**系统可行性分析报告**

学号：2022337621031 姓名：姚仁杰

**1．引言**

**1.1 背景**

来源：浙江理工大学 ACM 校集训队

现状：新生缺少对ACM 国际大学生程序设计竞赛入门阶段的指导。

研发组织：个人

**1.2 项目的意义**

基于知识图谱的 ACM 国际大学生程序设计竞赛智能问答系统，核心价值在于把题库、算法知识、解题思路、样例代码等碎片化资源用结构化图谱组织起来，并结合自然语言问答接口，让新生能更快、更清晰地进入竞赛学习状态。

新生往往不熟悉“常见题型—算法—数据结构”的对应关系。系统能把题目与知识点关联起来，新生只需提问如“哪些题需要用二分？”即可得到题目列表和对应算法解释，降低学习门槛。

系统支持自然语言检索，如“简单的图论入门题”，会基于知识图谱返回合适的题目，并给出难度、来源、解法标签，帮助新生循序渐进地练习。

知识图谱能链接“题目 → 算法 → 经典模板代码”，新生在做题时可得到典型思路提示和参考实现，避免无从下手。

系统可记录学生做题情况，把“未通过的题目”映射到相关知识点，再推荐相似但稍简单的练习，实现定向补弱。

新生不仅能得到答案，还能看到“题目 → 涉及知识点 → 对应算法 → 示例代码”的推理链路，更容易理解为什么要用某个算法，提升知识掌握的系统性。

**2 国内外研究现状**

国外研究现状

代码/程序语义知识图谱（Programming Knowledge Graph, PKG）：研究者将代码元素（函数、方法、数据结构、调用关系、语义模式）抽象为图，支持代码搜索、语义检索与生成式任务。此类工作证明了把代码与元信息作为图结构可以提升检索与生成质量。代表性工作包括 GRAPH4CODE、近两年多篇将 PKG 用于代码检索与生成的预印本/会议论文。

KG 驱动的题目/问题检索与解释（KG-based question retrieval）：针对程序设计题或 StackOverflow/编程问答，有研究把问题和知识点构建成 KG，用于可解释的题目检索与相似题推荐（例如 KGXQR）。这类方法常把语义检索（BERT 等）与图结构证据结合来实现可解释性。

KG + GNN 的多跳推理/提示增强：为回答复杂问题（需要跨多个知识点或多步推理），研究用图神经网络从 KG 抽取相关子图/证据，再用于问答或作为 LLM 的提示（soft prompt / retrieval-augmented input）。最新工作提出“Question-Aware KG Prompting”等动态方法以提高相关性。

面向教育/训练的 KG＋ITS（智能辅导系统）：国外有不少把 KG 用于编程教学与练习推荐、个性化诊断的工作，把学生做题记录、知识点掌握度与题库知识图谱结合，提供个性化练习或错误诊断。

国内研究现状

工程化与教学场景导向明显：国内高校/机构在“竞赛训练平台”“智能题库”“自动评判/反馈”上把 KG 用作知识组织与推荐的底层结构，很多硕博论文和工程项目集中在题目本体构建、知识点标注、学生模型与训练路径推荐上。相关工作也在将 KG 与 ITS、自动批改、错题本结合。[MDPI+1](https://www.mdpi.com/2227-7102/15/9/1102?utm_source=chatgpt.com" \t "_blank)

题目检索与可解释推荐研究：国内研究者也关注如何用 KG 为给定自然语言查询（如“二分 + 离散化 相关题目”）返回可解释的题目与参考解法，这类工作常结合文本相似度与图证据（即混合检索）。[Dr. Liu Mingwei (刘名威)](https://mingwei-liu.github.io/assets/pdf/icsme2023-KG4QuesRecomm.pdf?utm_source=chatgpt.com" \t "_blank)

**3 系统概述**

**3.1 目标**

实现目标：

构建一个小规模、可演示的“基于知识图谱的 ICPC 智能问答 原型”，能从自然语言或关键词检索竞赛题目、返回关联的知识点/算法与参考代码片段。

功能要求：

题目检索

输入：“查找适合新手的图论题”

输出：列出相关题目（带难度、来源链接），并附上知识点/算法标签。

知识导航

输入：“哪些题需要二分查找？”

输出：知识图谱中与“二分查找”相关联的题目列表。

解题提示

输入：“ACM题：最短路相关题有哪些解法？”

输出：Dijkstra、Bellman-Ford 等方法的简要解释，并给出对应题目。

**3.2 新系统功能范围及划分说明**

1. 数据采集与预处理子系统（Data Ingest） (MVP)

爬取 / 导入题目与解法（ICPC Archive / Kattis / Codeforces / 本校题库或手工）清洗文本（去 HTML、统一编码）、抽取字段（题名、描述、示例、来源、难度、解法摘要、参考代码链接）。输出用于 KG 导入的 CSV/JSON 格式文件

核心脚本：scrape.py、extractor.py（正则/规则抽取初始标签：题目 tag, example）

2. 知识图谱构建子系统（KG Builder） (MVP)

定义 schema（实体类型、关系类型、属性），把 CSV/JSON 导入图数据库（推荐 Neo4j）并建立索引/同义词映射。保证可视化与导出（dump）能力

节点类型：(:Problem {id, title, difficulty, source, url})

节点类型：(:KnowledgePoint {name, synonyms, description})

节点类型：(:Algorithm {name, complexity, template\_url})

关系：(Problem)-[:考察]->(KnowledgePoint)

关系：(KnowledgePoint)-[:可用算法]->(Algorithm)

关系：(Problem)-[:参考实现]->(CodeSnippet)

3. 查询解析与映射子系统

将用户的自然语言或关键词输入解析成图谱查询意图（实体候选、过滤条件）。执行字符串匹配、同义词扩展；可选：调用语义匹配模型提升召回（Stretch）

实现策略（分阶段）：

阶段1：基于分词 +同义词表 +正则，做候选实体映射。

阶段2（Stretch）：用 sentence-transformers/BERT 做语义相似度检索 → 返回 top-k KP/算法候选。

4. KG 查询与检索子系统（Query Engine） (MVP)

把解析模块输出的实体/条件转成 Cypher，并在 Neo4j 上执行。支持两类查询：检索题目列表 & 返回题目详情（带证据路径）

5. 前端与可视化子系统（UI） (MVP)

提供用户交互界面：搜索框、过滤面板、题目列表、题目详情页、图谱可视化（路径展示）。支持点击节点展开/折叠、展示参考代码链接/解题思路文本。

6. 用户模型与推荐子系统（User Profile & Recommender） (Stretch)

记录用户做题情况（attempts, pass/fail, time）。根据掌握度推荐下一个题目（规则或简单协同过滤）

7. 后端服务层与集成子系统（Backend / Orchestration） (MVP)

提供 REST API 聚合解析、查询、用户、推荐等子系统。负责错误处理、日志、配置（Neo4j 连接）、简单认证。

**5．方案设计**

**设计方案：**

前端层（UI）：搜索界面、结果列表、题目详情。

服务层（REST API）：解析（NL2Query）、检索（Query Engine）、详情、用户/推荐模块、管理接口（数据导入）

数据层：Neo4j（知识图谱）、文件/对象存储（参考代码片段/题目原文）、轻量关系/文档 DB（用户记录，可用 SQLite / JSON）

辅助模块：爬取、导入脚本、同义词表与短语词典、语义向量服务

**系统功能模块：**

数据管道模块

知识图谱构建模块

KG Schema（关键节点与关系）

查询解析与实体映射模块

查询引擎（Query Engine / KG Query）

前端模块

用户/推荐模块

**实现算法：**

算法 A：关键词/实体映射（MVP 实现 — 基于词典+同义词）

目标：把用户 Query 映射为 KG 中的 KnowledgePoint / Algorithm 候选

算法 B：语义检索（Stretch — 向量化 + ANN）

目标：使用 sentence-transformers 把 Query 与节点 description/名称做向量相似度检索，提高召回

算法 C：KG Query 构造与执行（Cypher 生成）

目标：把解析出的实体与过滤条件转换为安全的 parameterized Cypher 查询

**6．项目进度计划（Software Project Schedule）**

需求分析 第1-2 天

系统概要/详细设计 第3-4天

数据准备与图谱构建 第5-7 天

后端功能开发 第8-10 天

前端开发与可视化 第 11-12 天

集成与测试 第 13-14 天