目录

Proposition

Proposition 表示的是一个陈述，换一种说法是只有可能是 true 或者 false，只有一种可能. 比如 “明天下雨”是一个 Proposition，但$3x = 6$不是，因为 x 可能使这个陈述不仅 true 或者 false

Predicate

- ∧: 与，也就是 and
- ∨: 或，也就是 or
- ~: 非，也就是 not
- ≡: 相等，更确切的说是逻辑上相等
- ∈: 属于，指某个元素在某个 set (Domain) 中
- $P(x)$: 像是一个计算机的函数，我们把变量 x 放进去，会返回 true 或者 false，他们自身输入不了 true 或 false, x 只能输入事件进去
- ∀: 指所有，比如 $\forall x \in D, P(x)\ is\ true$ 指的是对于所有在 Set 里的 x，对于 P(x)都是正确的
- ∃: 存在，比如$\exists\ x \in \mathbb{R}, x^2 > 2$，指的是在实数域中，有些数的方是大于 2 的
- ¬: 相反，在一些情况下很有用，如$\forall\ x\ \in \mathbb{Z}, \exists y \in \mathbb{Z}, x + 2y = 3$，看起来很难解，但整体相反一下，得到$\exists x \in \mathbb{Z}, \forall y \in \mathbb{Z}, x + 2y \neq 3$，这样就成了一个正伪题
- →: implication
- $|S|$: 集合 S 的大小，也就是里面有多少元素

英语歧义

在有些情况下，细微的语序变化会导致结果不一样

Jim is tall and Jim is thin $p \wedge q$ -> Jim is not tall or Jim is not thin $\sim p \vee \sim q$

Jim is tall and thin -> Jim is not tall and thin 这里是把 tall 和 thin 的pq看作一个

使用

在使用的时候，我们需要先定义域和事件，再定义剩下的，比如我们想写只要一个整数的平方是双数，那么这个整数就是双数.

定义域:

$$\forall x \in \mathbb{Z}$$

定义事件:

$$S(x) = squre\ of\ the\ integer\ is\ even$$
$$E(x) = this\ integer\ is\ even$$

$$\forall x \in \mathbb{Z}, S(x) \rightarrow E(x)$$

Implication

表示在某种条件下会发生某件事情，翻译过来就是 if ... then ... ，也就是$P(x) \rightarrow Q(x)$

在计算机中，可以这么表示:

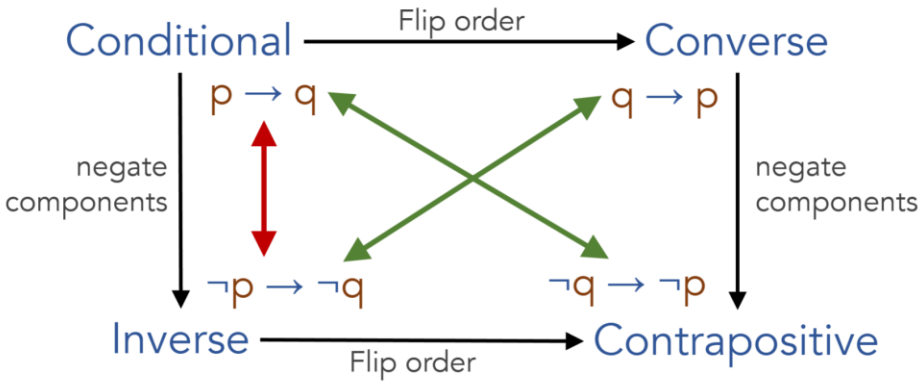
```
if(p){
    // perform q function/method
 q();
}else{
   // do something
}
```

对应真值表会有下面几种情况。

- 如果 p 是 true，那么方法 q()会被执行，因为真值表的 q 是 true，表明方法 q()有被执行，所以命题正确。
- 如果 p 是 true，那么方法 q()会被执行，因为真值表的 q 是 false，表明方法 q()没有被执行，所以命题错误。
- 如果 **p 是 false**，直接跳到 else 语句。我们重新看下蕴含的定义"**if p, then q"，**如果满足 p，那么执行 q。前置条件 p 都不满足了，那么讨论 then q 会不会执行是没有意义的(比如今天都不下雨了，带不带伞也没意义了)，因为我们只 claim 了 p 是真的情况下 q 会怎样，并没有 claim p 为假时 q 会怎样，所以命题为真。

如果我们想把这个用基础符号写出来，我们可以这么写：$\sim P \vee Q$，他们俩是逻辑上相等的.

由于我们已经写成这样的形式了，我们现在也可以¬→了，因为我们直接相反上面这个基础符号的等式就行



An *implication* and its *contrapositive* are logically equivalent
The *converse* and the *inverse* are logically equivalent
An *implication* is not logically equivalent to its *inverse*

如果我们还想进一步表示关系，可以用双箭头，表示 if and only if... $P(x) \leftrightarrow Q(x)$

这个的意思跟上面有点相似，但是这次必须是 P 和 Q 都必须相等(true 或 false 都可以) 最

终结果才是 true

Sets

Set，也就是合集，跟编程语言中定义的一样，是一个容器里没有顺序的存放着一堆不重复的物品，如果里面有重复的，我们需要忽视掉，或者说这个 Set 是不正规的

就像在编程里一样，我们也可以把 Set 放进 Set 中.如$S = \{\{1,2,3\},\{4,5,6\}\}, |S| = 2$，而且，关系也和编程里一样，我们需要说$1 \notin S$，而是说$\{1,2,3\} \in S$，就算一个空集在集合$S$中，$S$的大小依然是 1：$S = \{\emptyset\} = \{\{\}\}$

上面的∈符号是表示单个元素属于某个 Set 中，如果想表示 Set B 中的元素 Set A 中都有,就说 A 包含 B，$B \subseteq A$，如果$(B \subseteq A) \wedge (A \subseteq B)$，则我们认为$A = B$

创建一个 Set

如果要创建一个 Set，可以用以下格式：

$$set\ name = \{domain : predicate\}$$

例：

Power set of A 表示的是一个包含 Set A 中所有可能的 Sub set 的集合.

$$P(A) = (S : S \subseteq A)$$

Power set 的大小是$|A|^2$，比如$A = \{1,2\}, P(A) = \{\{\emptyset\},\{1\},\{2\},\{1,2\}\}$

Partition of set:

如果我们想说一个 Set 的 Partition，表示的是把这个 Set 分割成多个小 Sets，每个 Set 不能

为空，且每个 Set 不会有重复的元素

Cartesian Products

是指两个集合 X 和 Y 的所有可能的有序对组成的集合，其中有序对的第一个元素属于 X，第二个元素属于 Y $A \times B = \{(x, y) : (x \in A), (y \in B)\}$

例如，如果$X = \{a, b\}$，$Y = \{0, 1\}$，那么$X \times Y = \{(a, 0), (a, 1), (b, 0), (b, 1)\}$. 笛卡尔乘积可以用来表示集合之间的关系，或者构造更高维度的集合.

Relation

在上这节课之前，我们用来表示两个物体的关系的符号用的最多的就是比大小符号$a < b$

后面，我们又学了 set 之间的关系 - 子集，$A \subseteq B$，或者物体的 set 的关系 - 属于，$a \in B$

对于两个 Set，如果 A 是 B 的子集，我们也可以说 A 是与 B 相关的(A related to B)

如何规范的定义一个关系?

如果我们说 set A 相关 set B，相关的规则是 R，那我们可以这么写如果想表示 A,B 里的元素根据 R 规则相关

$$a \in A, b \in B, a\ R\ b\ iff\ satisfy\ the\ rule\ R$$
$$a \in A, b \in b, (a, b) \in R\ \ iff\ \ satisfy\ the\ rule\ R$$

例:

$$A = \{1,2\}, B = \{1,2,3\}, xRy: \frac{x - y}{2} \in \mathbb{Z}$$

这个意思就是，从 Set A 和 Set B 各取一个数做 xy，如果相减除以 2 还是整数，那么就可以说这 x related to b

最后可以得到$R = \{(1,1),(1,3),(2,2)\}$

如果我们想说 Inverse of Relation，那么就可以把正常的 Relation 直接反过来就行．比如在正常的 Relation 中，$a_1Rb_1, a_2Rb_1, a_3Rb_2$

那么反过来就是 $b_1R^{-1}a_1, b_1R^{-1}a_2, b_2R^{-1}a_3,$ $R^{-1} = \{(b_1,a_1),(b_1,a_2),(b_2,a_3)\}$

Functions

Function 是一种特殊的关系，他需要每个 input 都对应一个 output．比如 Set A 里的所有元素 a 都有自己对应的元素 b 在 Set B 中，且每个 a 只对应一个 b，但 b 可以对应多个 A

如果我们想定义一个函数，可以说：

$$given\ two\ Sets, X\ and\ Y, a\ function\ f\ from\ X\ to\ Y\ relates\ each\ input\ element\ x \in X\ to\ only\ one\ element\ y \in Y, written\ as\ f: X \to Y$$

$$\forall x \in X, \exists y \in Y, such\ that\ f(x) = y$$
$$If\ f(x) = y, f(x) = z, then\ y = z$$

例：可以定义一个 function 是$f(x) = x^2$，可以看到对$\mathbb{Z}$做完变换以后每个 x 都有对应的数值，但不是所有正整数都可以由x^2得到，如 7

对于 Function 来说，也有 inverse，但是并不是所有的 Function 都有 inverse，如$f(x) = x^2$，由于没有数可以变成-1，所以只能从$\mathbb{Z}$变成$\mathbb{Z}^+$不能变回去，但是我们可以单纯的把他当做 Relation 来 inverse，也就是变成$f(x) = \sqrt{x}$

One-to-one, On to

*首先，需要清楚映射是不允许 1 对多的关系，只允许多对 1 的关系，且 Set A(原像)的每个元素都要有映射关系

对于 One-to-one 函数来说, Set A 里的每个元素必须有且只有一个在 B 中的元素与之对应.

$$\forall x_1, x_2 \in X,\ if\ f(x_1) = f(x_2), then\ x_1 = x_2$$

尽管 Set A 中的每个元素必须一一对应，但是 Set B 中的元素不需要都有元素与之对应，也就是说对应 Set B 中的每个元素的数量最多是 1 – 换句话说，对于 one to one 函数来说，输入不同，得到的结果也不同. 如果两个 Set 的关系满足单射(one to one)，那么 Set A 一定小于等于 Set B

对于 On to 函数来说, Set B 中的每个元素都要有 Set A 中的元素与之对应. 但 Set B 中的元素可以一次对应多个 Set A 中的元素

$$\forall y \in Y, \exists x \in X, f(x) = y$$

如果一个函数是 Invertible 的话，那就表示这个函数的逆也是个函数(必须保证同时 one to one 和 on to)

$$f^{-1}f(x) = x$$

Equivalence Relations

One Set arrow diagram

在之前，如果想表示关系，需要两个 set，但这种情况只需要一个 set，如$A = \{3,4,5,6,7,8\}$，

关系 R 表示的是$s.t.(x,y) \in R\ iff\ 2|(x-y)$

也就是 Set A 里面的两个数相减能被 2 整除，就是有关系，我们可以在一个 set 中按照这种关系连线

Reflexive

在连完线后，能发现所有的元素都和自己是有关联(0 可以被 2 整除)，这就是 Reflexive

Reflexive：$iff\ \forall\ a \in A,\ (a,a) \in R$

如果是 Irreflexive：$iff\ \forall\ a \in A,\ (a,a) \notin R$

Symmetric

还有一种特殊的关系是 Symmetric，$iff\ \forall\ a,b \in A,(a,b) \in R,(b,a) \in R$，表示的是如果这个关系可以互换，就是对称的，反之不行，但是某些情况可以不仅对称还不对称：

$$A = \{\{a,a\},\{b,b\}\}$$

不对称的条件是：

$$iff\ \forall a,b \in A, a \neq b \wedge (a,b) \in R \rightarrow (b,a) \in R$$

因为$a = b$，所以这个 imply 就是对的了.

Transitive

$$iff\ \forall a,b,c \in A,(a,b) \in R \wedge (b,c) \in R \rightarrow (a,c) \in R$$

也就是如果 ab 有关系，并且 bc 也有关系，那么 ac 一定要有关系

注：所有的 special Relation 都不等于它的 not not xx，如 Transitive 不等于 not Non

Transitive.

如果以上三个关系都满足的话，那么这个合集就是个偏序集，如果 Set 中每个元素的大小都可以比较(不会相等)，那么就是全序集.

例:

$$a, b \in \mathbb{Z} \ ， \ (x, y) \in R \ iff \ 3|(x - y)$$

问他有什么关系?

首先，肯定有 Reflective，因为所有的数都能除 0.

然后也有 Reflexive，因为所有的关系都一定是 3 的倍数，如果a, b交换，就回是-3的倍数，

对于 transitive

$x - y = 3k$，其中 k 是某个整数

$y - z = 3l$，其中 l 是某个整数

将上面两个等式相加，我们得到:

$x - z = 3(k + l)$ ，最后结果是 yes

Partition

Partition 表示的是把一个 Set 分为多个互不相关的子集

Equivalence Class

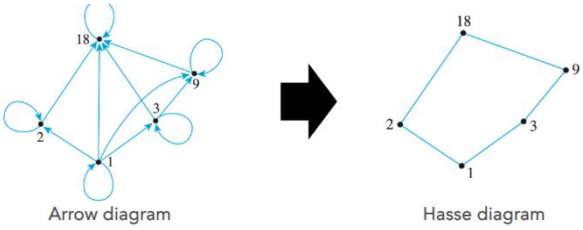
这个是关系最多的，因为如果想要一个 Set 是 Equivalence class，就需要一个 Set 里面所有的元素都跟它有关系，包括他自己

Arrow diagram -> Hasse Diagram

如果想画一个哈斯图，首先需要确定这个是一个偏序合集，也就是所有元素都有自反性，**非**对称性，和传递性.必须

绘制规则是以下几点:

- 如果 a 小于等于 b，那么 a 的点在 b 的点的下方。
- 如果 a 小于等于 b，且不存在 c，使得 a 小于等于 c 小于等于 b，那么 a 的点和 b 的点用一条线段相连。
- 如果 a 等于 b，那么 a 的点和 b 的点重合，只画一个点。



简单来说，在画哈斯图的时候，我们可以把箭头，和自环给省掉，并且如果$(a, b) \in R$，$(b, c) \in R$，那么(a, c)肯定也有关系，所以也可以省掉，这就是为什么哈斯图必须要求这是一个偏序合集

Graph

就像计算机里的图一样，离散数学的图也可以被分为两个东西: Vertices, Edges. 由这两个 Set 组成了一个 Graph, Set G.

图分为有向和无向，用数学语言表达是：$\{a, b\}, [a, b]$表示的是无向，而(a, b)表示的是有向.

而且如果在图中有两个节点互联了两次(也叫做 multiple)，我们可以在 edge list 中写两遍来表示，尽管我们之前学过 Set 不支持重复，但 Graph 打破了这个规定

图的术语：

- Adjacent：两个节点有个 Edge 相连我们就可以说这两个节点是相邻的
- Incident：表示某个 Edge 连接了某个节点
- Isolated：如果一个节点没有任何 Edge 指向它则说这个节点是独立的
- Degree： 表示一个节点上有多少个 Edge 连接它，对于无向图中每个边都算 degree+1，但是对于有向图来说分为 in-degree 和 out-degree，分别是指向节点和离开节点的 Edge 的数量
- Walk： 表示在节点之间移动，写出来就是v_0, e_1, v_1, e_2 … 一个节点后面跟着 edge，再跟着节点．并且路径可以重复经过同一个节点，我们说如果一个 walk 是 close 的，则表示出发点和终点是相同的
- Trail： 是一种特殊的 walk，它经过的边不能相同，但经过的顶点可以重复
- Path： 也是一种特殊的 walk，经过的边和顶点都不能相同
- Cycle： 循环是一个特殊的 Path，它不能有重复的边，除了第一个和最后一个节点相同，剩下的节点都只能走一遍
- Connected Graph：如果任意两个节点都有 path 可以去到，那么这个的图就是相连的．
- Spanning subgraph：$G = (V_G, E_G), H = (V_H, E_H)$，如果 H 是 G 的 Subgraph $iff\ V_H = V_G \wedge E_H \subseteq E_G$
- Simple Graph：指一个图没有重复的边(一个节点连另一个节点两次)或自环
- Eulerian：如果某个 path 或 circuit 正好使用每个节点和 edge 一次，我们就叫这个 trail 是 Eulerian trail．一个 graph 如果存在 Eulerian Circuit，那我们可以叫这个 Graph 是 Eulerian Graph．如果一个图是 Connected Graph，且所有节点的 Degree 都是双数，则这个图是 Eulerian Graph，如果一个 Connected Graph 但它不是

Eulerian Graph，而且它正好有两个节点的 Degree 是单数，则这个图肯定有一个 Eulerian Trail.

Tree

如果一个 Tree 是 Simple(没有节点自己连自己), connected, no cycle，那么他就是一棵树

树的特性:

- 对于两个节点，只有唯一的一条路径连接到它.
- 一个树的节点数量永远比边的数量多 1

Spanning tree 是指如果一个 T 是 G 的 Spanning Tree，那么 T 需要包含 G 的所有节点，但还需要符合 Tree 的性质

Minimal spanning tree

如果 T 是**有权重**图 G 的 MST，那么 T 一定 G 的 Spanning Tree，并且总共的 weight 还是最低的，如果我们想构造一个 MST，我们需要用下面步骤(Prim's algorithm):

1. 选择任意一个节点并高亮
2. 找到从高亮节点出发 cost 最小的 edge(且不会构成回路)
3. 把那个找到的 edge 连接到的新节点也高亮
4. 回到步骤 1

或者 Kruskal's algorithm

1. 把所有带有权重的边排好序

2. 从小到大的考虑这些边，如果选取这个边不会构成回路则加入到我们要构建的树中

虽然但是，上面的算法无法找到两个节点最短的路径，只能找到整体权重更低的.

要是想找两个节点之间最短路径，需要 Dijkstra's Algorithm，简单来说就是把每个节点的 weight 都设置成无限，从一个节点出发并更新它走到的节点.

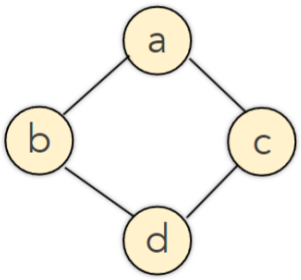
Isomorphic graphs

当两个图G, H的结构一样，但节点不一样的时候，可以说$G \cong H$在这种情况下，我们可以把这个关系写出一个 Function，这个函数可以把G中所有的节点一对一的映射到H上，这个函数必须是 One-to-one 和 onto，而且不能乱映射，也得保留结构关系.

Graph → Matrix

我们可以把图的关系放进矩阵里：

行表示的是从，列表示的是到，如 a 到 b 就是一行 2 列



$$\begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \end{array} = M$$

矩阵给了我们很多信息：首先对角线都是 0 表示矩阵没有自循环

每行的元素之和都是 2，表示的是矩阵每个顶点的 out degree.

矩阵的每列也都是 2，也表示的是矩阵每个顶点的 in degree，由于这是个无向图，所以行和列的和相等.

正是因为这是个无向图，所以矩阵的转置等于他自己

既然都写成矩阵了，那当然也可以做矩阵乘法.

$$M * M = M^2$$

得到的这个矩阵自己的方告诉了我们有多少种方法从一个地方去另一个地方通过只有 2 的长度的路：

比如上面的这个矩阵，我们想算$M_{12} = 0$运算过程都是 0

M_{ij}^k表示的是通过 k 长度的 Walk，我们从 i 到 j 有多少种选择

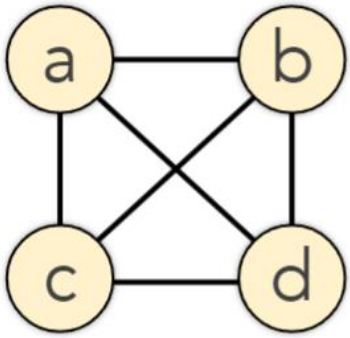
如果在相乘的时候，把乘法换成 Boolean，也就是只有两个 1，才会等于 1，这就叫做 Boolean product，运算简单了，但是也只能告诉我们有没有从i到j的 walk 存在，但不能告诉我们有几条了

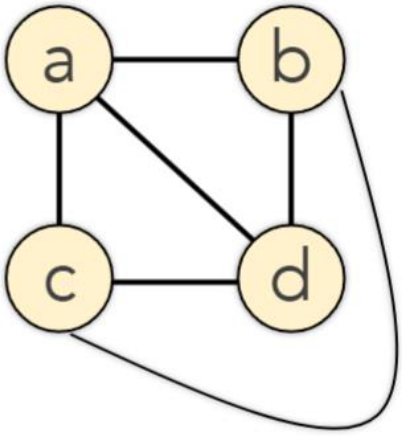
Bipartite Graph

Bipartite graph（二分图）是指一个图中的节点可以被分为两个不相交的集合，使得图中的每条边连接的两个节点分别属于这两个集合。换句话说，如果将图中的节点分成两组，那么每条边都连接一组中的一个节点到另一组中的一个节点，而不会存在同一组内的节点之间有边相连。

$K_{2,3}$表示的是分成两个集合，一个有 2 个节点，另一个有三个节点

Planar graph 表示的是在图中没有边会交叉





$$v - e + f = 2$$

Combinatorics

例：如果我们想排列 6 个人，一共有多少组合?

$$6! = 720$$

这就是组合学的一个例子

组合学的定义是：计算有限结构的数量(如某种任务有多少种实现方法)

组合学的用处是：检测某种设置(实现方法)是否存在，这样可以让人员优化

Sample Space & Events

Sample Space- 表示的是一个合集，包含了所有可能发生的结果

Event - Sample Space 的子集，如果这个 Event 无法分割成更小的事件，则叫做 simple event.

Task - 我们想干的事情

Domain - 所有我们可以构建的解决方案

Outcome - 最终有用的的解决方案

例：有 52 张牌，我们想抽到大于 10 点的牌，所以 Domain 是 52 张牌，而 Outcome 是 12

或者从 ABCD 四个人中选两个胜者，问有几种组合

Domain 是 set$\{A, B, C, D\}$，而 Domain 是所有两个元素的组合.

但 Outcome 由于题目没有给清导致有四种可能

	Order Matter	Order not Matter
Repetitions are allowed	AB and BA are two different solutions AA is allowed $$P(n,r) = n^r$$ $4^2 = 16$ Outcome	AB and BA are the same solution AA is allowed $4 + 3 + 2 + 1 = 10$ Outcome
Repetitions are not allowed	AB and BA are two different solutions AA is not allowed $$P(n,r) = \frac{n!}{(n-r)!}$$ $4 * (4-1) = 12$ Outcome	AB and BA are the same solution AA is not allowed $$C(n,r) = \frac{n!}{r!\,(n-r)!}$$ $\frac{24}{2*2} = 6$ Outcome

我们需要根据实际情况来选择：如奖品一样的话，就是顺序不重要.

Rule of counting

Rule of products	A 有 n 种解决办法, B 有 m 种解决办法，则有$n \times m$种解决办法 如果我们先做 A 再做 B 如在$100\sim1000$中有$5 * 4 * 3 = 60$个不相同的单数组成的数
Rule of Sums	A 有 n 种解决办法, B 有 m 种解决办法，则有$m + n$种解决办法 去做 A 或 B $\quad \lvert A \cup B \rvert = \lvert A \rvert + \lvert B \rvert$ 如扔三次骰子，顺序重要，只扔到一次 1 的组合有几种? 首先用 Rule Of Product，$1 * 5 * 5 = 25$种可能去得到第一个是 1 的组合，因为有三次扔骰子的机会，所以 1 可能在三个位置的某一处，得到最终结果是$25 + 25 + 25 = 75$
Rule of Complements	如果一共有 x 个物体，其中 y 个有某项性质，则有$x - y$个物体没有这项性质 如$1\sim11$有多少个单数，因为有 5 个双数，所以$11 - 5 = 6$

$(x + y)^n$

在计算$(x + y)^n$的时候经常很痛苦，但是依靠上面的那些公式可以很容易算出来:

$$for\ n \in \mathbb{Z}^+,\ (x + y)^n = C(n, 0)x^n y^0 + C(n, 1)x^{n-1}y^1 + \cdots + C(n, n)x^0 y^n$$

Permutation

在组合学中，如果我们遇到这样的问题：五张牌选三张，是和五张牌选两张是相等的．因为在当我们选择三张牌后，剩下的两张就是第二问的答案．

$$C(n,r) = C(n,n-r)$$
$$P(n,r) \neq P(n,n-r)$$

这种情况下，就叫 Combinatorics Equivalence

Overcounting

Overcounting 表示的是我们通过一种方法算出的数比实际上的要多：

如有六个人要坐圆桌，因为$\{1,2,3,4,5,6\}$和$\{2,3,4,5,6,1\}$是一样的，所以对于每种可能的组合都多出 5 种组合，所以要除以 6 来消除 overcounting.

$$\frac{6!}{(6-6)! * 6} = 5!$$

Distinguishable Permutations

如果我们想重新排序 MISSISSIPPI，我们有多少种选择呢?

一开始，我们可能直接用公式$P(11,11) = 11!$，但其实是错误的，因为这个单词里面有很多重复的字母，如果放一起，我们交换他们俩之间的位置，结果是相同的，也就是说出现了 overcounting．为了消除 overcounting，我们可以把所有可能(这里两个相同字母也可以不同组合)除以这些相同字母的所有组合．

$$\frac{11!}{4!\,4!\,2!\,1!}$$

换一种方法，我们可以用二进制有几个 1 的解法做也是一样的，先求放进 M 有几种可能，再放 I, S, P. 这里选择$C(n,r)$是因为我们想象有 n 个相同的字母(但是字母记录了在哪个位置)，

我们只是在抽取 r 个，抽取到哪个就代表哪位有这个字母

$$C(11,1)*C(10,4)*C(6,4)*C(2,2)$$

如果我们想个更复杂一点的：有多少个正整数，选三个可以组成10？$x+y+z=10$

这个看似很困难，但我们可以把它转换成 12 位的二进制有几种可能只出现两个 1. 这个看起来无厘头，但如果写出来就很明了：

010000100000

我们实际上是在用 1 做分隔，因为有 12 位，所以一定有 10 个 0，隔出来的最终结果就是x,y,z最终的值$C(12,2)$

可以看出来，对于问题:$x_1+x_2+x_3+\cdots+x_n=r$，从 n 种东西中选择 r 个物品有多少总组合，从一个 n 大小合集中取出一个 r 长度的无序组合(重复是允许的) 这些问题都可以总结成$C(r+n-1,r)$

还有个衍生题，对于已知 n，我们对i,j,k,m有多少种组合可以组成$1\le i\le j\le k\le m\le n$

这个题可以话简称$C(n+4-1,4)$，比如$n=5$，我们想象有 8 个位置．其中四个对应数字，四个棍子．每个棍子表示一个字母，棍子后面的数字表示字母的值．我们选择四个位置表示棍子，数字的位置自然出来了,如$xx||x|x|$前两个 x 就是 1,2，前两个棍子表示的都是 3，所以这个组合就是$1\le 3\le 3\le 4\le 5\le 5$

但上面这个公式并不可以直接帮我们解决用四个骰子扔出来 14 有多少组合．我们需要一系列修改让他尽可能靠近这个公式．由于$C(r+n-1,r)$包含组合为 0 的可能，但是骰子必定大于等于 1，所以我们可以把所有的结果-1，就变成了$d_1+d_2+d_3+d_4=10$ 但就算现在使用

$C(10+4-1,10)=286$也会出现 overcounting，因为这个公式还包含大于等于 6 的可能.

所以现在要求至少有一个骰子大于 7 的情况．因为我们已知一个已经≥ 7，且剩下的三个必须≥ 1，所以$14-7-3=4$，我们现在就是说已经有一个骰子大于 7 了，剩下 4 点怎么规划

$$C(4+4-1,4)=35$$
$$286-35*4=146$$

Probability

如果概率是均等的，我们可以我们在求的概率的事情看做一个 Set E，所有可能发生的事件看做U，$E\subseteq U$，则事件E的概率就是$\frac{|E|}{|U|}$.

很多时候顺序也是重要的：如扔两个骰子，如果顺序不重要$P(\{1,1\})\neq P(\{1,2\})$，否则$P\big((1,1)\big)=P\big((1,2)\big)$

Probability Axioms

- Discrete Probability：不可能出现 Sample Space 之外的结果
- Boundedness & normalization axiom：概率必须$0\leq P(x)\leq 1$
- Additivity Axiom：如果两个事件不重叠 $Prob(E_1\cup E_2)=Prob(E_1)+Prob(E_2)$
- Rule of Sum：如果两个 Event 不重叠，那么不可能同时发生两件事情$A\cap B=\emptyset$
- Rule of Complements：有一个事件 E 和 E 的补集 E′，则$P(E)+P(E')=1$

Independent

对于任意的两个事件，我们都可以这么写：

$$P(A \cup B) = P(A) + P(B) - P(A \cap B)$$

但如果两个事件独立，$P(A \cap B) = P(A) * P(B)$，比如扔硬币，两次投掷的结果互不影响.

需要注意的是，独立 Independent 和不重叠 disjoint 不是一个东西：如扔两个骰子, A:第一个骰子是双数, B：第二个骰子大于 3．这种就是独立但重叠，反过来我们还可能遇见不独立且不重叠的

Conditional Probability

对于条件概率，我们可以这么写:$P(A|B)$，也就是说 B 已经发生了，在这种情况下, A 发生的概率，而这个条件概率计算的方法是

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Bernoulli trail

Bernoulli trail，就跟 CS3130 里的定义一样，也就是只可能有两种结果，成功率由p确定.

虽然扔骰子不是 Bernoulli trail，因为有六个结果，但我们可以定义成功是≤ 4，失败是≥ 5.

这样就得到了一个 Bernoulli trail，他的$p = \frac{2}{3}$

问：球员进球的概率是$\frac{1}{3}$，并且每次结果不相关，那投 4 次刚好进一次的概率是多少?

$$C(4,1) * \frac{1}{3} * \left(1 - \frac{1}{3}\right)^3$$

我们可以得到通用公式：

$$Prob\left(E_{n,k}\right) = C(n,k) * p^k * (1-p)^{n-k}$$

Random Variable

Random Variable 是一个函数，会把 Event 映射到一个实数上，听着拗口，其实就是我们把一个事件用一个数来表示．如扔三次硬币，Random Variable X就可以是正面出现的次数

$$P(X = 3) = \frac{1}{8}$$

如果我们把所有可能的随机变量和对应的概率写出来，就得到了 Probability mass func

跟 CS3130 一样，我们也可以用概率密度函数来算期望：把所有的随机变量乘以它的概率再相加就得到了期望

$$E[X] = \sum X_i * P(X_i)$$

并且，期望是线性的：

$$E[aX] = aE[X], \quad E[X + Y] = E[X] + E[Y], \quad E[a + X] = a + E[X]$$

如一个公平的硬币，$E[X] = 0.5$，我们扔三次，就是$E[3X] = 1.5$

所以对于重复 n 次的 Bernoulli trail，$E[X] = n * p$

Proof

在数学/生活中，我们经常会遇到命题：如归 P 成立，则 Q 也成立... 等情况，我们需要证明它对不对．对于这种情况，我们有两类，每类有几个方法：

- Direct Proof：在数学中，我们可以证明两种逻辑是否相等，我们在开头学过了：就是把两边化简，化简到很简洁的时候比较就能比较出来了．

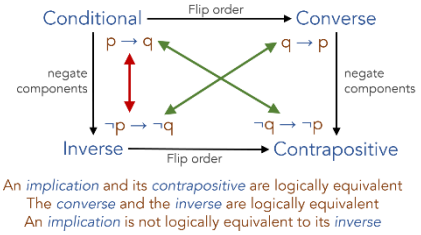
对于直接证明，还有一种是 induction 数学归纳法：就是首先假设事件在$n = 1$的情况下成立，并且也在$n = m$的情况下成立，m 是个任意的整数，我们现在只需要证明$n = m + 1$的情

况下成不成立就行了.

另外一种直接证明是 Proof by cases：也可以叫穷尽法．这里的穷尽并不是对于每个 x 都去求．而是把一个我们想证明的东西分成好几个情况，对于不同的情况进行证明．如证对于任意正整数 n 来说$n^2 + n$必然等于双数．我们就可以把这个情况分为 n 为单数和 n 为双数两种情况.

- Indirect Proof：这就要拿出之前学过的图了：

  这里有两种方法 证明的逆否命题（Proof by Contrapositive）和证明的矛盾法（Proof by Contradiction）



证明的逆否命题（Proof by Contrapositive）

对于这个，我们需要证明其他可能性都是错误的，就证明了之前的假设是正确的．我们需要根据上面的图标，把$p \to q$变成$\sim q \to \sim p$，因为这两个是等价的．所以我们只需要证明如果 q 不成立的情况下, p 也不成立就行.

如 m+n 是单数，则 m 或 n 必有一个是双数．所以我们可以写成 m,n 都是单数，单数的定义是$x = 2n + 1$，所以$m + n = 2x + 2y + 2 = 2(x + y + 1)$．因为整数的闭包性质 - 相加还是整数，所以可以化成$m + n = 2z$，就得出当 m,n 两个数都不是双数的时候，$m + n$一定不是单数.

证明的矛盾法（Proof by Contradiction）

我们首先要假设$p \to q$为假，就是说在 p 成立的情况下 q 并不成立，在这种情况下，我们取推到最终如果得到了一个悖论 - 说明我们的假设并不成立，也就是说 p 成立的情况下 q 也

成立.

Examples

如，我们想证明: m,n 都能被 4 整除，则$m+n$也可以，我们可以写$m=4i, n=4j \rightarrow m+n=4(i+j)$

Set Proofs

对于集合的证明，其实也是差不多的，还是上面的三种方法．我们如果想证明两个 Set 相等，我们就需要让两个 Set 各为对方的子集．为了达成这一点，可以说$\forall\, x \in A, x \in B$，就证明了$A \subseteq B$，再反过来证明$B \subseteq A$就行了. (proof by cases)

Proof By Induction

数学归纳法是另一种证明方法，这个方法是用来证明某个函数在自然数域或自然数域的子集中成立.

$$\sum_{k=1}^{n} a_k$$

如果想证明，第一想到的就是把数带进去，一个一个算．但这似乎是不可能的，因为数太多了.

这时候，就需要用数学归纳法了，它一共有四步:

1. 把数列总结成一个公式$P(n)$
2. 我们需要证明对于最小值(通常是 1)时，$P(n)$是否成立
3. 我们假设对于$P(m-1)$时成立(weak hypothesis)，或者假设对于$P(2), P(3), \dots, P(m-1)$都成立(strong hypothesis)
4. 如果$P(m-1)$成立，那么$P(m+1-1)=P(m)$也应该成立．所以如果我们成功证明了$P(m)$成立，则这个公式就是正确的

例：如果想证明所有> 1的数都是质数或者由质数相乘得到的数．

步骤 1 – 写出理论:$P(n)$不是质数就是由质数相乘的数

步骤 2 - $P(2)$是个质数，所以 base case 成立

步骤 3 – 假设对于$P(2), P(3), \dots, P(m-1)$都成立，也就是 m-1 不是质数就是由质数相乘

步骤 4 – 证明 m 是否让理论成立．这里有两个情况，情况 1: m 是质数，则理论直接成立，情况 2: m 不是质数，我们需要把它写成$m = r \times n,\ 2 \le r < m,\ 2 \le n < m$，由于我们假设对于$P(2), P(3), \dots, P(m-1)$都成立，则$r, n$也成立．所以最终 r,n 都可以被拆成由质数相乘的数．

例 2：所有≥8 的数都可以被 3 和 8 组成

步骤 1 – 写出理论:$P(n)$

步骤 2 - $P(8) = 3 + 8$, base case 成立

步骤 3 – 假设对于$P(n < m)$都成立

步骤 4 – 证明 m 是否让理论成立．这里有两个情况，情况 1：如果组成$m-1$的组合里含有一个 5，则我们可以把这个 5 换成两个 3，得到了额外的 1，所以成立．情况 2：如果组成 m-

1 的组合中没有 5，我们可以把三个 3 替换成两个 5，这样也能得到一个额外的 1.

鸽子洞原理

Pigeons hole 的原理很简单：如果我们有$n+1$个物体，想放到n个容器中，就表明一定有一个以上的物体在同一个容器中

其实我们可以把这个放置的过程写作一个映射：鸽子是 x，洞是 y. 如果每个鸽子只会在洞全满了以后才会进入有鸽子的洞：那么如果鸽子数量≤洞的数量，这个映射是 one to one，如果鸽子数量≥洞的数量，则这个映射是 onto.

我们还可以说把$xn+1$个物体放到n个容器中，则一定有某个容器至少有$x+1$个物体

通过悖论证明：把$xn+1$个物体放到n个容器中，则一定所有的容器最多有x个物体

$$xn+1=\sum k_i$$

因为有 n 个容器，每个容器里的物体不超过 x，所以所有容器的物体数量是nx

$$xn+1\neq xn$$

所以假设错误，证明原理论正确

例 2:

我们任意选四个正整数，其中必然有一对的差是可以被 3 整除.

我们可以把这个看作是 4 个鸽子

$$x3+1=4, x=1$$

接着我们把洞定义成 0,1,2，是某个数除以 3 剩下的数.

因为鸽巢原理，必然有一个洞有两个数，我们可以把这个数写成$3n+r$和$3m+r$

然后我们相减得到$3n+r-3m-r=3(n-m)$，因为整数闭包性质，所以一定可以整除