COMP9020 Math Cheat Sheet

数论

取整计算

- 上下取整常用于简化整数计算。例如，若 `⌊x⌋ = ⌈x⌉`，则表明 `x` 必为整数。
- **例题**：证明 `⌊x⌋ = ⌈x⌉` 意味着 `x` 是整数。
  - **解答思路**：使用夹逼原理 `⌊x⌋ ≤ x ≤ ⌈x⌉`，当等号两边相等时，`x` 必为整数。

整除性与最大公约数 (GCD) 和最小公倍数 (LCM)

- **Divisibility (整除):** 如果存在整数 `k` 使得 `n = k * m`，则 `m` 整除 `n`，记作 `m | n`。
- **gcd 和 lcm 的关系：** `gcd(m, n) * lcm(m, n) = |m| * |n|`。

模运算与余数

- **定义:** `m div n = ⌊m/n⌋` 和 `m % n = m - n * ⌊m/n⌋`。
- **目标:** 给定整数 `m` 和 `n`，求 `m div n` 和 `m % n`。
- **技巧**：对大数取余时可寻找幂的循环模式，以简化计算。

难点：

例题：求 `7^7^7` 的最后两位。 **解答思路**：通过模 100 获取最后两位，对 `7` 的幂取模，找到循环规律 `74 ≡ 1 (mod 100)`，然后简化问题为 `73 % 100 = 43`。

问题：在 1 到 1000 的整数中，(a) 能被 2, 3 或 5 整除的有多少个？(b) 不能被 2, 3 或 5 整除的有多少个？

解答思路： 定义集合：

- U = {x ∈ ℕ : x ∈ [1, 1000]}
- A = {x : 2 | x}, B = {x : 3 | x}, C = {x : 5 | x}

通过容斥原理计算： |A ∪ B ∪ C| = |A| + |B| + |C| - |A ∩ B| - |B ∩ C| - |C ∩ A| + |A ∩ B ∩ C|

计算： |A| = ⌊1000 / 2⌋ = 500
|B| = ⌊1000 / 3⌋ = 333
|C| = ⌊1000 / 5⌋ = 200
|A ∩ B| = ⌊1000 / 6⌋ = 166
|B ∩ C| = ⌊1000 / 15⌋ = 66
|C ∩ A| = ⌊1000 / 10⌋ = 100
|A ∩ B ∩ C| = ⌊1000 / 30⌋ = 33

- **复合关系 (Composition)**：`R;S = {(a, c) ∈ A × C : 存在 b ∈ B 使 a R b 且 b S c}`。

关系的性质

- **自反性 (Reflexive)**：对于所有 `a ∈ A`，有 `(a, a) ∈ R`。
- **反自反性 (Antireflexive)**：对于所有 `a ∈ A`，有 `(a, a) ∉ R`。
- **对称性 (Symmetric)**：若 `(a, b) ∈ R`，则 `(b, a) ∈ R`。
- **反对称性 (Antisymmetric)**：若 `(a, b) ∈ R` 且 `(b, a) ∈ R`，则 `a = b`。
- **传递性 (Transitive)**：若 `(a, b) ∈ R` 且 `(b, c) ∈ R`，则 `(a, c) ∈ R`。

等价关系与等价类

- **等价关系 (Equivalence Relation)**：关系满足自反性、对称性和传递性。
- **等价类 (Equivalence Class)**：给定 `a ∈ A`，`[a] = {b ∈ A : a R b}`。

最大值、极大元素、最小值、极小元素的关系

极大元素 (Maximal) 与最大值 (Maximum)

1. **极大元素 (Maximal Element)**：在偏序集中，若元素 (a) 没有比它大的元素，则称 (a) 为极大元素。可能不唯一。
2. **最大值 (Maximum)**：最大值是所有元素中绝对最大的一个，即对任意 (b \in S)，都有 (b \leq a)。最大值是唯一的，且必须是极大元素。
3. **关系**：
   - 最大值一定是极大元素。
   - 极大元素未必是最大值。

极小元素 (Minimal) 与最小值 (Minimum)

1. **极小元素 (Minimal Element)**：在偏序集中，若元素 (a) 没有比它更小的元素，则称 (a) 为极小元素。可能不唯一。
2. **最小值 (Minimum)**：最小值是所有元素中绝对最小的一个，即对任意 (b \in S)，都有 (a \leq b)。最小值是唯一的，且必须是极小元素。

格与全序关系

格 (Lattice)

1. **定义**：

- **合取范式 (CNF)**：布尔表达式以**最大项**形式存在，即形如 `(m1 && m2 && ... && mn)` 的表达式。
- **析取范式 (DNF)**：布尔表达式以**最小项**形式存在，即形如 `(m1 || m2 || ... || mn)` 的表达式。
- **例题**：将 `(x || y) && (!x || !y)` 转换为析取范式。
  - **解答思路**：
    1. **列出所有可能输入组合**：构建真值表，找出使表达式为真的组合。
    2. **写出对应的最小项**：对于每个满足条件的组合，写出相应的 `x` 和 `y` 形式的最小项。
    3. **组合最小项**：将所有满足条件的最小项用 `||` 连接起来，即得到析取范式结果 `(!x && y) || (x && !y)`。

大-O 记号

- **O-符号 (Big-O Notation)**：若 `f(n) ∈ O(g(n))`，则存在常数 `n0` 和 `c > 0`，使得对所有 `n ≥ n0`，有 `f(n) ≤ c * g(n)`。
- **Ω-符号 (Big-Omega Notation)**：若 `f(n) ∈ Ω(g(n))`，则存在常数 `n0` 和 `c > 0`，使得对所有 `n ≥ n0`，有 `f(n) ≥ c * g(n)`。
- **Θ-符号 (Big-Theta Notation)**：若 `f(n) ∈ Θ(g(n))`，则 `f(n)` 同时属于 `O(g(n))` 和 `Ω(g(n))`。
- **大-O 符号的对偶关系**：若 `f(n) ∈ O(g(n))`，则 `g(n) ∈ Ω(f(n))`。
- **例题**：确定 `f(n) = 4n + 2` 与 `g(n) = n^2 - 4` 的增长关系。
  - **解答思路**：
    1. **验证 O-记号条件**：找到 `c` 和 `n0` 使得对所有 `n ≥ n0, f(n) ≤ c * g(n)`，从而满足 `f(n) ∈ O(g(n))`。
    2. **验证 Ω-记号条件**：判断是否存在 `c` 和 `n0` 使得 `f(n) ≥ c * g(n)` 成立，若不满足则 `f(n) ∉ Ω(g(n))`。
    3. **综合结论**：若 `f(n)` 满足 `O(g(n))` 但不满足 `Ω(g(n))`，则 `f(n) ∉ Θ(g(n))`。

逻辑符号 (Logical Symbols) 与逻辑运算

逻辑运算的等价组合

- **条件 (→)**：条件 (→)：$p \to q \equiv \neg p \vee q$
- **双向条件 (↔)**：p↔q≡(p∧q)∨(¬p∧¬q)

良构公式 (Well-Formed Formula, WFF)

代入公式得：
|A ∪ B ∪ C| = 500 + 333 + 200 - 166 - 66 - 100 + 33 = 734
因此，不能被 2, 3 或 5 整除的数为 1000 - 734 = 266。

欧几里得算法 (Euclidean Algorithm)

- **算法概念:** 欧几里得算法通过递归计算来求解最大公约数 (GCD)，其规则如下：
  1. 当 `n = 0` 时，`gcd(m, 0) = m`。
  2. 否则，递归使用 `gcd(m % n, n)`，直到余数为 0。
- **两个数互质的条件**：若 `gcd(a, b) = 1`，则称 `a` 和 `b` 为互质数。
- **连续整数的互质性**：任意整数 `n` 与 `n+1` 总是互质，因为其 GCD 必为 1。

集合论

- **差集 (Difference)**：`A \ B = A ∩ Bc`，即 `A` 中不属于 `B` 的元素。
- **对称差 (Symmetric Difference)**：`A ⊕ B = (A \ B) ∪ (B \ A)`，包含仅在 `A` 或 `B` 中的元素。
- **幂集 (Power Set)**：`Pow(A)` 表示所有 `A` 的子集构成的集合。
- **基数 (Cardinality)**：`|A|` 表示集合 `A` 的元素数量。
- **集合互斥性**：`A` 和 `Ac` 互为补集，因此 `A ∩ Ac = ∅` 且 `A ∪ Ac = U`。
- **幂集大小**：若集合 `A` 含有 `n` 个元素，则 `|Pow(A)| = 2^n`。
- **形式语言的运算封闭性**：语言 `A*` 包含所有可能的串联组合，是包含空串的无限集合。

集合律与代数性质

- **分配律 (Distribution)**：`A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)`，`A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)`。
- **德摩根律 (de Morgan's Laws)**：`(A ∩ B)c = Ac ∪ Bc`，`(A ∪ B)c = Ac ∩ Bc`。

形式语言 (Formal Languages)

- **字母表 (Alphabet)**：有限的符号集，通常记作 `Σ`。
- **单词 (Word)**：由 `Σ` 中符号组成的有限序列。
- **语言 (Language)**：由 `Σ*` 中符合某种特定规则的单词集合。
- **闭包运算 (Kleene Star)**：`A*` 表示 `A` 的所有可能串联组合，包括空串。
- **串联 (Concatenation)**：`AB = {ab : a ∈ A and b ∈ B}`。
- **交集的闭包性**：如果 `w ∈ (L1 ∩ L2)*`，则 `w ∈ L1* ∩ L2*`。

难点

  - **最小上界 (Least Upper Bound, lub)**：
    存在 (a) 和 (b) 的公共上界，并且是所有上界中最小的，记为 (a \vee b)。
  - **最大下界 (Greatest Lower Bound, glb)**：
    存在 (a) 和 (b) 的公共下界，并且是所有下界中最大的，记为 (a \wedge b)。

2. **格的性质**：
   - 如果偏序集中任意两元素的 (lub) 和 (glb) 存在，则该偏序集是格。

全序关系 (Total Order)

1. **定义**：
   偏序关系 (\leq) 是全序关系，当且仅当对任意 (a, b \in S)，满足 (a \leq b) 或 (b \leq a)。
2. **全序与格的关系**：
   - 每个全序集都是一个格，因为总能找到 (lub) 和 (glb)。
   - 但格未必是全序集。

拓扑排序 (Topological Sort)

1. **定义**：
   在有向无环图 (DAG) 中，将顶点按照偏序关系排列，满足如果 (u \to v)，则 (u) 在 (v) 之前。
2. **作用**：
   将偏序集扩展为全序排列，便于对集合中的元素进行线性排序。

布尔逻辑

- **结合律**：`(x || y) || z = x || (y || z)`，`(x && y) && z = x && (y && z)`
- **分配律**：`x || (y && z) = (x || y) && (x || z)`，`x && (y || z) = (x && y) || (x && z)`

难点：

- **例题**：化简 `[x && (x && !y)] || [(x && y) || (y && !x)] = x || y`
  - **解答思路**：
    1. **观察整体结构**：将表达式分成几个小部分，尝试应用布尔代数的交换律、结合律、分配律逐步简化。
    2. **应用恒等律和补全律**：将不必要的项去除，使用 `x || (!x) = 1` 和 `x && (!x) = 0` 等基础公式。
    3. **整理结果**：最后得到简化形式 `x || y`。

- **定义**：良构公式 (WFF) 是满足逻辑结构和语法规范的公式，通常由基本命题和逻辑连接词构成。
- **构造规则**：
  - 单个命题变元、`>` (总真命题)、`⊥` (总假命题) 本身就是良构公式。
  - 若 `φ` 和 `ψ` 是良构公式，则 `¬φ`、`φ ∧ ψ`、`φ ∨ ψ`、`φ → ψ` 和 `φ ↔ ψ` 也是良构公式。

真值表 (Truth Table)

- **结构**：每一行表示一种真值组合，列表示每个子公式的真值，最后一列为整个公式的真值。

逻辑等价 (Logical Equivalence)

- **定义**：若公式 `φ` 和 `ψ` 在所有真值赋值下的结果相同，则称 `φ` 和 `ψ` 逻辑等价，记作 `φ ≡ ψ`。
- **证明方法**：
  - **真值表比较**：逐行检查真值表，确保所有组合下 `φ` 和 `ψ` 真值一致。
  - **使用等价律**：使用逻辑等价律直接变换。
  - **构建恒真公式**：证明 `φ ↔ ψ` 恒为真。

蕴涵与有效性 (Entailment and Validity)

- **定义**：蕴涵关系 `φ1, ..., φn |= ψ` 表示在所有满足前提的真值赋值下，结论 `ψ` 也为真。
- **验证方法**：
  - **真值表法**：构建前提和结论的真值表，验证在所有前提为真时，结论也为真。
  - **恒真公式法**：证明 `(φ1 ∧ ... ∧ φn) → ψ` 恒为真。
- **逻辑等价的证明**：可以通过真值表、等价律或构造恒真公式来证明两个命题的等价性。
- **矛盾蕴涵**：在矛盾条件下，可以推导出任意结论，因此矛盾是逻辑上最强的蕴涵条件。
- **例题**：证明 `p ∧ ¬p |= q`。
  - **解题步骤**：
    1. **观察矛盾**：`p ∧ ¬p` 本身构成矛盾，逻辑上为假。
    2. **推导**：在矛盾条件下，可以推导出任意结论，因此 `p ∧ ¬p` 可以蕴涵任何命题 `q`。

逻辑等价律 (Logical Equivalence Laws)

- **例题**：证明 `(L1 ∩ L2)* ⊆ L1* ∩ L2*`
  - **解答思路**：
    1. 若 `w ∈ (L1 ∩ L2)*`，则 `w = w1w2...wn`，其中每个 `wi ∈ L1 ∩ L2`。
    2. 因为 `wi` 同时属于 `L1` 和 `L2`，因此 `w ∈ L1*` 且 `w ∈ L2*`，即 `w ∈ L1* ∩ L2*`。

关系

二元关系的定义

- **关系 (Relation)**：集合 `A` 和 `B` 之间的关系 `R` 是 `A × B` 的子集。如果 `(a, b) ∈ R`，则称 `a R b`。
- **图表示法**：用点代表元素，用箭头表示关系。将关系绘制成带箭头的图。
- **矩阵表示法**：在矩阵中用 `•` 表示关系 `(a, b) ∈ R`。 **例题**：设集合 `A = {1, 2, 3}` 和 `B = {a, b}`，定义关系 `R ⊆ A × B` 满足以下条件：
- `(1, a), (2, a), (2, b)` 为 `R` 的元素。

我们可以将 `R` 表示成如下的矩阵和图：

矩阵表示

	a	b
1	•	
2	•	•
3		

- 其中 `•` 表示 `(a, b) ∈ R`。
- 行代表集合 `A` 的元素，列代表集合 `B` 的元素。

图表示

```
A   B

1 → a
2 → a
2 → b
```

- 图表示法中，箭头从 `A` 中的元素指向 `B` 中的元素，以表示关系的方向。

关系的运算

- **逆关系 (Converse)**：`R← = {(b, a) ∈ B × A : a R b}`。

卡诺图 (Karnaugh Map)

使用规则：

- 将布尔表达式中的每种变量组合在卡诺图上标出。
- 通过合并相邻的"1"形成方块（覆盖区域），每个方块大小必须为 `2^n` (如1、2、4个格子)。
- 方块可沿卡诺图边界环绕，找到覆盖所有"1"所需的最少方块数量。

例题

- **题目**：将 `(x̄y) ∨ (xy)` 转换为最小项数量的析取范式 (DNF)。
- **解题步骤**：

1. **构建卡诺图**：将各项 `(x̄y)` 和 `(xy)` 的对应格子标记为"1"。
2. **合并项**：观察相邻的"1"，合并成一个方块，尽量减少表达式中的项数。
3. **写出最简表达式**：最终得到最小项形式。

函数与其性质

- **函数 (Function)**：对于集合 `X` 到 `Y` 的二元关系 `f ⊆ X × Y`，若每个 `x ∈ X` 对应唯一的 `y ∈ Y`，则 `f` 是一个函数。
- **单射 (Injective)**：若 `f(a) = f(b)` 则 `a = b`。
- **满射 (Surjective)**：对于每个 `y ∈ Y`，存在 `x ∈ X` 使得 `f(x) = y`。
- **双射 (Bijective)**：若函数既单射又满射。
- **例题**：判断给定的二元关系是否为函数，并确定其定义域、值域和映射结果。
  - **解答思路**：
    1. **检查唯一性**：确保每个 `x ∈ X` 都对应唯一的 `y ∈ Y`。
    2. **确定函数的性质**：根据映射关系判断其是否为单射或满射。
    3. **求解定义域、值域和映射结果**：确认 `Dom(f)`、`Codom(f)` 和 `Im(f)`，理解函数的作用范围。

复合函数与逆函数

- **复合函数 (Composition)**：若 `f : X → Y` 和 `g : Y → Z`，则 `(g ∘ f)(x) = g(f(x))`。
- **逆函数 (Inverse Function)**：若 `f` 为双射，则存在唯一的逆函数 `f−1`，满足 `f−1(f(x)) = Idx` 且 `f(f−1(x)) = Idx`。

主范式：合取范式 (CNF) 和析取范式 (DNF)

- **分配律 (Distribution)**：`p ∨ (q ∧ r) ≡ (p ∨ q) ∧ (p ∨ r)`，`p ∧ (q ∨ r) ≡ (p ∧ q) ∨ (p ∧ r)`
- **逆否命题 (Contrapositive)**：`p → q ≡ ¬q → ¬p`
- **德摩根律 (De Morgan's Laws)**：`¬(p ∨ q) ≡ ¬p ∧ ¬q`，`¬(p ∧ q) ≡ ¬p ∨ ¬q`

递归 (Recursion)

基础概念

递归包含以下两部分：

1. **基础情况 (Basis)**：定义初始项。
2. **递归关系 (Recurrence)**：通过前面的项定义后续项。

数学归纳法 (Mathematical Induction)

基础概念

归纳法用于证明命题 P(n) 对所有 n 成立。步骤如下：

1. **基础情况 (Base Case)**：验证 P(m) 成立。
2. **归纳假设 (Inductive Hypothesis)**：假设 P(k) 成立。
3. **归纳步骤 (Inductive Step)**：证明 P(k) ➞ P(k+1)。

主定理 (Master Theorem)

基础概念

对于递归式T(n) = a * T(n / b) + f(n) 设 (d = log_b a)，则：

- **Case 1**：若 (c < d)，则 T(n) = Theta(n^d)。
- **Case 2**：若 (c = d)，则 T(n) = Theta(n^c \log n)。
- **Case 3**：若 (c > d)，则 T(n) = Theta(f(n)) 。

问题：求解以下递归的渐近复杂度：

1. T(n) = 8T(n/2) + 6n^3
2. T(n) = 4T(n/5) + 2\log n
3. T(n) = T(n/2) + sqrt{n}\log n

解答思路：

1. (a = 8, b = 2, c = 3)，有 (d = \log_2 8 = 3)。因为 (c = d)，故 (T(n) = \Theta(n^3 \log n))。
2. (a = 4, b = 5, c = 0)，有 (d = \log_5 4)。因为 (c < d)，故 (T(n) = \Theta(n^{\log_5 4}))。

3. (a = 1, b = 2, c = 0.5)，有 (d = 0)。因为 (c > d)，故 (T(n) = \Theta(\sqrt{n}\log n))。

结构归纳法 (Structural Induction)

基础概念

结构归纳法是一种针对递归定义的结构，证明其性质的数学方法。基本步骤如下：

1. **基础情况**：验证所有基础结构的命题成立。
2. **递归情况**：假设前序结构的命题成立，证明递归结构的命题也成立。

问题：证明对于所有 w 属于 Σ*，有 length(rev(w)) = length(w)，其中 rev(w) 是 w 的逆序。 **解答思路**：

1. **基础情况**：
   如果 w = λ（空字符串），则 length(rev(λ)) = length(λ) = 0，命题成立。
2. **递归情况**：
   假设对于某字符串 u，命题 length(rev(u)) = length(u) 成立。
   对于 w = xu（其中 x 是单个字符，u 是字符串），需要证明命题对 w 成立：
   length(rev(xu)) = length(rev(u)x) = length(rev(u)) + 1 = length(u) + 1 = length(xu)。
   因此，递归情况成立。 **结论**：通过结构归纳法，命题对所有 w 属于 Σ* 成立。

组合数学 (Combinatorics)

集合的基本计数规则

基础公式

1. |X \ Y| = |X| - |X ∩ Y|
2. |X ∪ Y| = |X| + |Y| - |X ∩ Y|
3. |X ∪ Y ∪ Z| = |X| + |Y| + |Z| - |X ∩ Y| - |Y ∩ Z| - |Z ∩ X| + |X ∩ Y ∩ Z|

排列与组合

1. 排列数：
   P(n, r) = n! / (n - r)!
2. 组合数：
   C(n, r) = n! / [r! * (n - r)!]

例题 2

问题：将 20 个相同的气球和 32 个不同的棒棒糖分配给 4 个孩子，要求：

1. 每个孩子至少获得 1 个气球和 8 个棒棒糖。
2. 总共有多少种分配方式?

2. **邻接矩阵**：

```
0 1 0 0 0
1 0 1 1 0
0 1 0 1 0
0 1 1 0 0
0 0 0 0 0
```

3. **邻接表**：

```
a: b
b: a, c, d
c: b, d
d: b, c
e:
```

4. **关联矩阵**：

```
1 0 0 0
1 1 0 1
0 1 1 0
0 0 1 1
0 0 0 0
```

度数

1. **无向图**：顶点 v 的度数 deg(v) 为与其相连的边数。
   - 性质：所有顶点度数之和等于边数的两倍，即 Σ deg(v) = 2 |E|。
2. **有向图**：分为入度 indeg(v) 和出度 outdeg(v)。
   - 性质：所有顶点的入度和等于出度和，总和等于边数，即 Σ indeg(v) = Σ outdeg(v) = |E|。

路径与连通性

1. **步 (Walk)**：顶点序列 v0, v1, ..., vn，边 (v(i-1), v(i)) 属于 E。
2. **轨迹 (Trail)**：步中无重复边。
3. **简单路径 (Path)**：步中无重复顶点。
4. **闭环 (Cycle)**：首尾相连的简单路径。

问题：判断以下顶点序列的路径类型：

1. a → b → c → d → b → e

Hamilton 路径与回路

- **Hamilton 路径**：经过每个顶点一次。
- **Hamilton 回路**：闭合的 Hamilton 路径。
  - 无通用判定方法。

图染色与平面性

染色

- **定义**：为图的顶点分配颜色，要求相邻顶点颜色不同。
- **色数 (χ(G))**：最少颜色数。

平面图

- **定义**：图可以在平面上绘制，且无边交叉。
- **判定**：图中不包含 K5 或 K3,3 的子图。

例题 5

问题：证明以下图为非平面图。

- 图中包含 K5 的子图。

解答： 通过找到 K5 或 K3,3 的子图即可证明非平面性。

随机变量与期望值

随机变量

1. 随机变量 X：将样本空间 Ω 映射到整数或实数。
   - 示例：X 表示硬币正面出现的次数。
2. 常见操作：
   - 加法：X + Y = X(ω) + Y(ω)
   - 乘法：X * Y = X(ω) * Y(ω)

期望值

定义： E(X) = ∑ P(X = k) * k

- 线性性质：E(aX + bY) = aE(X) + bE(Y)
- 独立同分布时：E(X1 + ... + Xn) = n * E(X1)

问题：从一副 52 张牌中抽取一张，定义随机变量： X = 5, 若是黑桃； -2, 若是红心； 0, 其他。 求 E(X)。 **解答**：

1. 概率： P(黑桃) = 13/52 = 1/4， P(红心) = 1/4， P(其他) = 1/2。

解答思路：

1. 对于气球的分配：
   - 去掉每人 1 个气球后分配剩余的 20 - 4 = 16 个气球。
   - 使用"组合有放回"的公式：
     C(16 + 4 - 1, 16) = C(19, 16)。
2. 对于棒棒糖的分配：
   - 每人获得 8 个棒棒糖。
   - 第一位有 C(32, 8) 种选法，第二位有 C(24, 8)，以此类推。

最终答案：
C(19, 16) * C(32, 8) * C(24, 8) * C(16, 8) * C(8, 8)

---

1. **排列与组合的基本分类**
   排列与组合问题通常需要判断以下两点：
   - 是否允许重复抽取元素 (With Replacement)。
   - 抽取的顺序是否重要 (Order Matters)。

分类公式汇总

With Replacement	Order Matters	公式
Yes	Yes	n^k
No	Yes	P(n, k) = n! / (n - k)!
No	No	C(n, k) = n! / [k! * (n - k)!]
Yes	No	C(n + k - 1, k)

重要公式总结

- **排列公式**：
  P(n, k) = n! / (n - k)!，适用于无重复且顺序重要的场景。
- **组合公式**：
  C(n, k) = n! / [k! * (n - k)!]，适用于无重复且顺序不重要的场景。
- **重复组合公式**：
  C(n + k - 1, k)，适用于允许重复且顺序不重要的场景。
- **全排列公式**：
  n!，适用于无重复顺序的重要排列。

2. a → b → c → d → c → a
3. a → c → e **解答**：
4. **walk**：重复顶点和边，非轨迹或路径。
5. **cycle**：是一个首尾相连的简单路径。
6. **path**：没有重复顶点。

树

- **定义**：无环连通图，满足 |V| = |E| + 1。
- **叶子**：度数为 1 的顶点。

问题：在一棵树中，叶子是度数为 1 的顶点。设 Δ 为树 T 的最大度数。证明：树中至少有 Δ 个叶子。

解答：

设树 T 有 n 个顶点和 k 个叶子。我们将顶点按照度数递增排序：

- 顶点 v1, ..., vk 是 k 个叶子，度数为 1；
- 顶点 vk+1, ..., vn-1, vn 是剩余的非叶子顶点，其度数至少为 2；
- 设 vn 的度数为 Δ，即树的最大度数。

分析：

1. 根据树的性质，树的边数为：

```
|E| = n - 1
```

   因为树是一种连通无环图。

2. 所有顶点的度数和满足：

```
Σ(deg(vi)) = 2 × |E| = 2(n - 1)
```

3. 将顶点的度数和分为叶子和非叶子两部分：

```
Σ(deg(vi)) = Σ(deg(vi)) (i = 1 到 k) + Σ(deg(vi)) (i = k+1 到 n-1) + deg(vn)
```

4. 对于叶子顶点：

```
Σ(deg(vi)) (i = 1 到 k) = k
```

2. 期望值： E(X) = 5 * (1/4) + (-2) * (1/4) + 0 * (1/2) = 3/4。

二项分布

对于 n 次独立实验，每次成功概率为 p，随机变量 X 表示成功次数，则： P(X = k) = C(n, k) * p^k * (1-p)^(n-k) 期望值与方差： E(X) = n * p, Var(X) = n * p * (1-p)

问题：某过滤器对垃圾邮件的识别率为 98%。每天收到 100 封邮件，其中 60% 是垃圾邮件。

1. 每天漏判垃圾邮件的期望值是多少?
2. 超过 3 封垃圾邮件漏判的概率是多少?

解答：

1. **期望值**： E(X) = n * p = 60 * 0.02 = 1.2 封。
2. **概率计算**：
   - 随机变量 X：漏判垃圾邮件数，服从 B(60, 0.02)。
   - P(X > 3) = 1 - P(X ≤ 3)，用二项分布计算 P(X ≤ 3)。

正态分布近似

大样本情况下，二项分布 B(n, p) 可用正态分布 N(μ, σ²) 近似： μ = n * p, σ² = n * p * (1-p)

问题：二进制通信信道误码率为 0.1%，每个数据包包含 1000 位。

1. 每包期望错误数是多少?
2. 超过 3 位错误的概率是多少?

解答：

1. **期望值**： μ = n * p = 1000 * 0.001 = 1。
2. **概率计算**： 用正态分布近似，σ² = n * p * (1-p) = 0.999，标准化后进行概率计算。

算法分析

时间复杂度分析

基本操作

- 算术操作、比较、赋值、返回值：O(1)

循环时间复杂度

- 单层循环：T(n) = b * O(1)。
- 嵌套循环： T(n) = ∑ O(i) = O(n^2)

离散概率论 (Discrete Probability)

1. **样本空间 (Sample Space)**：所有可能结果的集合，记作 Ω。
2. **事件 (Event)**：样本空间的子集。
3. **概率分布 (Probability Distribution)**：
   - P(Ω) = 1
   - 对于互斥事件 A 和 B，有 P(A ∪ B) = P(A) + P(B)。

问题：两个骰子同时掷出：

1. 骰子点数之和为 6 的概率是多少?
2. 至少有一个骰子出现偶数的概率是多少?

解答思路：

1. 样本空间大小为 |Ω| = 6 × 6 = 36。
   - 点数和为 6 的组合为 {(1, 5), (2, 4), (3, 3), (4, 2), (5, 1)}，共有 5 种。
   - 概率为 P = 5 / 36。
2. 至少一个偶数：
   - 两个骰子全是奇数的概率为 P(全奇) = (3 / 6) × (3 / 6) = 1 / 4。
   - 至少一个偶数的概率为 P(至少一个偶数) = 1 - P(全奇) = 3 / 4。

条件概率与独立性

1. 条件概率公式 P(A | B) = P(A ∩ B) / P(B)
2. 独立性 若 P(A ∩ B) = P(A) × P(B)，则事件 A 和 B 独立。

图论 Cheat Sheet

图的表示方法

1. **图形表示**：顶点用点表示，边用线连接。
2. **邻接矩阵**：矩阵元素 (i, j) 为 1 表示顶点 i, j 有边相连。
3. **邻接表**：每个顶点对应一个列表，列出与其相连的顶点。
4. **关联矩阵**：顶点为行，边为列，矩阵值表示顶点是否关联某条边。

问题：无向图 G = (V, E)，其中 V = {a, b, c, d, e}，E = {{a, b}, {b, c}, {c, d}, {b, d}}。

解答：

1. **图形**：

```
a
|
b -- c
|    |
d    e
```

5. 对于非叶子顶点：

```
Σ(deg(vi)) (i = k+1 到 n-1) ≥ 2 × (n - k - 1) （每个非叶子顶点的度数至少为 2）
```

6. 最大度数顶点 vn 的度数为：

```
deg(vn) = Δ
```

合并计算：

将上述结果代入总度数公式：

```
2(n - 1) = k + Σ(deg(vi)) (i = k+1 到 n-1) + Δ
```

使用非叶子顶点的下界：

```
2(n - 1) ≥ k + 2 × (n - k - 1) + Δ
```

化简：

```
2(n - 1) ≥ 2(n - 1) - k + Δ

k ≥ Δ
```

完全图与二分图

1. **完全图 (Kn)**：所有顶点两两相连，边数 |E| = n(n-1)/2。
2. **完全二分图 (Km,n)**：两部分顶点间所有顶点相连，边数 |E| = m × n。

Euler 和 Hamilton 问题

Euler 路径与回路

- **Euler 路径**：包含图中所有边，且无重复。
  - 条件：所有顶点度数为偶数或仅有 2 个顶点度数为奇数。
- **Euler 回路**：闭合的 Euler 路径。
  - 条件：所有顶点度数为偶数。

代码： nested_power(n): for i in range(1, n+1): for j in range(1, i+1): result += j

分析：

1. 外层循环 i = 1 → n，内层循环运行 i 次。
2. 总时间复杂度为 ∑ i = n(n+1)/2 = O(n^2)。

递归时间复杂度

递归复杂度可用递推式表示，常用主定理解决： T(n) = a * T(n/b) + f(n)

根据 d = log_b a 比较 f(n) 增长：

1. 若 f(n) = O(n^d)，则 T(n) = O(n^d * log n)。
2. 若 f(n) < O(n^d)，则 T(n) = O(n^d)。

例题：递归求解

代码： bad_split(n): if n <= 0: return 0 elif n % 2 == 1: return 1 + bad_split(n // 2) else: return 1 + bad_split(n // 4)

解答：

1. 递推式：T(n) = T(n/2) + O(1)
2. 解法：用主定理得 T(n) = O(log n)。