresponding with 0

Example: Let $x=y \Rightarrow x^2 - xy = x^2 - y^2 \Rightarrow (x-y)x = (x+y)(x-y) \Rightarrow x = x+y \Rightarrow x = y = 0$

but from statment we just order $x=y$ that $x=y=k$, k can be any intergers.

The reason is that $x-y=0$ cannot be deleted.

3. Be careful when mixing negatives with inequalities.

~~Like~~: Like: $-2 \le 1 \Rightarrow (-2)^2 \le 1^2 \Rightarrow 4 \le 1$ (X) Reason: $|a| \le |b| \Rightarrow |a|^2 \le |b|^2$ only

Style and substance in proofs: Think before proving.

在大型证明中由很多引理构成 $\xrightarrow{\text{类比}}$ 我们应有很多子程序构成 可以在生活中经常被使用

Amazing Proof:

1. $x^5 - x + 1 = 0$ has no solution in rationals. (contradiction + cases!).

Assume $x^5 - x + 1 = 0$ has solution in nationls, defined as $\frac{a}{b}$, $a, b \in \mathbb{Z}$, $b \neq 0$; $(a, b) = 1$

$\Rightarrow a^5 - a^5 b^4 + b^5 = 0$ Case 1. a is odd, b is odd. $\Rightarrow a^5 - a^5 b^4 + b^5$ is odd ×.

Case 2. a is odd, b is even $\Rightarrow a^5 - a^5 b^4 + b^5$ is odd ×.

Case 3 a is even b is odd ⇗

Case 4 a and b is even $\Rightarrow (a, b) = 2 \neq 1$ ×

$\Rightarrow$ ~~R~~ statement is true.

2. $f(n) = n^2 + n + 41$. $\forall n \in \mathbb{N}$, $f(n)$ is a prime number. Whether the statmort is true?

$f(n) = n(n+1) + 41$ Let $n = 41$ $\Rightarrow f(41) = 41 \times 42 + 41 = 41 \times 43$. $\Rightarrow$ the statment is false

2.

ne:

Mathematical Induction

2. Strengthening the Induction Hypothesis

3. Simple Induction vs. Strong Induction

4. Recursion, programming, induction.

5. False proofs

I Mathematical Induction 数学归纳法

Theorem 1. $\forall n \in N, \sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

Proof: We proceed by induction on the variable n.

(Base case) $n=0$: we have $\sum_{i=0}^{0} i = 0 = \frac{0(0+1)}{2}$, Thus the base is correct.

(Induction Hypothesis) For arbitrary (任意) $n=k \geq 0$, assume that $\sum_{i=0}^{k} i = \frac{k(k+1)}{2}$

(Inductive Step) Prove the statement for $n=k+1$, i.e. show that $\sum_{i=0}^{k+1} i = \frac{(k+1)(k+2)}{2}$

$\sum_{i=0}^{k+1} i = \sum_{i=0}^{k} i + (k+1) = \frac{(k+1)k}{2} + (k+1) = \frac{(k+1)(k+2)}{2}$, where the second equality follows from the Induction Hypothesis. By the principle of mathematical induction, the claim follows.

⇒ Three steps: Base case: Prove that P(0) is true

Induction Hypothesis: $\forall k \geq 0$, assume P(k) is true

Induction Step: With the assumption, show that P(k+1) is true.

II Strengthening the Induction Hypothesis to improve the Proof: $\forall n \geq 1, n \in N, \sum_{i=1}^{n} \frac{1}{i^2} \leq 2$.

Proof: If we prove $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n}$ ✗✗, then we can prove the statment.

~~n=0~~ $n=1$, $1 \leq 2-1$ the base is correct.

$\forall k \geq 1$, $n=k$, assume that $\sum_{i=1}^{k} \frac{1}{i^2} \leq 2 - \frac{1}{k}$, we need prove for $n=k+1$, $\sum_{i}^{k+1} \frac{1}{i^2} \leq 2 - \frac{1}{k+1}$

$$\sum_{i}^{k+1} \frac{1}{i^2} = \sum_{i}^{k} \frac{1}{i^2} + \frac{1}{(k+1)^2} \leq 2 - \frac{1}{k} + \frac{1}{(k+1)^2} = \frac{k - k^2 - 2k - 1}{k(k+1)^2} + 2 \leq 2 - \frac{k^2+k}{k(k+1)^2} = 2 - \frac{1}{k+1}$$

~~$\frac{1}{k+1} \leq \frac{1}{k} \Rightarrow \frac{1}{k+1} - \frac{1}{k+2} \leq \frac{1}{k} \Rightarrow k4$~~ By the principle of mathematical induction, the claim holds.

① you can change the hypothesis.
if we change it to $\sum_{i=1}^{n} \frac{1}{i^2} \leq 2 - \frac{1}{n^2} \Rightarrow \sum_{i=1}^{k} \frac{1}{i^2} \leq 2 - \frac{1}{k^2}$, $\sum_{i=1}^{k+1} \frac{1}{i^2} = \sum_{i=1}^{k} \frac{1}{i^2} + \frac{1}{(k+1)^2} \leq 2 - \frac{1}{k^2} + \frac{1}{(k+1)^2} \leq 2 - \frac{1}{(k+1)^2}$

$2k^2 \Leftrightarrow k^2+2k+1$

$k^2 \Leftrightarrow 2k+1$

$(k-1)^2 \Leftrightarrow 2$

$k-1 \Leftrightarrow \sqrt{2}$

$\Rightarrow k \geq 3$ ✓

② to promise the hypotheis is the subsets of provment to help calculate and reasoning

③ if you do not strengthen the hypothesis, sometimes it will be hard to solve

III. Simple Induction & Strong Induction

simple / weak induction: just assume P(k) is true $P(k) \Rightarrow P(k+1)$

strong induction: Assume P(0), P(1), P(2), ... P(k) is true. $P(0) \wedge P(1) \wedge P(2) \wedge \cdots \wedge P(k) \Rightarrow P(k+1)$

Pay attention to the following things:

1. Can strong induction prove statements which weak statements cannot? No!
2. Strong induction is much easier to use than Simple induction.

Theorem 2. $\forall n \geq 12, n \in \mathbb{N}$, n can be written as $4x+5y$, for $x, y \in \mathbb{N}$.

We proceed by induction on n.

$n=12$, $12 = 4\cdot 3 + 5\cdot 0$

$n=13$, $13 = 4\cdot 2 + 5\cdot 1$

$n=14$, $14 = 4\cdot 1 + 5\cdot 2$

$n=15$, $15 = 4\cdot 0 + 5\cdot 3$

Assume the claim holds for all $12 \leq n \leq k$, for $k \geq 15$, $k-3 = 4x_0 + 5y_0$

$n = k+1 \geq 16$, ~~$\Rightarrow (k+1)-4 \geq 12$~~ $\Rightarrow k+1-4 = k-3 \geq 12$,

$k-3 = 4x_0 + 5y_0 \Rightarrow k+1 = 4(x_0+1) + 5y_0$. Setting $x' = x_0+1$, $y' = y_0$ completes the proof.

Theorem 3 证明 $n \geq 2$, n 可以被写成一个或多个质数乘积.

① $n = 2 = 2\times 1$ ✓ $n=3$, $n = 4 = 2\times 2$, $n = 5\times 1 = 5$.

② 假设 $2 \leq n \leq k$ 时, ~~k=k~~ $k \geq 5$ 假设成立 $n=k$.

③ $n = k+1 \Rightarrow$ i $k+1$ prime ✓ ii $k+1 = pq$, $p \geq 2, q \geq 2 \Rightarrow 1 \leq p, q \leq k$., p, q 成立 $\Rightarrow$ $k+1$ 成立

说实话我觉得证明这个是有问题的. ↑ 这步是定义 但又是结论 也就是说这里从证明上升 循环论证了!

1 但是从此我们可以看出 Strong Induction 在创建多个已知条件的优势, 如单个证明 $42 = 6\times 7$ 就需要回溯 6 和 7 证明 C 在假设不知道 6, 7 是什么数下.)

IV Recursion (递归), programming and induction

1. F(0)=0, F(1)=1, $\forall n \geq 2, n \in \mathbb{N}$, F(n) = F(n-1) + F(n-2)

```
def function(x):
    if x == 0:
        return 0
    if x == 1:
        return 1
    return function(x-1) + function(x-2)
```

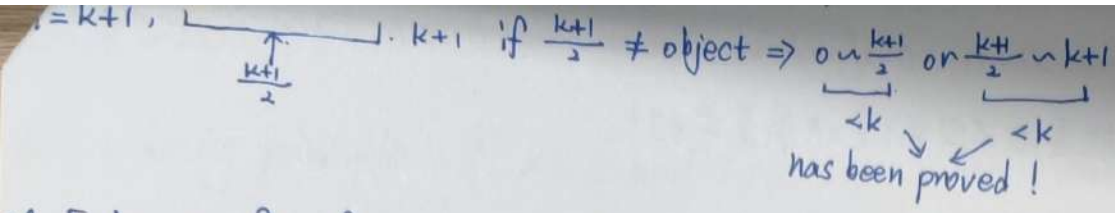
更简洁公式

但缺乏迭代次数了解 ⟹

```
def function2(x)
    if x == 0:
        return 0
    if x == 1:
        return 1
    a = 1
    b = 0
    for i in range(2, x+1):
        tmp = a
        a = a+b
        b = tmp
    return a
```

[Python]

8



V. False proofs often occur in "$\forall n \geq 1, P(n) \Rightarrow P(n+1)$" step.

Lecture 3.

Outline: Stable Matching Problem / Propose-and-Reject Algorithm / Residency Match / Properties of the Propose-and-Reject Algorithm / Optimality /

I. Stable Matching Problem

For example, n candidates and n jobs, you need to match each job with a candidate in order to make everyone happy.

Then we need the "Propose and Reject algorithm" aka. the "Gale Shapley algorithm"

II. The Propose-and-Reject Algorithm

We need to think of the algorithm as proceeding in "days" to have a clear unambigous sense of discrete time. 在算法中时间可以被抽象为特定步骤或对象作进行处理的不同阶段或周期.

Do the following step until no jobs are requested with "no":

Every morning: each job give an offer to most preferred candidate who has not yet reject your job

Every afternoon: each candidate choose the best offer he requested with "maybe" response and to other jobs with "no" response

Every evening: each job delet the candidate name who has rejected them from the list.

Ex:

Day	Candidates	Offers
1.	A	a, b
	B	c
	C	—
2.	A	a.
	B	c, b
	C	—
3.	A	a
	B	c
	C	b

III. The impacts of Propose-and-reject algorithm on Residency(医师) Match

早期抢人被禁止 → 时间紧张的患者是否加入/拒绝 offer → 医院对住院医生非常+
医生对医院排序 → 当下 Propose and reject algorithm [2012 Nobel Prize in Economis for extend this Algorithm]
有些人可能喜欢不同并不喜欢最靠前的!

IV. Properties of the Propose and Reject Algorithm

Basics we can see: doesnot run forever and output a "good" matching.

Lemma 4.1 This algorithm always halts(停止)

Proof: if algorithm doesnot halt ⇒ some candidate request "no" to a job

We have max n 个candidate so at max ~~we~~ one can only send ~~n(n-1)~~ n-1 "no" request.

for n jobs, can at max get n(n-1) "no", in at max n(n-1) days.

What properties should a "good" matching have?

⇛ Stability: a matching is unstable if exists a (job, candidate) both prefer working with each other over current situation. we call this pair "a rogue couple".

Do stable matching always exist? Surely yes! but it has some constraints.

Counter Example: we have an people who must be paired up to be roommates.

For n=2 ⇒

A	B → C → D.
B	C → A → D.
C	A → B → D
D	None

it doesn't have a stable matching:

{(A, B), (C, D)} B⇐⇒C

{(A, C), (B, D)} A⇐⇒B

{(A, D), (B, C)} A⇐⇒C

Conclusion: There always exists a stable matching in problem that have two different types. and we will prove it below.

Proof: from Observation: each job available option will get worse, while each candidate's offer will only get better and they must "meet" in the middle

C has a job offer in hand (on a string) which she likes at least as much as J. C(显然).
可以用 induction / contradiction 证明

Well ordering principle: if $S \subseteq N$ and $S \neq \emptyset$, then S has a smallest element.

Lemma 4.3: The algorithm always terminates with a matching.

terminates 意味未没有人 send "no", 若没有匹配, 则只可能 { ○ ○ ○ △ □ } ~~有人没有 offer 发不~~

~~若 no, 则此时由~~ 4.2引理说明 它至少有比余下的 J 你更喜欢的 J' ⇒ 这就有 n+1 个 Job 矛盾!

Theorem 4.1: The algorithm always terminates with a stable matching.

对∀(J, C), suppose J prefer C* than C, then J will first send C* then to C
so only when C* has a better choice, it will refuse J ⇒ (J, C*) cannot be a rogue couple

V. Optimality

There exists two or more probability for stable matching!

Ex:

Jobs	Candidates
1	A → B → C → D
2	A → D → C → B
3	A → C → B → D
4	A → B → C → D

Candidates	Jobs
A	1 → 3 → 2 → 4
B	4 → 3 → 2 → 1
C	2 → 3 → 1 → 4
D	3 → 4 → 2 → 1

⇒ {(1, A), (4, B), (2, C), (3, D)} (Optimal job for a candidate 4.3)
{(1, A), (4, B), (2, D), (3, C)} (Optimal candidate for a job) 4.2

Then there are two definition:

4.2 definition: For a given job J, the optimal candidate for J is highst rank candidate on J's perference list that J could be paired with any stable matching ~~4.3 definition~~

4.3 definition: For a given C, the optimal job for C is highest-ranked job on C's perference list that C could be paired with in any stabled matching.

Theorem 4.2: The matching output by the Propose-and-Reject algorithm is (a ~~job~~ optimal candidate for a job)

演示：

Day	Candidate	
1	A	1 2 3 4
	B	
	C	
	D	
2	A	1
	B	4
	C	3
	D	2

证明：如果不是 optimal candidate for a job

对取 (J, C) 由 suppose $\exists\ (J, C^*)$：

但算法 $\Rightarrow (J, C) \Rightarrow E(J^*, C^*)$

$(J^*, C^*) \neq (J, C)$

$J: C^* \to C$ 无法交换

$C^*: J^* \to J$

$\Rightarrow (J, C^*)$ 不存在 $\Rightarrow$ Algorithm 得到就是 optimal

Theorem 4.3 if a matching is a optimal candidate for a job, then it's also the pessimal (最差的) job for a candidate!

①从反面先由人发起让 job 接收来着的话

day 1. 由

1	A
2	C
3	D
4	B

③从理论上讲 candidate 每天在变好，达到最差的"好"就此停止了！

②证明：对 $\forall J$，均有 C^*（条件）

Suppose $\exists$ more pessimal ~~(J, C^*)~~ $\exists\ C_1^*: J_1 \to J_2 (A)$ $J_1: C_1^* \to C_2^*$（条件）

已有 (J, C_1^*)，(J_2, C_2^*) ~~(J_2, C_2^*)~~ $J_2: C_2^* \to C_1^*$（条件）

现实

(J_2, C_1^*) (A)

如果有 (J_2, C_1^*)，对于 (J_1, C') 和 (J', C_2^*)

$C_1^*: J_2$ 就此停止了没有收到 $J_1 \Rightarrow J_1$ 而言：$C' \to C_1^* \to C_2^*$

J_2 先发给 $C_2^* \to C_1^* \Rightarrow C_2^*: J' \to J_2$

继续 原本 $\exists (J_3, C')$ 现在是 $(J_{原}, C')$，(J_3, C'')

(J_1, C_1^*)

J_1 先发给 C' 但被拒了 $\Rightarrow C': J_3 \to J_1$ / $J_3: C'' \to C' \Rightarrow \cdots$

对 $\forall (J_i, C_i)$ 重建后 $C_i: \exists J_{原} \to J_i$ / $J_{原}: C_i \to C_{原}$

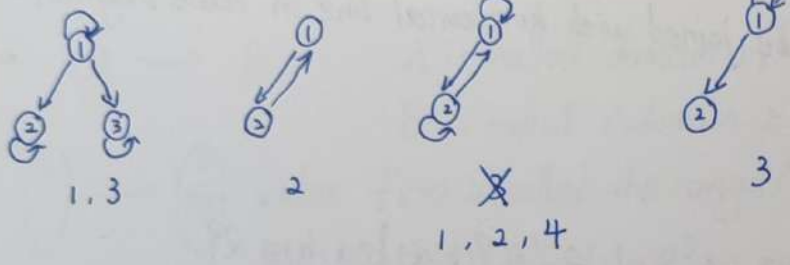
那么 (J_i, C_i) 反而是 optimal Candidate for a job 矛盾！

Summary：从这个算法最后揭示道理（T4.3）可见 Impact 对 Candidate 自由选择和生孩子选择是存在冲突的！

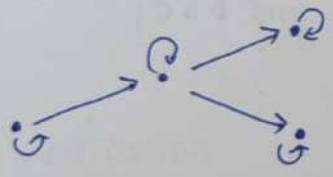
Relation:
A binary relation between two sets A and B (二元关系) is a subset of $A \times B$.
written in: $a \in A$, $b \in B$, R is a binary relation then we write "aRb" if $(a,b) \in R$.
read as "a is R-related to B"
*ternary: relations between three sets.
unary: relations are just subsets of a set $\{x \in \mathbb{Z}^+ \mid x \text{ is even}\}$

There are two representations $\begin{cases} \text{directed graphs 有向图} \\ \text{matrixes 矩阵 } M = \begin{cases} T, \text{ if } (a_i, b_i) \in R. \\ F, \text{ if } (a_i, b_i) \notin R. \end{cases} \end{cases}$

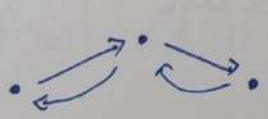
A binary relation on set A ($C A \times A$) is
- reflective 自反性 when xRx for all $x \in A$
- symmetric 对称性 when xRy and yRx for all $x, y \in A$.
- antisymmetric 非对称性 when xRy then yRx if $x \neq y$; only $x=y$ fulfill xRy and yRx.
- transitive 传递性 when xRy, yRz there must exist xRz for all $x, y, z \in A$.



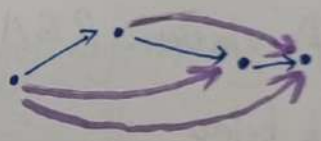
Reflective closure:



Symmetric closure:



Transitive closure:



defination: the transitive closure 传递闭包 是(a binary relation R on a set A, the transitive closure R^* of R is relation on A with following properties:
R^* is transitive; $R \subseteq R^*$; if S is a transitive relation on A and $R \subseteq S$ then $R^* \subseteq S$).

Equivalence Relations: A binary relation on R on a set A is called an equivalence relation if it's reflective, transitive and symmetric.
Equivalence Class (E_x) is defined by $E_x = \{y \mid yRx\}$, when R is a equivalence relation where exists an $x \in A$ and $y \in A$

Partial ordering 偏序关系: a relation that is reflective, transitive, [illegible]

for example: "$\geq$" labeled as "$\succcurlyeq$"

A set S with a partial ordering R is called a partially ordered set or poset, denoted by (S, R)

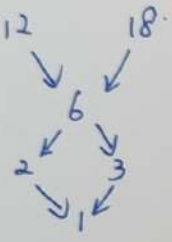
- Comparable: The elements (a & b) of a poset (S, R) are comparable if either $a \succcurlyeq b$ or $b \succcurlyeq a$
- Incomparable: The elements a & b of a poset (S, R) are incomparable if neithe $a \succcurlyeq b$ nor $b \succcurlyeq a$

Let $(\mathbb{Z}, \leq)$ be a poset, if $x \leq y$ and $x \neq y$ then we write $x < y$ and say x is a predecessor of y or y is a successor of x; What's more when $x < y$ and there is no z with $x < z < y$ then x is an immediate predecessor of y.

If $(S, \succcurlyeq)$ is a poset and every two elements of S are comparable, S is called totally ordered set/chain (linearly $\succcurlyeq$ total/linear order)

We use "Hasse Diagram" to visuall describe a partially ordered set.

for example "x divides y" on $\{1, 2, 3, 6, 12, 18\}$

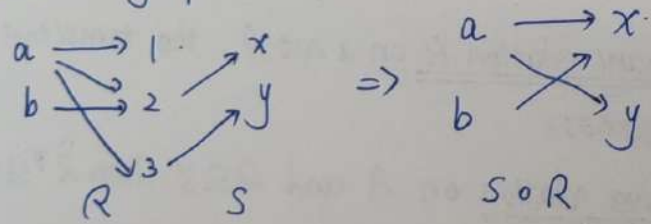
$\Rightarrow$ 

* Two nodes shouldn't be joined with horizontal line in Hasse Diagram.

Inverse Relation: $R \subseteq A \times B$ define $R^{-1} \subseteq B \times A$ and $R^{-1} = \{(b,a) \mid (a,b) \in R\}$

Composition of relation: $R \subseteq A \times B$, and $S \subseteq B \times C \Rightarrow S \circ R = \{(a,c) \mid$ existis $b \in B$ that aRb and $bSc\}$

Like the graph below:



Also, it can be written in logical matrix product: $P(i,j) = T$ if there exists l, $1 \leq l \leq m$ such that $M(i,l) = T$ and $N(l,j) = T$ with $P = M_R N_S$ $(S \circ R)$

$$\begin{pmatrix} T & T & T \\ F & T & F \end{pmatrix} \times \begin{pmatrix} F & F \\ T & F \\ F & T \end{pmatrix} = \begin{pmatrix} T & T \\ T & F \end{pmatrix}$$

* A relation S is transitive if and only if $S \circ S \subseteq S$

S is a relation then $S^1 = S$, $S^2 = S \circ S$, $S^n = \underbrace{S \circ S \circ \cdots \circ S}_{n \uparrow S}$

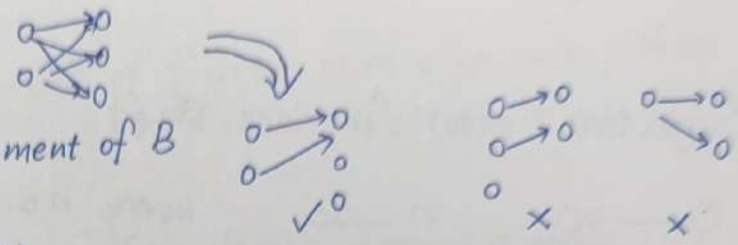
actions:

1. a Cartesian product $A \times B = \{(a,b) \mid a \in A \text{ and } y \in B\}$.
2. relation R is a subset of $A \times B$.
3. A function is a special kind of relation. we called A' as Domain 定义域 and B' as the Range of the relation 关系的值域 (A', B' are the set of all elements in R).

for example $R = \{(1,2), (2,-3), (3,5)\} \Rightarrow$ Domain $\{1,2,3\}$. Range $\{2,-3,5\}$

What is the meaning of function?

1. firstly it must be a binary ~~fun~~ relation



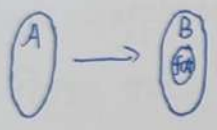
2. every element of A has only one specific element of B

written: f from A to B ($f: A \to B$), $b = f(a)$

a function is also called a mapping or a transformation.

$f: A \to B$. A is called domain of f.

B is called codomain 上域 of f.



value $f(x)$ is called ~~the~~ range/image of x under f

set $f(R) = \{f(r) \mid r \in R\}$ is called the range/image of R under f (where R is any subset of A and $f(R)$ is set of ~~ranges~~ images of element of R under f)

*$f(A) \subseteq B$

set $f(A) = \{f(a) \mid a \in A\}$ is called the range/image of A under f

⇕

range/image of f.

Clarification:

1. X Y.

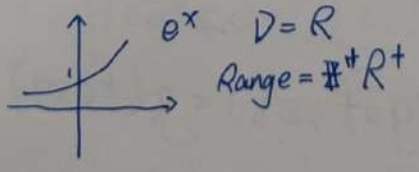
⇒ domain $\{1,2,3\}$.

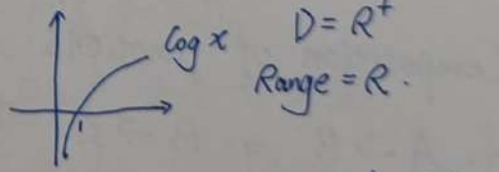
codomain $\{a,b,c,d\}$

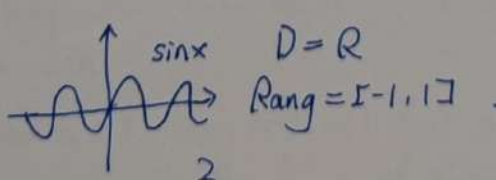
range of f $\{a,b\}$

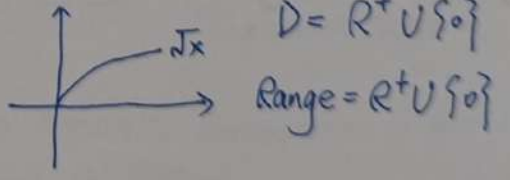
2. All functions are relations, not all relation are functions

3. example

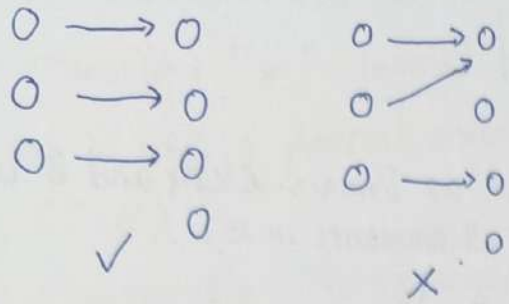








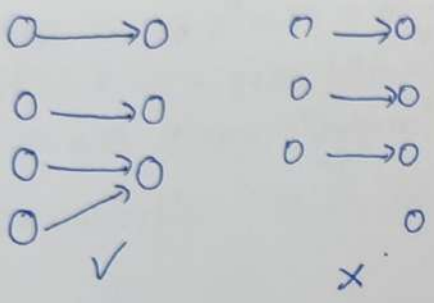
Injective (one to one) functions 单射



Properties of injective functions:

$a_1 \neq a_2$ then $f(a_1) \neq f(a_2) \Leftrightarrow f(a_1) = f(a_2)$ then $a_1 = a_2$

Surjective (onto) functions 满射

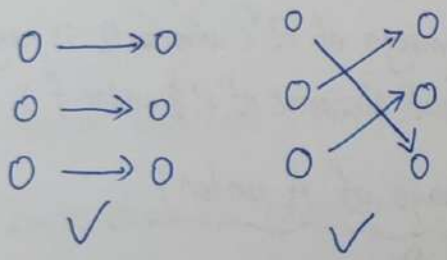


every $b \in B$ there exists a $a \in A$ that $b = f(a)$.

证明一定从定义出发如 $f: \mathbb{Z}^+ \to \mathbb{Z}^+$ $f(x) = x^2$ (X).

$\forall y \in \mathbb{Z}^+, y = f(x) = x^2 \Rightarrow x = \pm\sqrt{y} \notin \mathbb{Z}^+$ e.g. $y = 3$, $x = \pm\sqrt{3}$ X.

Bijective functions 双射 (surjective & injective).



Inverse functions 反函数 $f^{-1}: B \to A$.

$a = f^{-1}(b)$

Theorem: A function is invertible if and only if it's a bijection

Composition of functions.

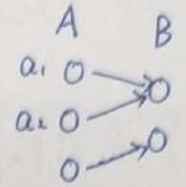
$f: A \to B$ $g: B \to C \Rightarrow (g \circ f)(x) = g(f(x))$

$A \to B$, $f^{-1} : B \to A$

$f \circ f^{-1} = f(f^{-1}(b)) = b$

$f^{-1} \circ f = f^{-1}(f(a)) = a$

The pigeonhole principle PHP (鸽笼定理).



A, B finite (有限的) sets and $f: A \to B$.

$A = \{a_1, a_2, \cdots a_n\}$

Theorem: if $|A| > |B|$ then at least one value of f occurs more than once, there can be no one-to-one function.

Strong PHP: $f: A \to B$, A, B finite sets, if $|A| > k|B|$, there is a value of f occurs at least $(k+1)$ times.

Logic Supplement:

- Parse tree 解析树: a binary tree to represent how the formula is constructed.
- Truth tables (Look at lecture 0 CS70) & Truth Values.

An interpretation I is a function than $I(p_i) \in \{0, 1\}$

if $I(p_i) = 1$ then p_i is called true under the interpretation I

false

$I(p)$	$I(\neg P)$
1	0
0	1

negation

$I(p)$	$I(Q)$	$I(P \wedge Q)$	$I(P \vee Q)$	$I(\neg P \vee Q)$ III $P \to Q$
1	1	1	1	1
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

conjunction　disconjunction　implication

$(P \Leftrightarrow Q) \equiv [(P \Rightarrow Q) \wedge (Q \Rightarrow P)] \equiv [(P = T \,\&\, Q = T) \text{ or } (P = F \,\&\, Q = F)]$

↑ biconditional

$P \vee \neg P \equiv T$, $\neg\neg P \equiv P$, $P \wedge \neg P \equiv \bot$

$\neg \bot \equiv T$, $\neg T \equiv \bot$ ↘ Complement laws.

Define: $T = (p_1 \vee \neg p_1)$ tautology.

$\bot = (p_1 \wedge \neg p_1)$ contradiction.

$\Gamma \models P$ (if $I(Q) = 1$ for all $Q \in \Gamma$, then $I(P) = 1$) P is a semantic consequence of Γ / P follows from Γ

"≡" is an equivalence relation:

- reflective : $P \equiv P$
- transitive : $P \equiv Q, Q \equiv R \Rightarrow P \equiv R$
- symmetric : $P \equiv Q \Rightarrow Q \equiv P$

Logical equivalence: P and Q are called equivalent if $I(P) = I(Q)$ which denoted by $P \equiv Q$

$\forall P_1, P_2 \; \exists \{ P_1 \wedge P_2 \}$ * 这里 set 只包含一个元素 T / F (对应 $P_1 \wedge P_2$)

$\{ P_1 \wedge P_2 \} \models P_1 \vee P_2$

Proof: if $I(P_1 \wedge P_2) = 1$, then $I(P_1) = 1, I(P_2) = 1$

$\Rightarrow I(P_1 \vee P_2) = 1$ ✓

*

$[\{P\} \models Q] \equiv (P \Rightarrow Q) \equiv \neg(P \wedge \neg Q) \equiv (\neg P \vee Q)$

"∨" - or "∧" - and "⟶" if ... then ...

Example: $\forall x (Student(x) \rightarrow \exists y (Younger(x, y) \wedge Instructor(y)))$

general domain $D = \{Alice, Bob, Carol\}$

if Alice, and Bob are Students, then predicate Symbol Student with Set {Alice, Bob}

age: Alice > Bob > Carol, then interpret Younger with the set: {(Alice, Bob), (Bob, Carol), (Alice, Carol)}

* A predicate becomes a proposition when the variables are substitued with specific values

neither ~~true~~ nor false

* The domain of a predicate is the set of all values that may be substituted for valuable

nbinatorics 组合数学

When the number of ways of doing A and B are independent, then the number of ways to do A and B is $n(A) \times n(B)$

General: $|A_1 \times A_2 \times \cdots \times A_m| = |A_1| \cdot |A_2| \cdots \cdot |A_m|$ - Product Rule
$= \Pi_{i \in m} |A_i|$

$|A \cup B| = |A| + |B|$ — Sum Rule (where A and B are disjoint sets)

General: $|A_1 \cup A_2 \cup \cdots A_m| = |A_1| + \cdots + |A_m|$, $A_i \cap A_j = \emptyset (i \neq j)$
$= \sum_{i \in m} |A_i|$

* Principle of inclusion-exclusion: $|A \cup B| = |A| + |B| - |A \cap B|$
* Division principle: (example 24 ears how many people → 24/2 = 12)
* Factorial function 阶乘函数 $n! = n \times (n-1) \times \cdots \times 1$

1. Permutations 排序

- Without repetition: select one object one time → $\underbrace{n \times (n-1) \times \cdots (n-r+1)}_{r个} = \frac{n!}{(n-r)!}$
- with repetition: select the same objects multiple times → $\underbrace{n \times n \times n \cdots n}_{r个} = n^r$
- of identical objects: some objects to be arranged are identical 相同的

$$\frac{n!}{n_1!\, n_2! \cdots n_k!}$$

type 1 相同个数 ... type k 相同个数

2. Combinations 组合

二项式系数 binomial coefficient $\binom{n}{r} = \frac{n!}{r!(n-r)!} = C_n^r$

with repetition: C_{n+r-1}^r

without repetition: C_n^r — Proof

n个元素种类能重复选择同种种类选出r个 ⟺ n个盒子有r个球放入可以重复放

⟺ ○ | ○ ○ ○ ○ ⋯ ○ (插入n-1个板子, r个球)

⟺ 最终有 n-1+r 个位置 ○ - - - ○ (r个, n-1+r)

放入r个球(无区分)

⟺ 选编号从1,2,⋯ n-1+r号的人 - - - (r个位置)

⟺ C_{n+r-1}^r

Probability Theory

Experiments
- Deterministic experiment.
- Non-deterministic experiment.

↓

Random Process (Any action with unpredictable outcomes).

Sample Space Ω (all possible outcomes of a random process or experiment).

Event: a specific outcome, a subset of the sample space Ω

probability $P(X=x)$: the probability of outcome x occuring (X is a random variable)

special events
- null events $\emptyset$ never occurs.
- entire events Ω, may occurs.

discrete random variables

$X=\{x_1, x_2, x_3, \cdots x_n\}$ if $P(X=x_1)=P(X=x_2)=\cdots P(X=x_n)$

then $P(X=x_1)=\frac{1}{n}=\cdots P(X=x_n)$

If the probability of an event is 0, the event is impossible.

1 ~ certain

discrete random variable

X' is the complement of an event X in Ω $P(X')=1-P(X)$.

$P(\emptyset)=0$ $P(\Omega)=1$ $P(X)=\sum_{x\in X} P(x)$.

Compound events
- independent events 扔了7次硬币都是头朝上 $(\frac{1}{2})\times\underbrace{(\frac{1}{2})\times\cdots\times(\frac{1}{2})}_{7个\frac{1}{2}}$
- Mutually exclusive events (A) (B) $P(A\cup B)=P(A)+P(B)$.
- Mutually inclusive events (A)(B) $P(A\cup B)=P(A)+P(B)-P(A\cap B)$
- Dependent events 扑克牌选2张，第二次选时第一次选到牌不放回去

Simple events 扔硬币 $\frac{1}{2}$

Conditional Probability $P(X|Y) = \frac{P(X \wedge Y)}{P(Y)}$

will happen ↗ ↑ Already happen / Given the condition

eg: 打2个骰子，和为7，出现至少一个3的概率

↑ P(Y)　　↑ P(X)

$$P(X|Y) = \frac{P(X \wedge Y)}{P(Y)} = \frac{\frac{2}{36}}{\frac{6}{36}} = \frac{1}{3}$$

↑ 在和为7出现至少一个3概率

7 = 1+6 = 2+5 = 3+4 = 4+3 = 5+2 = 6+1　$P(Y) = \frac{6}{36}$　$P(X) = \frac{11}{36}$

$P(X \wedge Y) = \frac{2}{36}$　(3,4)/(4,3)

理解：1. $P(Y) = P(X_1 \wedge Y) + \cdots P(X_n \wedge Y)$.

2. $P(X \wedge Y) = P(X \wedge Y | \Omega)$. 在全样本上满足X,Y下事件概率.

$$E[X] = \sum_{x \in X} x \cdot P_X(x)$$

$$E[XY] = E[X]E[Y] + \mathrm{cov}(X,Y).$$

$E(ag(X) + bf(Y)) = aE(g(X)) + bE(f(Y))$ two random variables.

CS 70 Discrete Mathematics and Probability Theory
Spring 2022 Course Notes Note 5

1 Graph Theory: An Introduction

One of the fundamental ideas in computer science is the notion of *abstraction*: capturing the essence or the core of some complex situation by a simple model. Some of the largest and most complex entities we might deal with include the internet, the brain, maps, and social networks. In each case, there is an underlying "network" or *graph* that captures the important features that help us understand these entities more deeply. In the case of the internet, this network or graph specifies how web pages link to one another. In the case of the brain, it is the interconnection network between neurons. We can describe these ideas in the beautiful framework of *graph theory*, which is the subject of this lecture.

Remarkably, graph theory has its origins in a simple evening pastime of the residents of Königsberg, Prussia (nowadays Kaliningrad, Russia) a few centuries ago. Through their city ran the Pregel river, depicted on the left in Figure 1 below, separating Königsberg into two banks A and D and islands B and C. Connecting the islands to the mainland were seven bridges. As the residents of the city took their evening walks, many would try to solve the challenge of picking a route that would cross each of the seven bridges precisely once and return to the starting point. 详见 → section 2

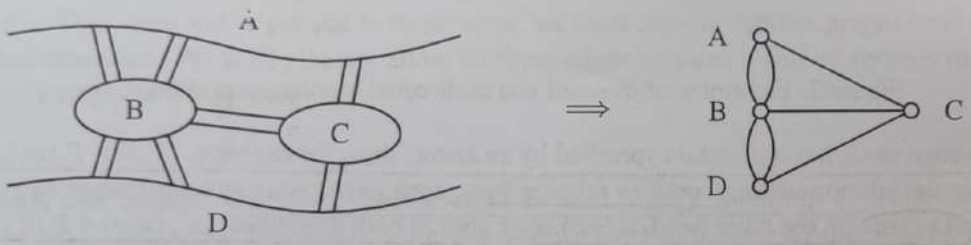


Figure 1: (Left) The city of Königsberg. (Right) The (multi-)graph modeling the bridge connections in Königsberg.

In 1736, the brilliant mathematician Leonhard Euler proved this task to be impossible. How did he do it? The key is to realize that for the purpose of choosing such a route, Figure 1a can be replaced with Figure 1b, where each land mass A, B, C, and D is replaced by a small circle, and each bridge by a line segment. With this abstraction in place, the task of choosing a route can be restated as follows: trace through all the line segments and return to the starting point without lifting the pen, and without traversing any line segment more than once. The proof of impossibility is simple. Under these tracing rules, the pen must enter each small circle as many times as it exits it, and therefore the number of line segments incident to that circle must be even. But in Figure 1b, each circle has an odd number of line segments incident to it, so it is impossible to carry out such a tracing. Actually Euler did more. He gave a precise condition under which the tracing can be carried out. For this reason, Euler is generally hailed as the inventor of graph theory.

1.1 Formal definitions

Formally, a (undirected) graph is defined by a set of vertices V and a set of edges E. The vertices correspond to the little circles in Figure 1 above, and the edges correspond to the line segments between the vertices. In Figure 1, $V = \{A, B, C, D\}$ and $E = \{\{A,B\}, \{A,B\}, \{A,C\}, \{B,C\}, \{B,D\}, \{B,D\}, \{C,D\}\}$. However, note that here E is a multiset (a set where an element can appear multiple times). This is because in the Königsberg example there are multiple bridges between a pair of banks. We will generally not consider such a situation of multiple edges between a single pair of vertices, so in our definition, we require E to be a set, not a multi-set. What this means is that between any pair of vertices there is either 0 or 1 edge. If there are multiple edges between a pair of vertices, then we collapse them into a single edge.

More generally, we can also define a directed graph. If an edge in an undirected graph represents a street, then an edge in a directed graph represents a one-way street. To make this formal, let V be a set denoting the vertices of a graph G. For example, we can have $V = \{1,2,3,4\}$. Then, the set of (directed) edges E is a subset of $V \times V$, i.e. $E \subseteq V \times V$. (Recall here that $U \times V$ denotes the Cartesian product of sets U and V, defined as $U \times V = \{(u,v) : u \in U \text{ and } v \in V\}$.) Continuing with our example, let $E = \{(1,2),(1,3),(1,4)\}$. Then, the corresponding graph is given by G_1 below.

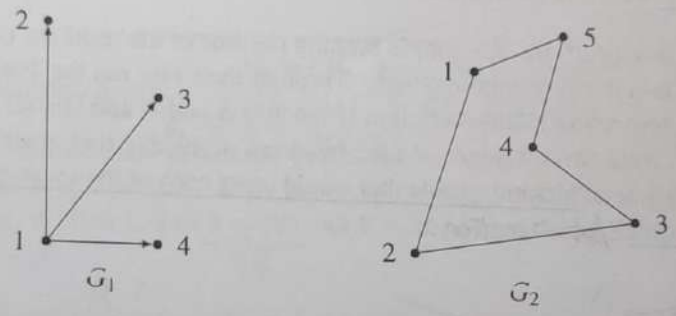


Figure 2: Examples of directed and undirected graphs, respectively.

Note that each edge in G_1 has a *direction* specified by an arrow; thus, for example, $(1,2) \in E$ but $(2,1) \notin E$. Such graphs are useful in modeling one-way relationships, such as one-way streets between two locations, and are called *directed*. On the other hand, if each edge goes in both directions, i.e., $(u,v) \in E$ iff $(v,u) \in E$, then we call the graph *undirected*. For undirected graphs we drop the ordered pair notation for edges, and simply denote the edge between u and v by the set $\{u,v\}$. Undirected graphs model relationships such as two-way streets between locations naturally, and an example is given by G_2 above. For simplicity, we omit the arrowheads when drawing edges in undirected graphs. We conclude that a graph is thus formally specified as an ordered pair $G = (V,E)$, where V is the vertex set and E is the edge set.

Concept check! What are the vertex and edge sets V and E for graph G_2?

Let us continue our discussion with a working example from *social networks*, an area in which graph theory plays a fundamental role. Suppose you wish to model a social network in which vertices correspond to people, and edges correspond to the following relationship between people: We say Alex *recognizes* Bridget if Alex knows who Bridget is, but Bridget does not know who Alex is. If, on the other hand, Alex knows Bridget and Bridget knows Alex, then we say they *know each other*.

Concept check! Suppose first that an edge between two people (say) Alex and Bridget means that Alex recognizes Bridget; would you use a directed or undirected graph for this? How about if an edge instead means Alex and Bridget know each other? (Answer: directed and undirected, respectively.)

Moving on with our example, we say that edge $e = \{u, v\}$ (or $e = (u, v)$) is *incident* on vertices u and v, and that u and v are *neighbors* or *adjacent*. If G is undirected, then the *degree* of vertex $u \in V$ is the number of edges incident to u, i.e., $\text{degree}(u) = |\{v \in V : \{u, v\} \in E\}|$. A vertex u whose degree is 0 is called an *isolated* vertex, since there is no edge which connects u to the rest of the graph.

Concept check! What does the degree of a vertex represent in our *undirected* social network in which an edge $\{u, v\}$ means u and v know each other? How should we interpret an isolated vertex?

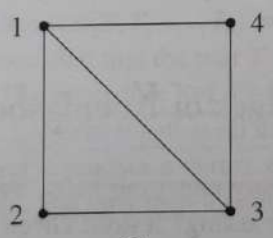
A directed graph, on the other hand, has two different notions of degree due to the directions on the edges. Specifically, the *in-degree* of a vertex u is the number of edges from other vertices to u, and the *out-degree* of u is the number of edges from u to other vertices.

Concept check! What do the in-degree and out-degree of a vertex represent in our *directed* social network in which an edge (u, v) means u recognizes v?

Finally, our definition of a graph thus far allows edges of the form $\{u, u\}$ (or (u, u)), i.e., a *self-loop*. In our social network, however, this gives us no interesting information (it means that person A recognizes him/herself!). Thus, here and in general in these notes, we shall assume that our graphs have no self-loops, unless stated otherwise. We shall also not allow multiple edges between a pair of vertices (unlike the case of the Seven Bridges of Königsberg).

Paths, walks, and cycles. Let $G = (V, E)$ be an undirected graph. A *path* in G is a sequence of edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-2}, v_{n-1}\}, \{v_{n-1}, v_n\}$. In this case we say that there is a path *between* v_1 *and* v_n. For example, suppose the graph G_3 below models a residential neighborhood in which each vertex corresponds to a house, and two houses u and v are neighbors if there exists a direct road from u to v.

$v_1, v_2, \cdots v_n$ are distinct.



G_3

Concept check! What is the shortest path from house 1 to house 3 in G_3? How about the longest path, assuming no house is visited twice? $1 \to 3$; $1 \to 2 \to 3 \to 4 \to 1 \to 3$ (X).

$1 \to 2 \to 3$ / $1 \to 4 \to 3$.

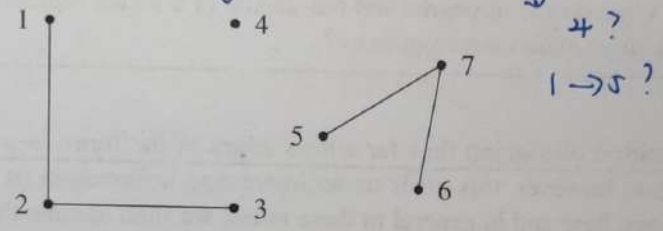
Length of paths is equal to the edges that $V_1 \to V_2 \to \cdots V_n$ which is $(n-1)$.

Usually, we assume a path is *simple*, meaning $v_1, \ldots, v_n$ are distinct. This makes complete sense in our housing example G_3; if you wanted drive from house 1 to 3 via house 2, why would you visit house 2 more than once? A *cycle* (or *circuit*) is a sequence of edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-2}, v_{n-1}\}, \{v_{n-1}, v_n\}, \{v_n, v_1\}$, where $v_1, \ldots, v_n$ are distinct.

Concept check! Give a cycle starting at house 1 in G_3. $1 \to 2 \to 3 \to 4 \to 1$

Suppose now that your aim is not to go from 1 to 3 as quickly as possible, but to take a leisurely stroll from 1 to 3 via the sequence $\{1,2\}, \{2,1\}, \{1,4\}, \{4,3\}$. A sequence of edges with repeated vertices, such as this one, is called a *walk* from 1 to 3. Analogous to the relationship between paths and cycles, a *tour* is a walk which starts and ends at the same vertex. For example, $\{1,2\}, \{2,3\}, \{3,1\}$ is a tour.

Connectivity. Much of what we discuss in this note revolves around the notion of connectivity. A graph is said to be *connected* if there is a path between any two distinct vertices. For example, our residential network G_3 above is connected, since one can drive from any house to any other house via *some* sequence of direct roads. On the other hand, the network below is *disconnected*.



4 ?

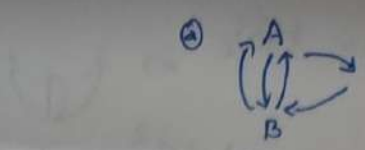
1 → 5 ?

Concept check! What would a disconnected vertex represent in our residential network? Why would you not want to design a neighborhood this way?

Note that *any* graph (even a disconnected one) always consists of a collection of *connected components*, i.e., sets $V_1, \ldots, V_k$ of vertices, such that all vertices in a set V_i are connected. For example, the graph above is not connected, but nevertheless consists of three connected components: $V_1 = \{1,2,3\}$, $V_2 = \{4\}$, and $V_3 = \{5,6,7\}$.

2 Revisiting the Seven Bridges of Koenigsberg: Eulerian Tours

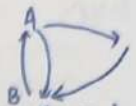
With a formal underpinning in graph theory under our belts, we are ready to revisit the Seven Bridges of Königsberg. What exactly is this problem asking? It says: Given a graph G (in this case, G is an abstraction of Königsberg), is there a walk in G that uses each edge exactly once? We call any such walk in a graph an *Eulerian walk*. (In contrast, by definition a walk can normally visit each edge or vertex as many times as desired.) Moreover, if an Eulerian walk is closed, i.e., it ends at its starting point, then it is called an *Eulerian tour*. Thus, the Seven Bridges of Königsberg asks: Given a graph G, does it have an Eulerian tour? We now give a precise characterization of this in terms of simpler properties of the graph G. For this, define an *even degree* graph as a graph in which all vertices have even degree.



这是 Eulerian tour : A→A

Theorem 5.1 (Euler's Theorem (1736)). *An undirected graph $G = (V, E)$ has an Eulerian tour iff G is even degree, and connected (except possibly for isolated vertices).*

$G = (V, E)$



① 逆是 Eulerian walk A→B.

Proof. To prove this, we must establish two directions: if, and only if.

Only if. We give a direct proof for the forward direction, i.e., if G has an Eulerian tour, then it is connected and has even degree. Assume that G has an Eulerian tour. This means every vertex that has an edge adjacent to it (i.e., every non-isolated vertex) must lie on the tour, and is therefore connected with all other vertices on the tour. This proves that the graph is connected (except for isolated vertices).

$V_1 \to V_2 \to V_3 \to \cdots \to V_n$ ⇒ 对 $\forall V_i$, $(V_{i-1} \to V_i), (V_i \to V_{i+1})$ $2 \le i \le n-1$. exception

Next, we prove that each vertex has even degree by showing that all edges incident to a vertex can be paired up. Notice that every time the tour enters a vertex along an edge it exits along a different edge. We can pair these two edges up (they are never again traversed in the tour). The only exception is the start vertex, where the first edge leaving it cannot be paired in this way. But notice that by definition, the tour necessarily ends at the start vertex. Therefore, we can pair the first edge with the last edge entering the start vertex. So all edges adjacent to any vertex of the tour can be paired up, and therefore each vertex has even degree.

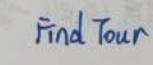
$V_{n-1} \to V_n$

$V_n \to V_1$ ✓

If. We give a recursive (递归的) algorithm for finding an Eulerian tour, and prove by induction that it correctly outputs an Eulerian tour.

We start with a useful subroutine (子程序), FINDTOUR(G, s), which finds a tour (not necessarily Eulerian) in G. FINDTOUR is very simple: it just starts walking from a vertex $s \in V$, at each step choosing any untraversed edge incident to the current vertex, until it gets stuck because there is no more adjacent untraversed edge. We now prove that FINDTOUR must in fact get stuck at the original vertex s.

Claim: FINDTOUR(G, s) must get stuck at s.



Find Tour 可找到 $s \to A \to s$，但这不是 Eulerian

Proof of claim: An easy proof by induction on the length of the walk shows that when FINDTOUR enters any vertex $v \neq s$, it will have traversed an odd number of edges incident to v, while when it enters s it will have traversed an even number of edges incident to s. Since every vertex in G has even degree, this means every time it enters $v \neq s$, there is at least one untraversed edge incident to v, and therefore the walk cannot get stuck. So the only vertex it can get stuck at is s. The formal proof is left as an exercise. □

The algorithm FINDTOUR(G, s) returns the tour it has traveled when it gets stuck at s. Note that while FINDTOUR(G, s) always succeeds in finding a tour, it does not always return an Eulerian tour.

We now give a recursive algorithm EULER(G, s) that outputs an Eulerian tour starting and ending at s. EULER(G, s) invokes (调用) another subroutine SPLICE$(T, T_1, \ldots, T_k)$ which takes as input a number of edge disjoint tours $T, T_1, \ldots, T_k$ ($k \geq 1$), with the condition that the tour T intersects each of the tours $T_1, \ldots, T_k$ (i.e., T shares a vertex with each of the T_i's). The procedure SPLICE$(T, T_1, \ldots, T_k)$ outputs a single tour T' that traverses all the edges in $T, T_1, \ldots, T_k$, i.e., it splices together all the tours. The combined tour T' is obtained by traversing the edges of T, and whenever it reaches a vertex s_i that intersects another tour T_i, it takes a detour to traverse T_i from s_i back to s_i again, and only then it continues traversing T.

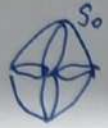
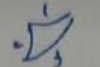
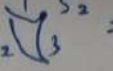
The algorithm EULER(G, s) is given as follows:

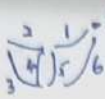
```
Function Euler(G, s)
    T = FindTour(G, s)
    Let G_1,...,G_k be the connected components
        when the edges in T are removed from G
    Let s_i be the first vertex in T that intersects G_i
```

⇒

```
def fuction_Euler(G,S):
    T = Find Tour(G,S)
    Let G' are the components
        when T are removed from G.
    Let s' are the first vertex that
    T intersects with G'
    ~~function Euler(G',s')~~
    ~~list.append(T)~~
    if G',s' exist, T=Split(T, fuction_Euler
    (G',s')
    return T
```

我觉得 i 这个 k 代表迭代次数直接代码见右侧

示例：  $T_0 =$  ⇒  $T_1 =$  ⇒  $= T_2$

$T_0 = Split(T_0, T_1) \Leftarrow$  $\Leftarrow T_1 = Split(T_1, T_2)$

Euler Tour

```
    Output Splice(T, Euler(G_1, s_1), ..., Euler(G_k, s_k))
end EULER
```

We prove by induction on the size of G that EULER(G,s) outputs an Eulerian Tour in G. The same proof works regardless of whether we think of size as number of vertices or number of edges. For concreteness, here we use number of edges m of G.

Base case: $m = 0$, which means G is empty (it has no edges), so there is no tour to find.

(Strong Induction) Induction hypothesis: EULER(G,s) outputs an Eulerian Tour in G for any even degree, connected graph with at most $m \geq 0$ edges.

$G - T \Rightarrow G'$ has m_i $cm_i \leq m-1$ edges, could find Euler tour

Induction step: Suppose G has $m+1$ edges. Recall that $T =$ FINDTOUR(G,s) is a tour, and therefore has even degree at every vertex. When we remove the edges of T from G, we are therefore left with an even degree graph with less than m edges, but it might be disconnected. Let $G_1, \ldots, G_k$ be the connected components. Each such connected component has even degree and is connected (up to isolated vertices). Moreover, T intersects each of the G_i, and as we traverse T there is a first vertex where it intersects G_i. Call it s_i. By the induction hypothesis EULER(G_i, s) outputs an Eulerian tour of G_i. Now by the definition of SPLICE, it splices the individual tours together into one large tour whose union is all the edges of G, hence an Eulerian tour. □

Concept check! Why does Theorem 5.1 imply the answer to the Seven Bridges of Königsberg is no?

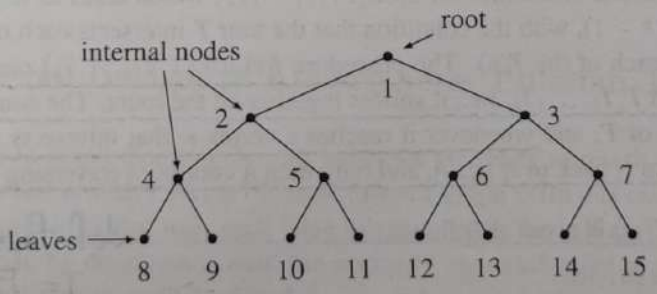
这里归纳法如何环环相扣，从递归角度说明一定能找到了对于(G,S)已知的边长 m 从而延伸到所有边长的情况

3 Planarity, Euler's Formula, Coloring.

$\forall G$ 一定能被拆分成已知的 G 的源头！

Trees

We need to discuss trees briefly here, though we will discuss them more later. A graph is a *tree* if it is connected and acyclic (contains no cycles). There are many other equivalent definitions. For example, a tree is a connected graph where the number of vertices is one more than the number of edges. Or, a tree is a connected graph such that if you delete any edge it becomes disconnected.

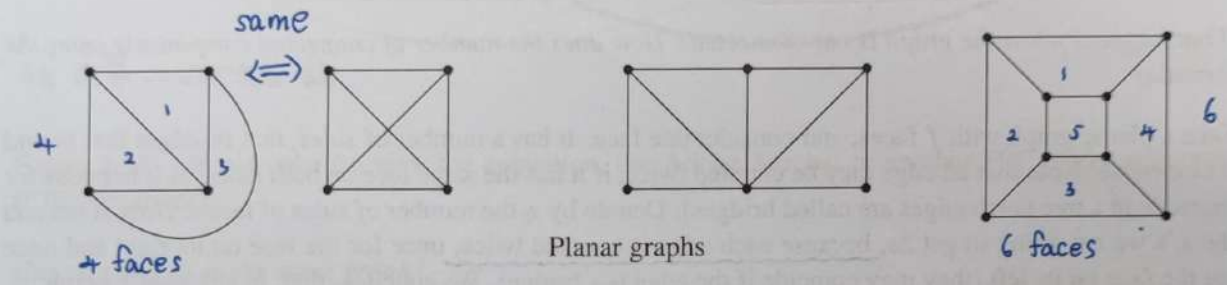


Planar Graphs 平面图

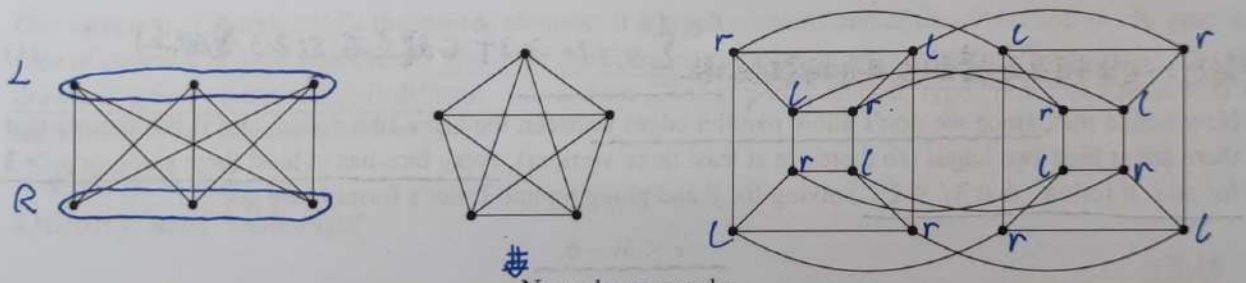
A graph is *planar* if it can be drawn on the plane without crossings. For example, the first four graphs shown below are planar. Notice that the first and second graphs are the same, but drawn differently. Even though the second drawing has crossings, the graph is still considered planar since it is possible to draw it without crossings.

The other three graphs are not planar. The first one of them is the infamous "three houses-three wells graph," also called $K_{3,3}$. (This notation says there are two sets of vertices, each of size three and all edges between the two sets of vertices are present.) The second is the "complete" graph (every edge is present) with five nodes, or K_5. The third is the four-dimensional cube. We shall soon see how to prove that all three graphs are non-planar.

A useful concept is the notion of a *bipartite* graph, $G = (V, E)$ which is a graph where the vertices are split into two groups and edges only go between groups. More formally, we have $V = L \cup R$ and $E \subseteq L \times R$. Clearly, $K_{3,3}$ is bipartite. Moreover, the four dimensional cube is bipartite: this follows from the fact that every cycle in the cube has even length. We will show the equivalence a bit later.



Planar graphs



Non-planar graphs

When a planar graph is drawn on the plane, one can distinguish, besides its vertices (their number will be denoted v here) and edges (their number is e), the *faces* of the graph (more precisely, of the drawing). The faces are the regions into which the graph subdivides the plane. One of them is infinite, and the others are finite. The number of faces is denoted f. For example, for the first graph shown $f = 4$, and for the fourth (the cube) $f = 6$.

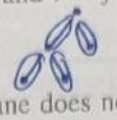
One basic and important fact about planar graphs is *Euler's formula*, $v + f = e + 2$ (check it for the graphs above). It has an interesting story. The ancient Greeks knew that this formula held for all polyhedra (check it for the cube, the tetrahedron, and the octahedron, for example), but could not prove it. How do you do induction on polyhedra? How do you apply the induction hypothesis? What is a polyhedron minus a vertex, or an edge? In the 18th century Euler realized that this is an instance of the inability to prove a theorem by

induction *because it is too weak*, something that we saw time and again when we were studying induction. To prove the theorem, one has to generalize polyhedra. And the right generalization is *planar graphs*.

Can you see why planar graphs generalize polyhedra? Why are all polyhedra (without "holes") planar graphs?

Theorem 5.2. *(Euler's formula) For every connected planar graph, $v + f = e + 2$*

Proof. By induction on e. It certainly holds when $e = 0$, and $v = f = 1$. Now take any connected planar graph. Two cases:



- If it is a tree, then $f = 1$ (drawing a tree on the plane does not subdivide the plane), and $e = v - 1$ (check homework).
- If it is not a tree, find a cycle and delete any edge of the cycle. This amounts to reducing both e and f by one. By induction the formula is true in the smaller graph, and so it must be true in the original one. □

What happens when the graph is not connected? How does the number of connected components enter the formula?

Take a planar graph with f faces, and consider one face. It has a number of *sides*, that is, edges that bound it clockwise. Note that an edge may be counted twice, if it has the same face on both sides, as it happens for example in a tree (such edges are called bridges). Denote by s_i the number of sides of face i. Now, if we add the s_i's we are going to get $2e$, because each edge is counted twice, once for the face on its right and once for the face on its left (they may coincide if the edge is a bridge). We conclude that, in any planar graph,

$$\sum_{i=1}^{f} s_i = 2e \tag{1}$$

Now notice that, since we don't allow parallel edges between the same two nodes, and if we assume that there are at least two edges (so there are at least three vertices), every face has at least three sides, or $s_i \geq 3$ for all i. It follows that $3f \leq 2e$. Solving for f and plugging into Euler's formula we get

$$e \leq 3v - 6.$$

This is an important fact. First it tells us that planar graphs are *sparse*, they cannot have too many edges. A 1,000-vertex connected graph can have anywhere between a thousand and half a million edges. This inequality tells us that for planar graphs the range is very small, between 999 and 2,994.

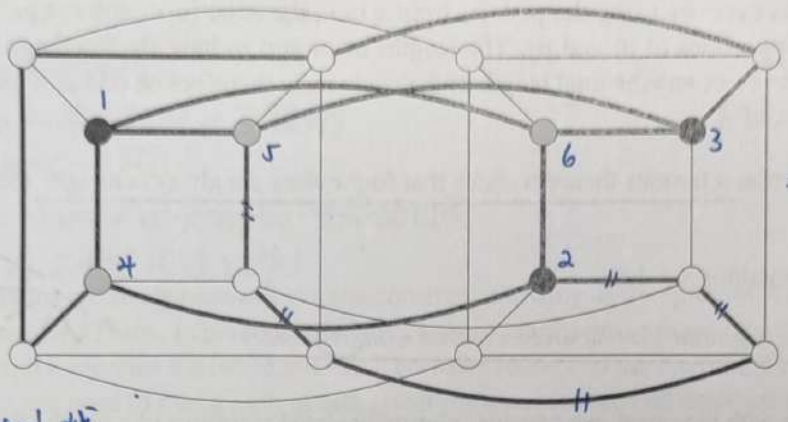
It also tells us that K_5 is not planar: Just notice that it has five vertices and ten edges.

$K_{3,3}$ has $v = 6, e = 9$ so it passes the planarity test with flying colors. We must think a little harder to show that $K_{3,3}$ is non-planar. Notice that, if we had drawn it on the plane, there would be no triangles. Because a triangle means that two wells or two houses are connected together, which is false. So, Equation (1) now gives us $4f \leq 2e$, and solving for f and plugging into Euler's formula, $e \leq 2v - 4$, which shows that $K_{3,3}$ is non-planar.

So, we have established that K_5 and $K_{3,3}$ are both non-planar. There is something deeper going on: In some sense, these are *the only non-planar graphs*. This is made precise in the following famous result, due to the Polish mathematician Kuratowski (this is what "K" stands for).

Theorem 5.3. *A graph is non-planar if and only if it contains K_5 or $K_{3,3}$.*

"Contains" here means that one can identify nodes in the graph (five in the case of K_5, six in the case of $K_{3,3}$) which are connected as the corresponding graph through paths (possibly single edges), and such that no two of these paths share no vertex. For example, the 4-cube shown below is non-planar, because it contains $K_{3,3}$, as shown.



打"="线也算 connected 线

Figure 3: $K_{3,3}$ in a 4-cube (to view the animation, use Adobe Acrobat or another PDF reader supporting PDF animations)

Can you find K_5 in the same graph?

One direction of Kuratowski's theorem is obvious: If a graph contains one of these two non-planar graphs, then of course it is itself non-planar. The other direction, namely that in the absence of these graphs we can draw any graph on the plane, is difficult. For a short proof you may want to type "proof of Kuratowski's theorem" in your favorite search engine.

对偶性与着色

Duality and Coloring

八面体
There is an interesting *duality* between planar graphs. For example, the Greeks knew that the octahedron and the cube 立方体 are "dual" to each other, in the sense that the faces of one can be put in correspondence with the vertices of the other (think about it). The tetrahedron 四面体 is self-dual. 十二面体 And the dodecahedron and the icosahedron 二十面体 (look for images in the web if you don't know these pretty things) are also dual to one another.

What does this mean? Take a planar graph G, and assume it has no bridges and no degree-two nodes. Draw a new graph G^*: Start by placing a node on each face of G. Then draw an edge between two faces if they touch at an edge — draw the new edge so that it crosses that edge. The result is G^*, also a planar graph. Notice now that, if you construct the dual of G^*, it is the original graph: $(G^*)^* = G$.

Duality is a convenient consideration when thinking about planar graphs. Also, it tells us that ("coloring a political map so that no two countries who share a border have the same color" is the same problem as "coloring the vertices of a planar graph (the dual of the political map) so that no two adjacent vertices have the same color.")


矛盾！ ✓ cycle 一定是：• × • × --- • 2的倍数

Returning for a moment to the concept of a bipartite graph in the context of coloring, one sees that a two colorable graph is equivalent to a bipartite graph as the coloring splits the vertices into two sets where edges only connect vertices in the two sets(colors). Here, with this observation, we can see that a graph that does not contain any odd length cycles is bipartite as we can greedily color any such graph with two colors as follows for a connected graph. Color one vertex, say x red, color its neighbors blue, and then continue by coloring their uncolored neighbors red, and continue alternatively assigning colors to uncolored neighbors. If this does not work, then some edge (u,v) must connect two vertices of the same color, in which case we can identify an odd cycle by using the path p_1 from x to u, the edge (u,v) and the path p_2 from v to x and removing common portions of p_1 and p_2. The lengths of p_1 and p_2 have the same parity (even or odd) as u and v have the same color and the total length of the cycle must therefore be odd as it includes the additional edge (u,v).

判定：

Back to planar graphs, a famous theorem states that four colors are always enough! (Search for "four color theorem".)

We shall prove something weaker:

1 → 2
2 → 1

Theorem 5.4. *Every planar graph can be colored with five colors.*

Before proceeding with the proof, we consider alternative legal colorings of a graph. We first take the subset of vertices of two colors, say 1 and 2, and compute the connected components. We note that we can switch the two colors in a single connected component; consider any edge that is not in the connected component is clearly fine, any edge in the connected component has both endpoints switched which is also fine.

Proof. Induction on v. The base case is not worth talking about, so we go directly to the inductive step. Let G be a planar graph. I claim there is a node of degree five or less. In proof, consider the inequality $e \leq 3v-6$. If all vertices had degree six or more, then e would be at least $3v$.

→ 一定存在否则 $e = \frac{6V}{2} = 3V > 3V-6$

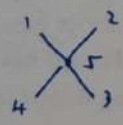
So, consider a node u of degree five or less. If it has degree four or less, we are done: Remove u, color the remaining graph with 5 colors (by induction), and then put u back in and color it by a color that is missing from its neighbors.

So, u must have 5 vertices, and in the coloring of $G-u$ they all got different colors. Look at them clockwise around u, and call them u_1, u_2, u_3, u_4, u_5, and their colors $1, 2, 3, 4, 5$.

Now try to change the color of u_2 to color 4 by determining the connected component containing u_2 and consisting of vertices that are colored 2 or 4. If u_4 is in the connected component, switching 2 and 4 is not helpful. But, we do know that there is a path between u_2 and u_4 colored 2 and 4.

Similarly, we can try to change the color of u_1 to 3, and this will succeed unless there is a path between u_1 to u_3 colored 1 and 3.

If both of these attempts fail, then we get two paths: one from u_1 to u_3 colored 1 and 3, and the other from u_2 to u_4 colored 2 and 4. But planarity says that these two paths must intersect at some vertex. What is the color of this vertex?

引理：Six color theorem
$e \leq 3V-6$ $(V>2)$

* average degree $\# = \frac{2e}{V} \leq \frac{2(3V-6)}{V} = 6 - \frac{12}{V}$

对一个点来说，附带的边才有邻接的点，而 1→2, 2→1 计为一种 共有 e 边 ⇒ 2e 个 total degree

对 V induction：1 ✓
$1 \leq n \leq k$ 成立时 在 k+1 上必有一个点 邻接边 ≤ average degree 得到）那么去掉这个点，剩余边 连接 ⇒ k 个点 成立必能用 6 种染而补上这个点(1,2,3,4,5)

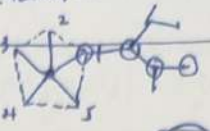
→ 保留各一种即可！

这里同样在5色定理用到 induction.

1. $n=1$ ✓

2. $1 \le n \le k$, ~~exist $V \le 5$~~ 成立时 考虑 $n=k+1$, exists $V \le 5$

($V_{min} \le 4$) 补其即可 $\exists V=5$ 时 ⇒ 先去掉 即退化成 $n=k$ 已知成立，再补上).



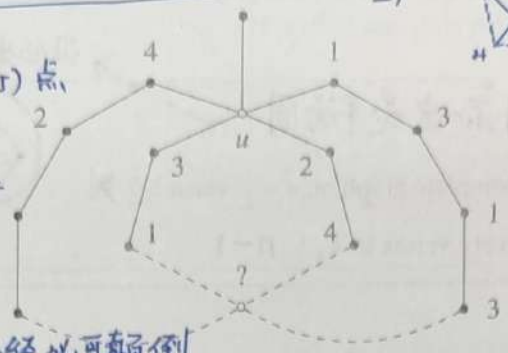
下面继续证明：① 若 (1, 2, 3, 4, 5) 点在 $n=k$ 涂色时用 ≤ 4 种色 ✓

② 若有 5 种，由于之前是用虚线连接相邻点必定颜色不同，现考虑 {1, 3}

I. {1, 3} 在外圈不相连此时颜色 {a, c}

则一定存在 a-c-a-c-a-… 染色路经必可颠倒

用 3 的 C 代替 1 的 a 达成 c-a-c-a 即可！



~~1 — 3 和 2 — 4 不能同时成立故一定有个能涂一样的等颜色~~

II. {1, 3} 相连 Important classes of graphs 重要的图类

由平面图可得 {2, 4} {2, 5} 不相连同理证毕！

As we have seen, graphs are an abstract and general construct allowing us to represent relationships between objects, such as houses and roads. In practice, certain classes of graphs prove especially useful. For example, imagine that our graph represents the interconnection between routers on the internet. To send a packet from one node to another, we need to find a path in this graph from our source and destination. Therefore, to be able to send a packet from any node to any other node, we only need the graph to be connected. A minimally connected graph is called a *tree*, and it is the most efficient graph (i.e., with minimum number of edges) we can use to connect any set of vertices.

$e = V-1$

But now suppose the connections between some routers are not too strong, so sometimes we can lose an edge between two vertices in the graph. If our graph is a tree, then removing an edge from it results in a disconnected graph, which means there are some vertices in the graph that we cannot reach. To avoid this bad case, we want some sort of redundancy or robustness in the graph connectivity. Clearly the most connected graph is the complete graph in which all nodes are connected to all other nodes. However, as we shall see below, the complete graph uses exponentially many edges, which makes it impractical for large-scale problems.

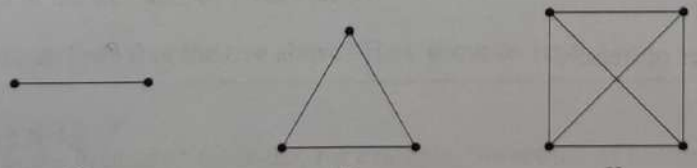
指数级的条边

There is also a nice family of graphs called the hypercube graphs, which combines the best of both worlds: they are robustly connected, but do not use too many edges. In this section, we study these three classes of graphs in more detail.

超立方图：强连接性 + 边数控制

4.1 Complete graphs 完全图

We start with the simplest class of graphs, the *complete* graphs. Why are such graphs called complete? Because they contain the *maximum* number of edges possible. In other words, in an undirected complete graph, every pair of (distinct) vertices u and v are connected by an edge $\{u, v\}$. For example, below we have complete graphs on $n = 2, 3, 4$ vertices, respectively.



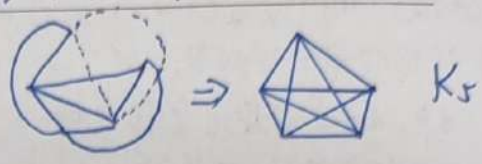
K_2 K_3 K_4

$e: C_n^2$

K_n

Here, the notation K_n denotes the *unique* complete graph on n vertices. Formally, we can write $K_n = (V, E)$ for $|V| = n$ and $E = \{\{v_i, v_j\} \mid v_i \neq v_j \text{ and } v_i, v_j \in V\}$.

Concept check! 不完全图不一定是平面图 → 没必要画成这样



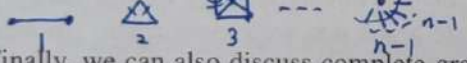
1. Can you draw K_5, the complete graph on $n = 5$ vertices? ✗
2. What is the degree of every vertex in K_n? $n-1$

Exercise. How many edges are there in K_n? (Answer: $n(n-1)/2$.) Verify that the K_5 you drew above has this many edges.

Next, let us return to the theme of connectivity. A complete graph is special in that each vertex is neighbors with every other vertex. Thus, such a graph is very "strongly connected" in that a large number of edges must be removed before we disconnect the graph into two components. Why might this be a good property to have (say) in a communications network, where vertices correspond to mainframes, and edges correspond to communications channels?

Concept check! What is the minimum number of edges which must be removed from K_n to obtain an isolated vertex?

K_2 K_3 K_4 ... K_{n-1} ⇒ $(n-1)$ vertex



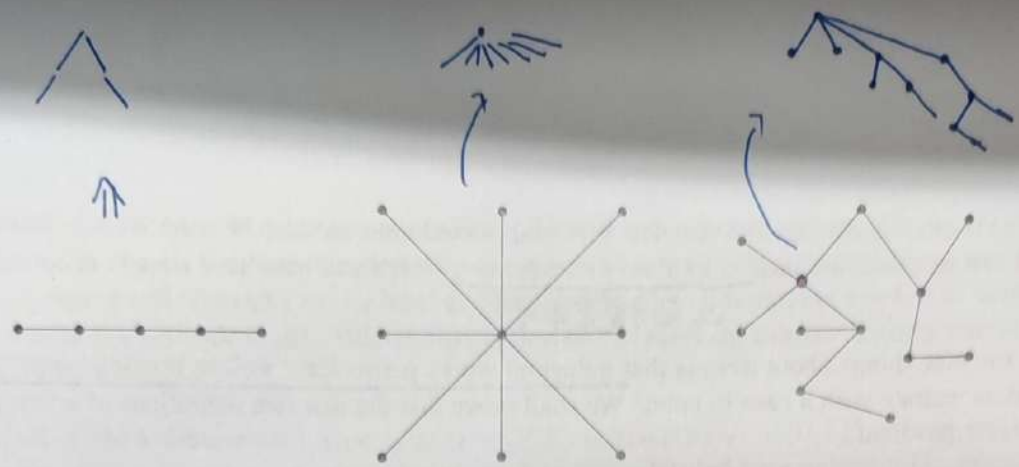
Finally, we can also discuss complete graphs for *directed* graphs, which are defined as you might expect: For any pair of vertices u and v, both $(u, v), (v, u) \in E$.

4.2 Trees

In this section, we return to trees. If complete graphs are "maximally connected," then trees are the opposite: Removing just a single edge disconnects the graph! Formally, there are a number of equivalent definitions of when a graph $G = (V, E)$ is a tree, including:

1. G is connected and contains no cycles.
2. G is connected and has $n - 1$ edges (where $n = |V|$).
3. G is connected, and the removal of any single edge disconnects G.
4. G has no cycles, and the addition of any single edge creates a cycle.

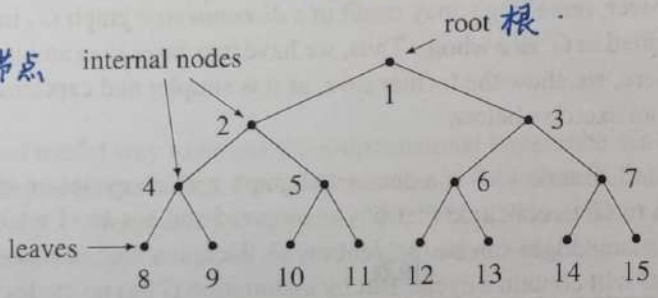
Here are three examples of trees:



Concept check!

1. Convince yourself that the three graphs above satisfy all four equivalent definitions of a tree.
2. Give an example of a graph which is *not* a tree.

Why would we want to study such funny-looking graphs? One reason is that many graph-theoretical problems which are computationally intractable on arbitrary graphs, such as the Maximum Cut problem, are easy to solve on trees. Another reason is that they model many types of natural relationships between objects. To demonstrate, we now introduce the concept of a *rooted* tree, an example of which is given below.



In a rooted tree, there is a designated node called the *root*, which we think of as sitting at the top of the tree. The bottom-most nodes are called *leaves*, and the intermediate nodes are called *internal nodes*. Note that in a rooted tree, a root is never a leaf. In particular, this differs from an unrooted tree; in an unrooted tree, *leaves* are any vertex of degree 1. The *depth* d of the tree is the length of the longest path from the root to a leaf. Moreover, the tree can be thought of as grouped into layers or *levels*, where the k-th level for $k \in \{0, 1, \ldots, d\}$ is the set of vertices which are connected to the root via a path consisting of precisely k edges.

Concept check!

1. What is the depth of the tree above? (Answer: 3)
2. Which vertices are on level 0 of the tree above? How about on level 3?

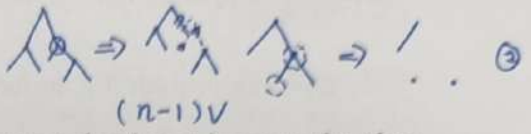
Where do rooted trees come in handy? Consider, for example, the setting of bacterial cell division. In this case, the root might represent a single bacterium, and each subsequent layer corresponds to cell division in

which the bacterium divides into two new bacteria. Rooted trees can also be used to allow fast searching of ordered sets of elements, such as in *binary search trees*, which you may have already encountered in your studies.

二叉搜索树

One of the nice things about trees is that induction works particularly well in proving properties of trees. Let us demonstrate with a case in point: We shall prove that the first two definitions of a tree given above are indeed equivalent.

Theorem 5.5. *The statements "G is connected and contains no cycles" and "G is connected and has $n-1$ edges" are equivalent.*



Proof. We proceed by showing the forward and converse directions.

(n-1)V

Forward direction. We prove using strong induction on n that if G is connected and contains no cycles, then G is connected and has $n-1$ edges. Assume $G=(V,E)$ is connected and contains no cycles.

Base case ($n=1$): In this case, G is a single vertex and has no edges. Thus, the claim holds.

I.

Inductive hypothesis: Assume the claim is true for $1 \leq n \leq k$.

Inductive proof: We show the claim for $n=k+1$. Remove an arbitrary vertex $v \in V$ from G along with its incident edges, and call the resulting graph G'. Clearly, *removing* a vertex cannot create a cycle; thus, G' contains no cycles. However, removing v may result in a *disconnected* graph G', in which case the induction hypothesis cannot be applied to G' as a whole. Thus, we have two cases to examine — either G' is connected, or G' is disconnected. Here, we show the former case, as it is simpler and captures the essential proof ideas. The latter case is left as an exercise below.

if G' disconnected

① m v.

So, assume G' is connected. But now G' is a connected graph with no cycles on k vertices, so we can apply the induction hypothesis to G' to conclude that G' is connected and has $k-1$ edges. Let us now add v back to G' to obtain G. How many edges can be incident on v? Well, since G' is connected, then if v is incident on more than one edge, G will contain a cycle. But by assumption G has no cycles! Thus, v must be incident on one edge, implying G has $(k-1)+1=k$ edges, as desired.

矛盾

⇒ II.

C+m+v+1=k+1

⇒C-1+m-1+1=k-1

= k-3.

edges ck-3+1

② C m.

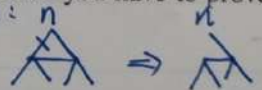
C+m+1=k+1

C-1+m-1+2.

=k ✓

Converse direction. We prove using contradiction that if G is connected and has $n-1$ edges, then G is connected and contains no cycles. Assume G is connected, has $n-1$ edges, and contains a cycle. Then, by definition of a cycle, removing any edge in the cycle does not disconnect the graph G. In other words, we can remove an edge in the cycle to obtain a new connected graph G' consisting of $n-2$ edges. However, we claim that G' must be disconnected, which will yield our desired contradiction. This is because in order for a graph to be connected, it must have at least $n-1$ edges. This is a fact that you have to prove in the exercise below. This completes the proof of the converse direction. □

vertice: n



不存在.

n-1 n-2 < n-1 minimum

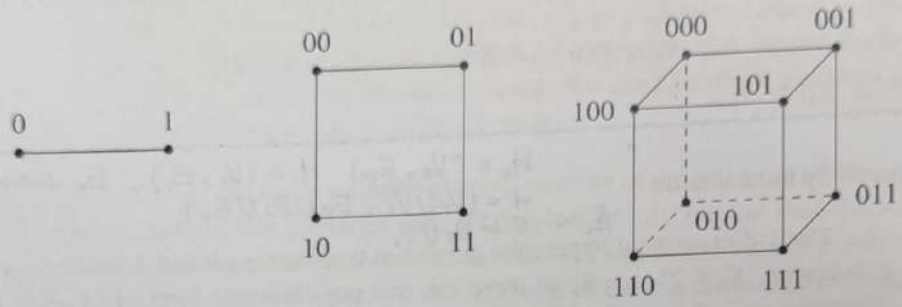
X

5.3 Hypercubes 超立方图

We have discussed how complete graphs are a class of graphs whose vertices are particularly "well-connected." However, to achieve this strong connectivity, a large number of edges is required, which in many applications of graph theory is infeasible. Consider the example of the *Connection Machine*, which was a massively parallel computer by the company Thinking Machines in the 1980s. The idea of the Connection Machine was to have a million processors working in parallel, all connected via a communications network. If you were to connect each pair of such processors with a direct wire to allow them to communicate (i.e., if you

used a complete graph to model your communications network), this would require 10^{12} wires! What the builders of the Connection Machine thus decided was to instead use a 20-dimensional *hypercube* to model their network, which still allowed a strong level of connectivity, while limiting the number of neighbors of each processor in the network to 20. This section is devoted to studying this particularly useful class of graphs, known as hypercubes.

The vertex set of the n-dimensional hypercube $G = (V, E)$ is given by $V = \{0,1\}^n$, where recall $\{0,1\}^n$ denotes the set of all n-bit strings. In other words, each vertex is labeled by a unique n-bit string, such as $00110\cdots0100$. The edge set E is defined as follows: Two vertices x and y are connected by edge $\{x, y\}$ if and only if x and y differ in exactly one bit position. For example, $x = 0000$ and $y = 1000$ are neighbors, but $x = 0000$ and $y = 0011$ are not. More formally, $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_n$ are neighbors if and only if there is an $i \in \{1,\ldots,n\}$ such that $x_j = y_j$ for all $j \neq i$, and $x_i \neq y_i$. To help you visualize the hypercube, we depict the 1-, 2-, and 3-dimensional hypercubes below.



There is an alternative and useful way to define the n-dimensional hypercube via recursion, which we now discuss. Define the 0-subcube (respectively, 1-subcube) as the $(n-1)$-dimensional hypercube with vertices labeled by $0x$ for $x \in \{0,1\}^{n-1}$ (respectively, $1x$ for $x \in \{0,1\}^{n-1}$). Then, the n-dimensional hypercube is obtained by placing an edge between each pair of vertices $0x$ in the 0-subcube and $1x$ in the 1-subcube.

Concept check! Where are the 0- and 1-subcubes in the 3-dimensional hypercube depicted above? Can you use these along with the recursive definition above to draw the 4-dimensional hypercube?

Exercise. Prove that the n-dimensional hypercube has 2^n vertices. Hint: Use the fact that each bit has two possible settings, 0 or 1.

We began this section by singing praises for the hypercube in terms of its connectivity properties; we now investigate these claims formally. Let us begin by giving two proofs of a simple property of the hypercube. Each proof relies on one of our two equivalent (namely, direct and recursive) definitions of the hypercube.

Lemma 5.1. *The total number of edges in an n-dimensional hypercube is $n2^{n-1}$.*

Proof 1. The degree of each vertex is n, since n bit positions can be flipped in any $x \in \{0,1\}^n$. Since each edge is counted twice, once from each endpoint, this yields a total of $n2^n/2 = n2^{n-1}$ edges. □

V : $\frac{V(V-1)}{2}$.

→ 对比：

V : $\frac{V}{2}\log_2 V$

Proof 2. By the second definition of the hypercube, it follows that $E(n) = 2E(n-1) + 2^{n-1}$, and $E(1) = 1$, where $E(n)$ denotes the number of edges in the n-dimensional hypercube. A straightforward induction shows that $E(n) = n2^{n-1}$. □

Exercise. Using induction to show that in Proof 2 above, $E(n) = n2^{n-1}$.

$V(n) = 2^n$ $E(n) = n \cdot 2^{n-1}$, n — dimensional $\{0,1\}^n$.

Let us focus on the question of connectivity, and prove that the n-dimensional hypercube is well-connected in the following sense: To disconnect any subset $S \subseteq V$ of vertices from the rest of the graph, a large number of edges must be discarded. In particular, we shall see that the number of discarded edges must scale with $|S|$. In the theorem below, recall that $V - S = \{v \in V \mid v \notin S\}$ is the set of vertices that are not in S.

Theorem 5.6. *Let $S \subseteq V$ be such that $|S| \leq |V - S|$ (i.e., that $|S| \leq 2^{n-1}$), and let E_S denote the set of edges connecting S to $V - S$, i.e.,*

$$E_S := \{\{u,v\} \in E \mid u \in S \text{ and } v \in V - S\}.$$

Then, it holds that $|E_S| \geq |S|$.

Proof. We proceed by induction on n.

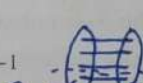
$H_0 = (V_0, E_0)$ $H_1 = (V_1, E_1)$, E_x connect them.
$H = (V_0 \cup V_1, E_0 \cup E_1 \cup E_x)$
点 ← $S = S_0 \cup S_1$.

1. *Base case ($n = 1$):* The 1-dimensional hypercube graph has two vertices 0 and 1, and one edge $\{0,1\}$. We also have the assumption $|S| \leq 2^{1-1} = 1$, so there are two possibilities. First, if $|S| = 0$, then the claim trivially holds. Otherwise, if $|S| = 1$, then $S = \{0\}$ and $V - S = \{1\}$, or vice versa. In either case we have $E_S = \{0,1\}$, so $|E_S| = 1 = |S|$.

2. *Inductive hypothesis:* Assume the claim holds for $1 \leq n \leq k$.

   Inductive step: We prove the claim for $n = k+1$. Recall that we have the assumption $|S| \leq 2^k$. Let S_0 (respectively, S_1) be the vertices from the 0-subcube (respectively, 1-subcube) in S. We have two cases to examine: Either S has a fairly equal intersection size with the 0- and 1-subcubes, or it does not.

   1. **Case 1: $|S_0| \leq 2^{k-1}$ and $|S_1| \leq 2^{k-1}$**

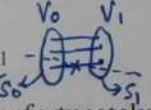
   $|S_0| \leq 2^{k-1}$, $|S_1| \leq 2^{k-1}$ > 符合 $1 \leq n \leq k$ 式子 ⇒ Edges cut in $H_0 \geq |S_0|$, Edges cut in $H_1 \geq |S_1|$

   In this case, we can apply the induction hypothesis separately to the 0- and 1-subcubes. This says that restricted to the 0-subcube itself, there are at least $|S_0|$ edges between $|S_0|$ and its complement (in the 0-subcube), and similarly there are at least $|S_1|$ edges between $|S_1|$ and its complement (in the 1-subcube). Thus, the total number of edges between S and $V - S$ is at least $|S_0| + |S_1| = |S|$, as desired.

   total $\geq |S_0| + |S_1| = |S|$

   2. **Case 2: $|S_0| > 2^{k-1}$**

   In this case, S_0 is unfortunately too large for the induction hypothesis to apply. However, note that since $|S| \leq 2^k$, we have $|S_1| = |S| - |S_0| \leq 2^{k-1}$, so we *can* apply the hypothesis to S_1. As in Case 1, this allows us to conclude that there are at least $|S_1|$ edges in the 1-subcube crossing between S and $V - S$.

   What about the 0-subcube? Here, we cannot apply the induction hypothesis directly, but there is a way to apply it after a little massaging. Consider the set $V_0 - S_0$, where V_0 is the set of vertices in the 0-subcube. Note that $|V_0| = 2^k$ and $|V_0 - S_0| = |V_0| - |S_0| = 2^k - |S_0| < 2^k - 2^{k-1} = 2^{k-1}$. Thus, we *can* apply the inductive hypothesis to the set $V_0 - S_0$. This yields that the number of edges between

~~$k+1 \Rightarrow 2^{k+1} = \ldots$~~

这里损失的 edges 三部分组成 {
① V_0 与 S_0 部分 可看成 V_0 与 S 的关系 $|V_0 - S_0| \leq 2^{k-1}$ 从中剥离
⇒ $E_1 \geq |V_0 - S_0| = |V_0| - |S_0|$
② V_1 与 S_1 部分 $|S_0| > 2^{k-1} \Rightarrow |S_1| \leq 2^{k-1}$ ⇒ $E_2 \geq |S_1|$
③ 无法对不 V_0 与 V_1 连接部分 ⇒ S_0 与 S_1 的连接 ⇒ $E_3 \geq |S| - |S_1|$

综上 $E = E_1 + E_2 + E_3 \geq (|V_0| - |S_0|) + |S_1| + (|V_0| - |S_1|) = 2^k \geq |S|$