CSP-J 初赛图论初步

一、图的基本概念

1. **图的定义**

   图是由顶点集（V）和边集（E）组成的二元组 $G=(V,E)$。其中，顶点（Vertex）是图的基本元素，边（Edge）是连接两个顶点的关系。边分为有向边和无向边：无向边用无序对（(u,v)）表示，有向边用有序对（<u,v>）表示（（u）为起点，（v）为终点）。

2. **图的分类**
   - **无向图**：所有边均为无向边，任意两顶点间的边没有方向。
   - **有向图**：至少包含一条有向边，边具有明确的方向。
   - **简单图**：不含重复边（两个顶点间最多一条边）和自环（顶点到自身的边）的图。
   - **完全图**：无向完全图中任意两顶点间均有一条边，含（n）个顶点的无向完全图有$(\frac{n(n-1)}{2})$条边；有向完全图中任意两顶点间有两条方向相反的边，含（n）个顶点的有向完全图有（n(n-1)）条边。
3. **顶点的度**
   - **无向图**：顶点（v）的度（\text{deg}(v)）是与（v）相连的边数。所有顶点的度之和等于边数的 2 倍（握手定理）。
   - **有向图**：顶点（v）的入度（\text{in-deg}(v)）是指向（v）的边数，出度（\text{out-deg}(v)）是从（v）出发的边数。所有顶点的入度之和等于出度之和，且等于边数。

二、图的存储结构

1. **邻接矩阵（Adjacency Matrix）**
   - 用$(n\times n)$的二维数组（A）表示图，其中（A[i][j]）表示顶点（i）与（j）之间的边关系：
     - 无向图：（A[i][j] = A[j][i] = 1）（若存在边），否则为 0；
     - 有向图：（A[i][j] = 1）（若存在有向边（<i,j>）），否则为 0；
     - 带权图：（A[i][j]）存储边的权重，不存在边时用(∞)或 0 表示（视场景而定）。
   - 优点：查询两顶点间是否有边的时间复杂度为（O(1)）；
   - 缺点：空间复杂度为$(O(n^2))$，适用于稠密图（边数接近完全图）。
2. **邻接表（Adjacency List）**
   - 用数组+链表（或动态数组）存储，数组每个元素对应一个顶点，其链表存储该顶点的所有邻接顶点（及边权）：

        - 无向图：每个边会在两个顶点的链表中各出现一次；
        - 有向图：每个有向边仅在起点的链表中出现。
    - 优点：空间复杂度为 $O(n + m)$（m 为边数），适用于稀疏图；
    - 缺点：查询两顶点间是否有边需遍历链表，时间复杂度为 $O(\text{deg}(v))$。

三、图的遍历算法

1. **深度优先搜索（DFS）**
    - **思想**：从起点出发，沿一条路径尽可能深地遍历，直到无法前进时回溯，选择未访问的分支继续。
    - **实现**：递归或栈（非递归），需记录顶点访问状态（访问数组 visited）。
    - **应用**：判断连通性、求路径、拓扑排序（有向无环图）、找强连通分量等。
2. **广度优先搜索（BFS）**
    - **思想**：从起点出发，优先访问所有距离为 1 的邻接顶点，再访问距离为 2 的顶点，以此类推（层次遍历）。
    - **实现**：队列，需记录顶点访问状态和距离（或前驱节点）。
    - **应用**：求无权图的最短路径、判断连通性、层次遍历等。
3. **遍历示例**
    - 对无向图 $G = (V, E)$，DFS 和 BFS 均可遍历所有连通顶点；若图不连通，则需对每个未访问的顶点重复遍历（称为“连通分量遍历”）。

四、图的基本应用

1. **连通性问题**
    - **连通图**：无向图中任意两顶点间存在路径；否则为非连通图，其极大连通子图称为连通分量。
    - **强连通图**：有向图中任意两顶点 (u, v)，均存在从 u 到 v 和从 v 到 u 的路径；极大强连通子图称为强连通分量。
    - **判断方法**：DFS/BFS 从任一顶点出发，若能访问所有顶点则为连通图（无向）；强连通分量可通过 Kosaraju 算法（两次 DFS）或 Tarjan 算法（一次 DFS 结合栈）求解。
2. **最短路径**
    - **无权图**：BFS 遍历，首次访问顶点时的路径即为最短路径（边数最少）。
    - **带权图（非负权）**：Dijkstra 算法，通过贪心策略逐步更新起点到各顶点的最短路径，时间复杂度 $(O(m \log n))$（堆优化）。
    - **带负权图**：Floyd-Warshall 算法（动态规划，时间复杂度 $(O(n^3))$）或 Bellman-Ford 算法（可检测负环）。
3. **拓扑排序（Topological Sort）**
    - **适用场景**：有向无环图（DAG），用于解决依赖关系排序问题（如任务调度）。
    - **算法步骤**：
        1. 计算所有顶点的入度，将入度为 0 的顶点入队；
        2. 依次出队顶点 u，输出 u 并删除其所有出边，更新邻接顶点的入度，若入度为 0 则入队；

3. 若输出顶点数小于总顶点数，则图中存在环，无法拓扑排序。

五、初赛常见考点与题型

1. **概念辨析**：判断图的类型、计算顶点度、握手定理应用（如已知各顶点度求边数）。
2. **存储结构对比**：根据图的稀疏/稠密程度选择邻接矩阵或邻接表，分析空间复杂度。
3. **遍历序列**：给定图的结构，写出 DFS/BFS 的遍历序列（注意起点和邻接顶点的访问顺序）。
4. **拓扑排序**：判断给定序列是否为拓扑序，或根据图求拓扑序列。
5. **最短路径计算**：针对简单带权图（如 3-4 个顶点），手动模拟 Dijkstra 算法或 BFS 求最短路径。
6. **连通性分析**：判断无向图的连通分量个数，或有向图的强连通分量个数。

六、CSP-S 初赛图论知识点解析

1. 图的基本概念

- **图的定义**：由顶点集（V）和边集（E）组成，记为 $G = (V, E)$。顶点（节点）是图的基本元素，边是连接顶点的关系。
- **有向图与无向图**：边有方向的图为有向图（边表示为 (<u,v>)），无方向的为无向图（边表示为 ((u,v))）。
- **度的概念**：
  - 无向图中，顶点（v）的度是与（v）关联的边数，记为 (\deg(v))。所有顶点的度之和等于边数的 2 倍（握手定理）。
  - 有向图中，顶点（v）的入度 (\text{in-deg}(v)) 是指向（v）的边数，出度 (\text{out-deg}(v)) 是从（v）出发的边数。所有顶点的入度之和等于出度之和，且等于边数。
- **子图与连通分量**：
  - 子图是顶点集和边集均为原图子集的图。
  - 无向图中，连通分量是最大的连通子图（任意两顶点间存在路径）；有向图中，强连通分量是最大的子图，其中任意两顶点互相可达。

2. 图的存储结构

- **邻接矩阵**：用二维数组（A）表示，(A[i][j] = 1)（或权值）表示存在边 ((i,j))，否则为 0（或无穷大）。空间复杂度($O(n^2)$)，适合稠密图。
- **邻接表**：每个顶点关联一个链表，存储与之相邻的顶点及边信息。空间复杂度 (O(n+m))（(n) 为顶点数，(m) 为边数），适合稀疏图。

3. 图的遍历算法

- **深度优先搜索（DFS）：**
  - 从起点出发，递归访问未 visited 的邻接顶点，直到无法前进时回溯。
  - 应用：判断连通性、寻找路径、拓扑排序（有向无环图）、求强连通分量（Kosaraju 算法、Tarjan 算法）。
- **广度优先搜索（BFS）：**
  - 从起点出发，按层次访问所有未 visited 的邻接顶点，使用队列实现。
  - 应用：求无权图最短路径、判断连通性、层次遍历。

4. 最短路径问题

- **单源最短路径：**
  - **Dijkstra 算法**：适用于非负权图。通过优先队列选择当前距离最短的顶点，松弛其邻接顶点的距离。时间复杂度($O(m\log n)$)（堆优化）。
  - **Bellman-Ford 算法**：适用于含负权边的图，可检测负环。通过 (n-1) 轮松弛所有边，时间复杂度 (O(nm))。
- **多源最短路径：**
  - **Floyd-Warshall 算法**：基于动态规划，通过中间顶点更新任意两点间的最短路径。时间复杂度($O(n^3)$)，空间复杂度($O(n^2)$)。

5. 最小生成树（MST）

- **定义**：无向连通图中，权值之和最小的生成树（包含所有顶点，边数为 (n-1)，无环）。
- **算法：**
  - **Kruskal 算法**：按边权值从小到大排序，用并查集（Union-Find）选择不形成环的边，直到选够 (n-1) 条边。时间复杂度($O(m\log m)$)（排序耗时）。
  - **Prim 算法**：从起点开始，通过优先队列选择连接树内外的最小权边，加入生成树。时间复杂度($O(m\log n)$)（堆优化），适合稠密图。

6. 拓扑排序

- **适用场景**：有向无环图（DAG），输出顶点序列，使得所有有向边从序列中靠前的顶点指向靠后的顶点。
- **算法：**
  1. 计算所有顶点的入度，入度为 0 的顶点入队。
  2. 出队顶点 (u)，加入结果序列，遍历其邻接顶点 (v)，将 (v) 的入度减 1；若 (v) 入度为 0，入队。
  3. 重复步骤 2，若结果序列长度小于顶点数，则图中存在环。

7.欧拉回路与欧拉图

一、基本定义

1. **欧拉回路**：在一个连通图中，若存在一条回路，使得图中每条边恰好被遍历一次，且起点与终点为同一个顶点，则称该回路为欧拉回路。
2. **欧拉图**：具有欧拉回路的图称为欧拉图。

二、判定条件（无向图）

1. **连通性**：图必须是连通的（无孤立顶点，任意两点间存在路径）。
2. **度数条件**：图中所有顶点的度数均为偶数。
    - 证明思路：回路中每个顶点进入与离开次数相等，故度数必为偶数；反之，若所有顶点度数为偶数且连通，则可通过构造法形成回路。

三、判定条件（有向图）

1. **连通性**：图必须是弱连通的（忽略边的方向后为连通图）。
2. **入度与出度条件**：
    - 所有顶点的入度等于出度；
    - （若存在欧拉通路但非回路，则允许有且仅有一个顶点出度比入度多 1（起点），一个顶点入度比出度多 1（终点），其余顶点入度等于出度）。

四、历史背景

由瑞士数学家莱昂哈德·欧拉（Leonhard Euler）于 1736 年研究**哥尼斯堡七桥问题**时首次提出。七桥问题中，4 个顶点的度数均为奇数，故不存在遍历每座桥一次的回路，直接证明了问题无解，开创了图论研究的先河。

五、应用场景

1. **一笔画问题**：判断图形能否不重复地一笔画出（欧拉图可一笔画且回到起点）。
2. **网络路由优化**：如邮递员送信路线规划，确保每条街道仅经过一次。
3. **数据传输**：设计环形网络中数据包的高效传输路径，避免重复处理。
4. **工业流程**：生产线工序优化，减少设备重复调度。

六、算法实现（Hierholzer 算法）

1. **步骤**：
    - 从任意顶点出发，深度优先遍历所有边，删除已访问边，若无法继续则回溯并记录顶点。
    - 回溯过程中，将顶点加入结果栈，最终栈中元素逆序即为欧拉回路。
2. **复杂度**：时间复杂度为 $O(V + E)$，其中 V 为顶点数，E 为边数，适用于大规模

图计算。

8. 关键路径

- **定义**：DAG 中，从源点到汇点的最长路径，决定整个工程的最短完成时间。
- **求解步骤**：
  1. **正向遍历**：计算各顶点的最早开始时间（ve[i]）（源点（ve[0] = 0），（ve[v] = \max(ve[u] + weight(u,v))）。
  2. **反向遍历**：计算各顶点的最晚开始时间（vl[i]）（汇点（vl[n-1] = ve[n-1]），（vl[u] = \min(vl[v] - weight(u,v))）。
  3. 关键活动：边（(u,v)）满足（ve[u] + weight(u,v) = vl[v]），关键路径由关键活动组成。

9. 常见考点与易错点

- **图的性质**：无向图连通的必要条件是边数($m \geq n-1$)；完全图的边数：无向图（n(n-1)/2），有向图（n(n-1)）。
- **算法复杂度分析**：区分不同存储结构下遍历、最短路径、最小生成树算法的时间复杂度。
- **并查集应用**：Kruskal 算法中判断环、连通分量计数，需掌握路径压缩和按秩合并优化（时间复杂度近似（O(1)））。
- **特殊图处理**：二分图（染色法判断）、欧拉图（所有顶点度为偶数的连通无向图，存在欧拉回路）、哈密顿图（存在经过所有顶点的回路，无多项式时间算法）。

典型例题思路

- **判断连通性**：使用 DFS/BFS 或并查集，检查所有顶点是否可达。
- **最短路径变种**：如限制路径长度、包含负权边时选择 Bellman-Ford 而非 Dijkstra。

生成树计数：Kirchhoff 定理（利用拉普拉斯矩阵的余子式计算），初赛较少涉及，但需了解基本概念。

七、例题解析

例 1：无向图有 5 个顶点，各顶点度分别为 1, 2, 3, 4, 5，该图是否存在？
解析：根据握手定理，所有顶点度之和必须为偶数。此处总和为（1+2+3+4+5=15）（奇数），故不存在。
例 2：有向图的邻接表中，每个顶点的链表节点数之和等于什么？
解析：有向图邻接表中，每个链表存储顶点的出边，故所有链表节点数之和等于边数。

例 3：对下图（顶点 0-3，边 0-1, 0-2, 1-3, 2-3）进行 BFS（起点 0），写出遍历序列。
解析：BFS 层次访问：0（距离 0）→ 1, 2（距离 1）→ 3（距离 2），遍历序列为 0,1,2,3（或 0,2,1,3，取决于邻接顶点顺序）。

附：CSP 初赛图论题：

1.CSP - J（入门级）初赛图论选择题

基础概念类

1. 一个无向图有 8 个顶点，若要保证该图是连通图，至少需要（ ）条边。

   A. 6

   B. 7

   C. 8

   D. 9

   答案：B。对于无向连通图，要保证其连通，边数最少的情况是树，树的边数等于顶点数减 1，即 8 - 1 = 7 条边。

2. 以下关于图的说法，正确的是（ ）

   A. 有向图中所有顶点的入度之和等于出度之和

   B. 无向图的邻接矩阵一定是对称矩阵

   C. 一个图中不可能同时存在欧拉路径和欧拉回路

   D. 图的连通分量是指图中的极大连通子图

   答案：ABD。在有向图中，每条边对应一个入度和一个出度，所以所有顶点的入度之和等于出度之和；无向图中若存在边 (u, v) 则必有边 (v, u)，所以邻接矩阵一定对称；一个图若存在欧拉回路（所有顶点度为偶数），也可看作特殊的欧拉路径；图的连通分量就是图中的极大连通子图。而当图所有顶点度为偶数时，既存在欧拉回路也可看作存在欧拉路径。

图的存储类

3. 对于一个有 100 个顶点，1000 条边的稀疏图，使用（ ）存储更节省空间。

   A. 邻接矩阵

   B. 邻接表

   C. 两者一样

   D. 无法确定

   答案：B。邻接矩阵的空间复杂度是 $O(n^2)$，对于 100 个顶点，需要 $100 \times 100 = 10000$ 的空间；而邻接表的空间复杂度是 $O(n+e)$，这里 $n = 100$，$e = 1000$，只需要 1100 左右的空间，所以稀疏图用邻接表存储更节省空间。

图的遍历类

4. 对一个无向图进行深度优先搜索（DFS），如果从顶点 v 开始，访问到的第一个顶点序列是 v, v1, v2, v3，那么以下说法正确的是（ ）

   A. 顶点 v 和 v1 之间一定有边相连

   B. 顶点 v2 和 v3 之间一定有边相连

   C. 该图一定是连通图

   D. 深度优先搜索的顺序是唯一的

   答案：A。深度优先搜索从起始顶点开始，沿着一条路径尽可能深地访问，所以起始顶点 v 和第一个访问的邻居顶点 v1 之间一定有边相连；访问到 v2 和 v3 不意味着它们之间一定有边；仅从这一次 DFS 访问序列不能确定图是连通的；DFS 的顺序不是唯一的，取决于邻接顶点的访问顺序。

2.CSP - S（提高级）初赛图论选择题

欧拉图类

1. 一个有向图，其顶点的入度和出度情况如下：顶点 A 入度为 2，出度为 2；顶点 B 入度为 3，出度为 3；顶点 C 入度为 1，出度为 2；顶点 D 入度为 2，出度为 1。该图（ ）

A. 存在欧拉回路

B. 存在欧拉路径

C. 既不存在欧拉回路也不存在欧拉路径

D. 无法判断

答案：B。有向图存在欧拉路径的条件是图弱连通，且存在一个顶点出度比入度多 1（起点），一个顶点入度比出度多 1（终点），其余顶点入度等于出度。这里顶点 C 出度比入度多 1，顶点 D 入度比出度多 1，所以存在欧拉路径。

图的综合性质类

2. 已知一个无向图 G 有 n 个顶点，m 条边，且 m > n - 1。则以下说法正确的是（ ）

A. 图 G 一定是连通图

B. 图 G 一定存在环

C. 图 G 的连通分量个数一定小于 n

D. 图 G 一定是完全图

答案：B。根据树的性质，无向树的边数等于顶点数减 1，当边数 $m > n - 1$ 时，图中一定存在环；$m > n - 1$ 不能保证图一定连通，可能有多个连通分量；连通分量个数不一定小于 n，有可能还是 n 个孤立顶点加上一些边；也不一定是完全图，完全图的边数是 $n(n-1)/2$。

图的算法复杂度类

3. 对于一个有 n 个顶点，e 条边的图，使用广度优先搜索（BFS）遍历该图的时间复杂度是（ ）

A. $O(n)$

B. $O(e)$

C. $O(n+e)$

D. $O(n \times e)$

答案：C。BFS 遍历图时，需要访问每个顶点一次（时间复杂度 $O(n)$），并且检查每条边一次（时间复杂度 $O(e)$），所以总的时间复杂度是 $O(n+e)$。