EDA 软件设计 i Assignment 1

你的姓名、学号 (1 分)

Hard Deadline: 2024-11-23（13:00）

关于拖延症，了解一下帕金森定律（Parkinson's Law）：“工作会扩展到足以填满所有可用的时间。” In other words, “你有多少时间，就会花多少时间去做一件事 (而跟这件事本身真正需要多少时间无关)。”

奖励前 10% **提交的朋友：总分 = 自身得分的** 105%

提交格式:PDF, 命名为 `EDA1-你的学号-你的姓名-Assignment1`

(1 分)：1. if you prefer 手写版: 完成后请扫描或者拍照转成 pdf (注意，务必追求书写工整、字迹清晰，看不清楚的内容评阅人无责任追加辨认，将直接跳过)

2. if you prefer 电子版: 推荐使用 LaTeX 排版，生成 pdf（若你有读研计划，即刻开始熟悉 LaTeX 是明智之选）

注：提交的版本可以不带题目，但题目号一定写清楚，比如 I.13,II.5,III.1)！

Plagiarism policy: zero tolerance

作业接收邮箱：eda1_2024@163.com

I “新手村”（63 分）

Either 开放题 or 课件上都找得到的

1. 用你自己的一句话来告诉你的朋友 EDA 是什么（1 分，写了就有分）

2. 给 EDA 取一个‘不那么枯燥’的名字，你会给它取什么名字（1 分，写了都有分）

3. 下面哪个科技产品会用到 EDA 工具来设计，智能手机？机器人？还是电动汽车？（2 分）
   都需要

4. 在软件发展史里面，你认为哪一个发明或突破最能让人意识到“软件的力量”，为什么？（1 分）

5. 工业软件的定义和目标分别是什么？（2 分）

6. 现如今，工业软件在工业设计、管理、制造流程里面起主角作用还是配角作用？为什么？（2 分）

7. 当今主流工业软件巨头主要都是哪 3 个国家的企业？(1 分)
   美、德、法

8. EDA 是属于哪个领域的工业软件？这个领域的常见系统有哪些（列出三个）（2 分）

9. EDA 软件的核心功能是什么？请简述它在电子设计中的作用 (3 分)

10. 列出 EDA 人才需要的最重要的三大背景能力（1 分）

11. 逻辑电路图和电路网表图最重要的区别是哪一点？(2 分)
逻辑电路图更倾向于表达电路的逻辑功能，而电路网表图更贴近电路的物理实现和连接关系。两者的核心区别在于一个关注高层逻辑功能，另一个关注底层连接拓扑

12. 图作为一种数据结构在现实生活中有哪些应用场景？至少列出 5 种 (2 分)

13. 简述数字逻辑电路和模拟逻辑电路的功能（2 分）

14. 画出芯片设计全流程的步骤（1 分，先试试不翻阅课件自己写，期末必考）

BFS、DFS、Topo Sort Practice

15. (5 分）在图 1 中，假设在多节点选择时，广度优先搜索（BFS）和深度优先搜索（DFS）算法都会优先选择最左边的节点。从顶部的黄色节点开始，哪个算法在访问指定节点（紫色）之前会访问的节点数量少?

BFS:8 个
DFS:7 个
DFS 访问的节点数量少

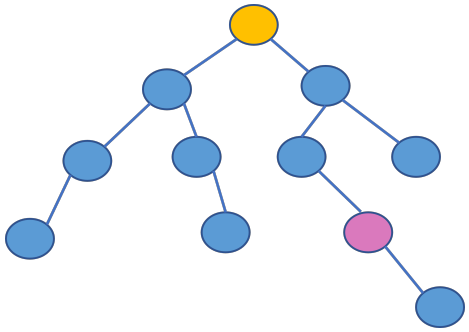

图 1: I.15

16. (10 分) 在图 2 中，如果 BFS 或 DFS 算法中的多个邻居节点之间存在决策，我们总是首先选择最接近字母表开头的字母。从节点 A 开始，使用广度优先搜索将按什么顺序访问节点？使用深度优先搜索将按什么顺序访问节点？

BFS: A、B、D、C、E、G、H、F（易错点：最后两位 H 和 F 的顺序。出错原因：没有 rigorouly 去 run BFS, 靠眼睛去 "数层数"））
DFS: A、B、C、E、H、F、G、D

17. (5 分) i) **以 Python 字典的形式（注意：需要完全符合 python 语法）**写出图 4 的邻接表（2 分）；在特定需求下，例如频繁地键初始化和更新，Python 里面哪个特有的数据结构比 dictionary 更优？为什么？（1 分）

- 注意：邻接表中字母一定要带双引号或单引号，没打的不符合 Python 语法

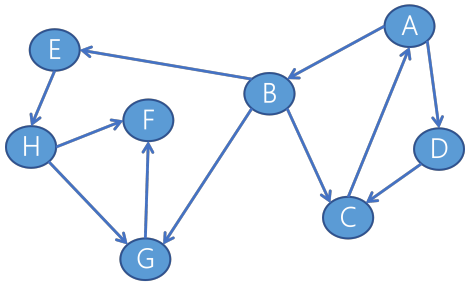

图 2: I.16

```
Adj = {'A':['B'],
       'B':['C','D'],
       'C':['E','F'],
       'D':['C','E','F'],
       'E':['F'],
       'F':['D']}
```

图 3: Adjacency List

defaultdict 比普通 dictionary 更优：自动处理缺失键

ii) 是否可以从图 4 中删除一条边，使其成为 DAG。说明具有此属性的每条边，并说明每个边的生成的 DAG 的拓扑排序顺序（2 分）

(F,D)
拓扑排序: A、B、D、C、E、F

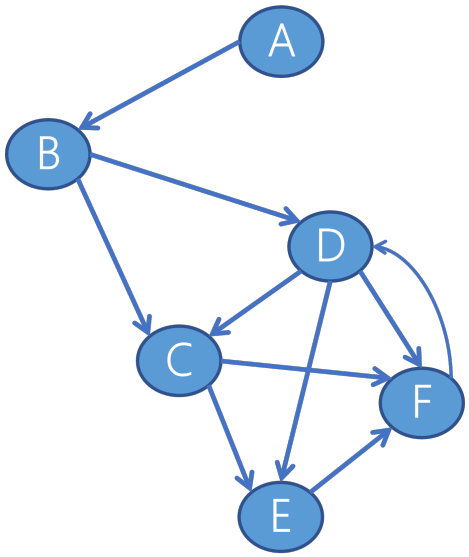

图 4: I.17

Big O Practice

18. (10 分) 游乐园里面你想玩一个射击气球游戏，射击板上有 n 个气球。假设你是神枪手，命中率 100%。游戏会出现有两种情况，请你看一下每种情况下完成游戏的时间复杂度是多少（以决定要不要玩，能不能赶上的回家的最后一班车）：

情况 1：你每射击 2 个气球，就会在板上出现一个新气球。比如，如果有 20 个气球，在您射击前 2 个气球后，板上还有 19 个。再射击 2 个气球后，板上还有 18 个。在板子空时，你玩游戏的时间复杂度和气球的个数的关系符合下面哪个 (2 分), 展示你的推导过程（3 分）：

(a) $O(n)$

(b) $O(1)$

(c) $O(\log n)$

(d) $O(n^2)$

如果棋盘上有一个气球，你总共射击 1 个气球；如果棋盘上有两个气球，你总共射击 3 个气球；如果棋盘上有三个气球，你总共射击 5 个气球。因此，依此类推，你总共射击 $2n-1$ 个气球。

情况 2: 在你射击完 n 个气球前，板上就会增加 $n-1$ 个新气球。在射击完这 $n-1$ 个新气球前，板上又会新增 $n-2$ 个新气球。在射击完这 $n-2$ 个气球后，板上有会有 $n-3$ 个新气球出现……以此类推，同样的模式继续，直到板上只新增 1 个新气球。那么，在板子清空时，你玩游戏的时间复杂度和气球个数的 big O 函数关系是什么？(2 分）展示你的推导过程（3 分）

$O(n^2)$
总共射击的气球数：$(n+(n-1)+(n-2)+(n-3)+\ldots\ldots+1) = n(n+1)/2$

Dijkstra、Bellman-Ford Practice

19. (10 分）在图 5 中：
i) 从 s 点开始寻找到其余点的最短距离，及最短距离对应的最短路径。
要求：展示你的解题过程（6 分）

最短距离:

- A: 8; 路径: S → A
- B: 9; 路径: S → A→ D → B
- C: 7; 路径: S → C
- D: 8; 路径: S → A → D
- E: 11; 路径: S → A → D → H → E
- F: ∞
- G: 12; 路径: S → A → D → H → G
- H: 10; 路径: S → A → D → H

ii) 假设 (G, C) 的权重变为 −11, 再次计算 s 到其余点的最短路径 (4 分), 同样要求展示过程

最短距离:

- A: 8
- B: 9
- C: 7
- D: 8
- E: 11
- F: ∞
- G: 12
- H: 10

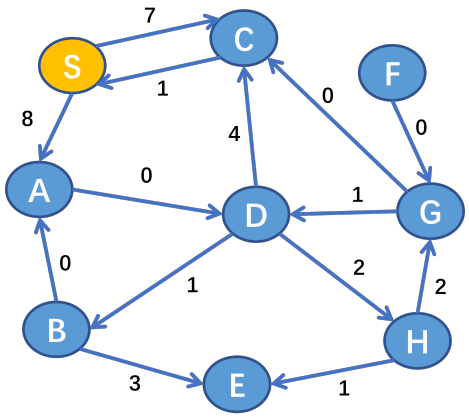

图 5: I.19

II ”练级地”（20 分）

1. （3 分）假设图中节点个数为 n, 分别列出在邻接表和邻接矩阵上执行以下 3 个操作的时间复杂度: 初始化、增加边、删除边

   邻接矩阵

   - 初始化：$O(n^2)$
   - 增加边：$O(1)$
   - 删除边：$O(1)$

   邻接表

   - 初始化：$O(n)$
   - 增加边：$O(1)$
   - 删除边：$O(n)$

2. （4 分）Modifying BFS for the s-t path problem：用 Python Code 实现一个基于 BFS 的算法，目标是寻找从节点 s 到 t 点的最短路径（注：input 为 general graph, 一切属性未知）。要求：在找到最短路径时返

回最短路径，否则返回 "None"（字符串 "None"）

```
def s-t-Search(graph, s, t):
```

3. (6 分) 在图的遍历和分析中，尤其是在深度优先搜索过程中，会遇到几种不同类型的边。了解这些边的类型对图的结构理解、循环检测和拓扑排序非常重要。下面介绍几种常见的边类型：

   - 树边（tree edge）
     - 定义：从一个节点访问另一个尚未被访问的节点时，形成的边叫树边
     - 作用：树边构成了 DFS 生成树，连接从根节点到每个节点的路径
   - 回边（back edge）
     - 定义：指从一个节点指向其祖先节点的边（在 DFS 树中，祖先节点是已经被访问并且在 DFS 栈中尚未退出的节点）
     - 作用：回边是有向图中存在环路的一个标志。若图中存在回边，则说明图包含环
   - 前向边（forward edge）
     - 定义：前向边是从一个节点指向其子孙节点的边，但这个子孙节点不是 DFS 树中直接连接的子节点（即不是树边）
     - 作用：前向边不是树的一部分，但它连接了树中的祖先节点和后代节点。它不影响环的判断，但有助于分析图的结构
   - 交叉边（cross edge）
     - 定义：交叉边是指连接两个不在同一 DFS 树路径上的节点的边。即，在 DFS 树中，两端节点没有祖先关系
     - 作用：交叉边只会出现在非连通图或有向图的不同强连通分量之间

   i. 根据以上定义，在图 7 中 E 点进行 DFS 的过程中，给图中所有边分类 (Note: E 出发的部分 "走" 完后，可再从 R 出发)

- Tree Edge(树边): (E,H), (H,G), (G,F),(E,T), (T,X), (X,Z),(R,U), (U,D)
- Back Edge(回边): (F,H), (Z,X)
- Forward Edge(前向边): (R,D)
- Cross Edge(交叉边): (D,E)

DFS 生成树: 在 DFS 遍历过程中，每当从一个节点访问到一个尚未访问的节点时，我们会记录一条边，这条边也就是上面定义的树边（Tree Edge），并把这个新访问的节点视为当前节点的子节点。这样，遍历的过程就逐渐形成了一棵树（或多棵树，如果图是非连通的），称为 DFS 生成树（一棵树的情况）或者 DFS 森林（多棵树的情况）。

ii.(3 分) 请画出图 5 中由 E 点为起始点的 DFS 生成树:

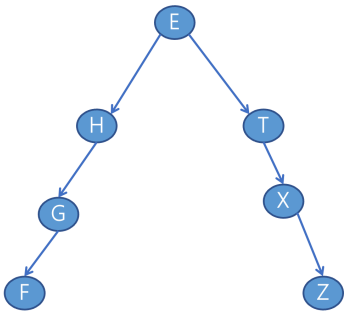

图 6: dfs 生成树

iii.(1 分) DFS 生成树和拓扑排序之间有一种直接、紧密的联系, 是什么?

遍历 DFS 生成树时，**节点的完成时间**逆序就是 DAG 的一个合法拓扑排序——In other words，利用递归的方式遍历 dfs 生成树，在回溯的时候记录节点（也就是节点完成时间），这个顺序的逆序就是合法的拓扑排序

(注：节点的完成时间是指当一个节点的所有子节点（或者所有邻接节点）都已经被访问并处理完成时，记录下该节点的"完成时间"。具体来说，完成时间表示该节点的搜索过程正式结束的时刻。)

iv.(2 分) 在**无向图**中，**连通分量（Connected Component）**是指无向图的一个**最大**连通子图，其中任意两个两个顶点都有路径相连，并且没有任何其他顶点可以添加到该子集使它依然连通。

在**有向图**中有**强连通分量（Strongly Connected Component）**和**弱连通分量（Weakly Connected Component）**，其中**强连通分量**是指有向图的一个**最大**子图，其中任意两个顶点之间都存在**双向路径**，即强连通分量中的从任意顶点 u 和 v, 都存在从 u 到 v 的有向路径，并且也存在 v 到 u 的有向路径。

根据上述定义，找出图 7 中的强连通分量

强连通分量 1：{H,G,F}, 强连通分量 2：{Z,X}
连通分量（Strongly Connected Component, SCC）不仅要求图中的每一对顶点都是可达的，而且还要求这个子图是"最大"的，意味着没有其他顶点可以被包含在这个分量中而仍然保持强连通性。

4. (3 分) 证明一个 DAG 必须包含至少一个**源点**和至少一个**汇点**

- 源点：在有向图中，源点是**没有入边**的顶点（即入度为 0 的顶点）

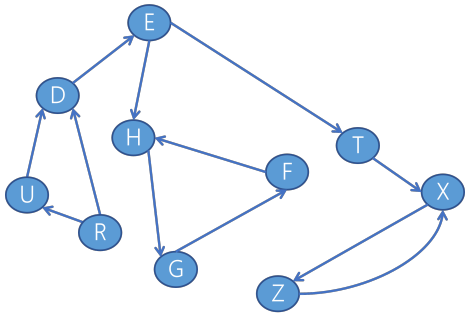

图 7: II.3

- 汇点：在有向图中，汇点是**没有出边**的顶点（即出度为 0 的顶点）

反证法：假设 DAG 图 G 中没有源点，即所有顶点的入度都大于 0

(a) 任取图中一个顶点 v_1，因其入度大于 0，所以存在一条边从某个顶点 v_2 指向 v_1

(b) 对于 v_2 同理，存在顶点某一点指向其自己

- 果这一点是 v_1, 那么已有环出现，得出矛盾，证明结束；
- 如果这一点不是 $v_1<$, 我们称其为 v_3

(c) 由于图中顶点数量有限，按此方式往前追溯，必然会重复使用某个已经出现过的顶点 v_k

(d) 这样就形成了一个有向环 $v_k \to \cdots \to v_2 \to v_1 \to \cdots \to v_k$, 这与 DAG 的定义矛盾

(e) 因此，原假设错误，DAG 必然至少存在一个源点

==========

类似反证法证明 DAG 图中至少有一个汇点

5. (4 分) 在 Leetcode 解第一题（两数之和），要求：至少写出两种不同时间复杂度的解题方式。对于每一种解题方式，
i), 在这里写清楚**解题思路**（包含如果需要用到的额外的数据结构）（2 分）

ii) 时间和空间复杂度如何计算得出？（2 分）

iii) 如果实在没有思路，再看解题，总结看完题解后的感悟和收获（注：此项为加分项，写的用心的多加分：鼓励你不用直接看题解来"骗自己学会了"）

III "小 Boss"(15 分)

1. 有人说："工业软件更贵，开发更难的原因是因为它使用了特殊的代码和资源"。你同意这个观点吗？谈谈你的看法，不少于 100 字。（3 分）

   核心：工业软件并不是需要多特殊的代码，它的核心难点在于，除了基本的编程能力，通常还需要开发者具备特定领域的专业知识（比如 EDA 需要微电子基础、解优化问题的能力），它是特定行业知识与经验"软件化"的体现——回顾一下课堂上放过的相关视频就能理解这一点

2. i)（1 分）解释算法复杂度的三种不同符号对应的意义（大 O 符号、Ω 符号和 Θ 符号）

   O 表示上界，描述最坏情况的复杂度；
   Ω 表示下界，描述最好情况的复杂度；
   Θ 表示进紧确界，描述平均情况的复杂度

   - 大 O 符号（big O notation）
     - 定义：大 O 符号用于描述算法的上界，即在最坏情况下，算法的运行时间或空间需求不会超过某个特定的函数。它提供了一个算法在输入规模趋近于无穷大时的增长率的上限

  - 数学表达：如果存在正的常数 C 和 n_0, , 使得对于所有 $n \geq n_0$, 都有 $T(n) \leq C \cdot f(n)$, 则我们说 $T(n)$ 是 $O(f(n))$

- 大 Ω 符号（big Omega notation）
  - 定义：大 Ω 符号用于描述算法的上界，即在最坏情况下，算法的运行时间或空间需求不会超过某个特定的函数。它提供了一个算法在输入规模趋近于无穷大时的增长率的上限
  - 数学表达：如果存在正的常数 C 和 n_0, 使得对于所有 $n \geq n_0$, 都有 $T(n) \geq C \cdot f(n)$, 则我们说 $T(n)$ 是 $\Omega(f(n))$

- 大Θ 符号（big Theta notation）
  - 定义：大 Θ 符号用于描述算法的紧确界，即算法的运行时间或空间需求在某个特定的函数的上下界之间。它表示算法的增长率是一个特定函数的精确描述。
  - 数学表达：如果存在正的常数 C_1、C_2 和 n_0, 使得对于所有 $n \geq n_0$, 都有 $C_1 \cdot f(n) \leq T(n) \leq C_2 \cdot f(n)$, 则我们说 $T(n)$ 是 $\Theta(f(n))$

ii) （6 分）对复杂度函数进行排序：使得复杂度等级较低得在前（增长速度较慢）在前，等级较高的（增长速度较快）在后。比如，$f_1 = n$ 排在 $f_2 = n^2$ 之前。若两个函数表示的算法复杂度等价，则用方括号表示它俩的等价关系，比如对 $f_1 = n$、$f_2 = n^2$ 和 $f_3 = n^2 + n$ 排序，表示为 $f_1, [f_2, f_3]$ 或者 $f_1, [f_3, f_2]$。

等价的严格定义：假设有两个函数 $f(n)$ 和 $g(n)$, 它们表示两个算法的时间或空间复杂度：

- 如果存在正的常数 c_1、c_2 和 n_0, 使得对于所有的 $n > n_0$，不等式 $c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ 城里，则可以说 $f(n)$ 和 $g(n)$ 是等价的，记作 $f(n) = \Theta(g(n))$

对下列不同函数组进行排序，并写出推导过程：

(a) $f_1 = (\log n)^n$, $f_2 = \log(n^{2024})$, $f_3 = \log(n^n)$, $f_4 = \log\log(2024n)$, $f_5 = (\log n)^{2024}$

(b) $f_1 = 2024^{n^2}$, $f_2 = 2024^n$, $f_3 = 2^n$, $f_4 = 2^{2024^n}$, $f_5 = 2024^{2^n}$

(a): f_4, f_2, f_5, f_3, f_1
$f_2 = \log(n^{2024}) = \Theta(\log(n))$
$f_3 = \log(n^n) = \Theta(n\log(n))$
$f_4 = \text{loglog}(2024n) = \Theta(\text{loglog}(n))$

(b): f_3, f_2, f_1, f_5, f_4
利用 $a^b = 2^{b\cdot\log_2(a)}$ 化简所有指数底数为 2

3. (5 分) 在某个科研合作网络中，有 n 个大型研究机构和 n^2 个小型实验室，它们之间通过双向通信链路直接连接。每一个实验室都依赖于某个大型研究机构提供的数据支持，这种支持可以是通过直接连接，也可以通过多个实验室递归地传递得到。网络具有如下特性：每一个实验室只能从唯一的研究机构获得支持，以避免数据的重复传递和冲突。现在，科研网络有机会在一个大型研究机构部署一个数据备份系统，以保证在该研究机构失效时，依旧能为所有依赖于它的实验室提供必要的数据支持。给定整个科研网络中所有通信链路的列表 L，设计一个时间复杂度为 $O(n^4)$ 的算法，来确定在哪个大型研究机构安装数据备份系统，以便在失效时能够为最多数量的实验室提供备份支持。

- 问题本质：找出哪个研究机构负责的实验室最多
- 分析：不清楚列表 L 的具体情况，限制条件
  - 将 n 个大型研究机构和 n^2 个小型实验室看做节点
  - 这 $n^2 + n$ 个节点之间的边由通信链路列表 L 描述
  - L 的具体情况有很多种，唯一的限制条件只有 "每一个实验室只能从唯一的研究机构支持，这种支持可以是实验室和研究机构的直接链接（即，二者之间存在 edge），或通过多个实验室递归传递得到（即，二者之间存在 path，path 的中间节点是不同的实验室）
  - 上面的限制条件说明：如果一个实验室 a 和某个研究机构 B 之间存在 edge 或 path，那么实验室 a 就不能跟除 B 以外的研究机构连通（否则不符合题设："每个实验室只能从唯一的研究机构获得数据支持"）

  - 所以图中，一个研究机构就对应一个 conected component(连通分量)，具有最大节点数量的连通分量里的研究机构，就是需要安装数据备份系统 (这样，在失效时最多数量的实验室能够得到备份支持)。算法的目标：找出最大的连通分量，return 其对应的研究机构

- 算法步骤：
  (a) 构建图，由 n^2+n 个节点构成，最差时间复杂度 $O(n^4)$
  (b) run BFS or DFS 去计算每个连通分量的节点数，确定最大的连通分量，时间复杂度跟图的 size 一样：$O(n^4)$
  (c) 在最大的连通分量里面查找到对应的研究机构，时间复杂度 $O(n)$
- 所以，以上算法的时间复杂度：$O(n^4)$

Last but not the least: 课程结束前，如果你在 leetcode 全站排名可达前 10000，总成绩中算法部分直接给满分