

第八届 CCF 开源创新大赛 · 项目申报书

一、基本信息

项	内容
作品名称	moonbit-pathfinding —— 面向 MoonBit 的生产级寻路 / 图算法与基础设施 (INFRA) 工具集
参赛赛道	MoonBit × CCF 开源创新大赛
作品仓库 (GitLink)	https://www.gitlink.org.cn/Taoyouce/moonbit-pathfinding
GitHub 镜像	https://github.com/Suquster/moonbit-pathfinding
包发布 (mooncakes.io)	https://mooncakes.io/docs/Suquster/moonbit-pathfinding (moon add Suquster/moonbit-pathfinding, 当前版本 v0.0.4)
开源许可	Apache-2.0
开发语言	MoonBit

二、项目概述

moonbit-pathfinding 是一个面向 MoonBit 语言生态的生产级寻路 / 图算法与基础设施 (INFRA) 工具集：以 38+ 寻路 / 图论算法为核心，同时涵盖 actor 并发模型、解析器组合子、正则引擎、序列化 / 编解码 (JSON / CBOR / Base64 / UTF-8 / CSV)、日志 / 度量 / 计时器、属性测试 (PBT) / 模糊测试 / 确定性仿真 (DST)、哈希与校验 (SHA-256 / HMAC / CRC-32 / FNV-1a)、日期时间 (civil 历法 / RFC 3339 / ISO 8601 时长)、CLI 参数解析、韧性原语 (指数退避 / 熔断器 / 令牌桶 / 滑动窗口限流)、文本 diff/patch (Myers) 与 SemVer 2.0.0、压缩 (DEFLATE RFC 1951 / gzip RFC 1952)、配置解析 (TOML / INI) 等二十余个 INFRA 方向子包，填补了 MoonBit 在最短寻路 / 图论算法与通用基础设施方向上的生态空白。设计哲学参考 Rust 社区成熟的 [pathfinding crate](#)——算法不强制绑定图数据结构，用户仅提供「后继函数」`fn(N) -> Array[(N, W)]` 即可调用；但所有算法实现均独立派生自原始论文，并在此基础上做了大量原创工程化贡献。

本库不满足于「能跑」，而是对标业界前沿路网加速技术 (CH / CCH / Hub Labeling 等)，在真实 OSM 路网数据上做可复现基准，并全程以可执行文档、运行时 合约谓词、三后端一致性差分测试为工程质量护栏。

项目定位——六层集合闭包链：项目不是单点作品，而是一条层层自举、真包含递进的集合链 (详见仓库 `docs/STRATEGY_CLOSURE.md`)：LO 寻路核心 C

L1 通用图算法 C L2 验证基础设施（可执行证明谓词 / PBT / DST / fuzz / bench）C L3 通用基础软件（19+ INFRA 子包，全 RFC/FIPS 向量对拍）C L4 语言工程工具链（mini 编译器 / 正则引擎 / LSP / 代码生成 / 构建工具 / 文档生成 / playground）C L5 「AI 原生软件工厂」方法论（acceptance 门禁 + 黄金向量对拍法 + paper-to-code 追溯 + 证据工件，即大赛核心理念“可复用、可演进、可持续的软件工程流程”的实体化）。各层互为消费方形成有向依赖闭环：L3 的 hash 为 L2 的基准做指纹、L3 的 diff 为 L4 的文档生成做回归、L2 的 PBT 反过来检验 L0-L4 全部层——39 个源码包、约 15 万行 MoonBit、2369 项测试即这条闭包链的可复现证据。在功能轴之外，项目还沿五条正交轴（正确性：单测 C 向量对拍 C PBT C 确定性仿真 C 三后端差分 C 证明谓词；性能：基准工件 C 回归 guard C 跨语言对标；平台：三后端 C 浏览器 C 包注册表；时间演进：semver C schema 破坏性检查；人机：可执行文档 C agent 指南）同步呈现完整闭包链，构成“闭包立方体”式的体系差异；向上的 L6 生态平台层（mooncakes 全量索引审计）与 L7 自治工厂层（自动门禁 → 自动定位回归）已有雏形证据，按 roadmap 持续推进。

三、核心功能与算法覆盖

已实现 38+ 个算法，覆盖从基础遍历到前沿路网加速的完整梯队：

1. 基础最短路与遍历

- BFS / DFS / 双向 BFS
- Dijkstra（含 indexed 快路径：CSR + 扁平数组 + 4 叉编码堆 + Dial 桶队列）
- A / IDA / 双向 Dijkstra
- Bellman-Ford（含负权 / 负环检测）、DAG 最短路
- Floyd-Warshall、Johnson 全源最短路
- Yen K 短路

2. 前沿路网加速梯队

- ALT（A*, Landmarks, Triangle inequality）
 - CH（Contraction Hierarchies，收缩层次）——查询仅数十微秒级
 - CCH（Customizable CH，可定制收缩层次）——度量无关预处理 + 快速换权（customization），含嵌套剖分（Nested Dissection）收缩序与消除树无堆查询（elimination-tree query）
 - Hub Labeling / HL、PHAST / RPHAST、many-to-many 批量查询
-

四、技术亮点与原创贡献

1. 可执行 proof predicates（运行时合约谓词）为 BFS / Dijkstra 等核心算法编写可执行的正确性合约，随每次测试运行时校验，并预留了向稳定 moon prove 形式化验证升级的清晰路径。

- 2. 可执行 Markdown 文档 README.mbt.md 通过 moon test
README.mbt.md 作为测试运行——文档里每一段 示例代码都会在每次 CI 上被编译并快照校验，杜绝「文档与代码漂移」。
- 3. 三后端一致性 CI 硬门禁 wasm-gc / native / js 三个后端结果做差分测试，作为持续集成的强制门禁，保证跨后端语义一致。
- 4. CCH 可定制收缩层次（本库进阶原创工程）
- 5. 度量无关预处理 + 快速换权：北京路网换权成本 1.89 s，相较 CH 全量重建 25.1 s 快 13.3×；厦门 111 ms vs 2.13 s 快 19.2×。
- 6. 消除树无堆查询 query_dist_tree (Bauer et al. / CCH 论文)：沿消除树 父链从 s、t 各自走到根，按 rank 升序松弛，无需优先队列。厦门 114 μs（比堆版 CCH 快 1.30×）、北京 728 μs (1.07×)，双城 48 组查询全量与 Dijkstra 对拍一致，并以 100 迭代差分 PBT + 换权 recustomize 守卫语义。
- 7. AI-agent 友好的 API 与用量记录 后继函数式 API + 图输入指南，让 AI 编程智能体可低摩擦调用；仓库内附 docs/AI_AGENT_USAGE.md 与完整开发记录。

五、性能验证（可复现基准）

1. 对标 Rust pathfinding crate（同机基准，中位口径，>1 = MoonBit 更快）

算法	规模 n	度数	Rust 中位 ms	MoonBit 中位 ms	加速比	正确性
BFS	1000	4	1.853	0.039	47.3×	
Dijkstra	1000	4	5.815	0.396	14.7×	
A*	1000	4	6.666	0.824	8.09×	
BFS	10000	4	17.616	0.232	75.8×	
Dijkstra	10000	4	55.927	1.230	45.5×	
Dijkstra	10000	16	131.671	3.940	33.4×	
A*	10000	16	162.709	5.192	31.3×	

- 12/12 用例全部快于 Rust，每用例加速比中位 31.7×，最低项仍有 7.9-8.2×。
- 全部用例两侧结果签名逐元素一致（黄金图交叉验证通过）。

2. 真实 OSM 路网基准（厦门 23,925 节点 / 北京 163,501 节点）

- CH 查询：厦门 39 μs、北京 132 μs（数十微秒级）。

- CCH 换权 vs CH 全量重建：北京 13.3×、厦门 19.2× 更快。
- CCH 消除树无堆查询：厦门 114 μs、北京 728 μs，与 Dijkstra 全量对拍一致。

诚实口径：嵌套剖分（ND）序当前的 BFS 层分隔符实现在大规模路网上填充过大（厦门骨架边 402 万 vs min-degree 20 万），北京规模构建不可行，已在基准报告中明确标注为已知问题，生产推荐 min-degree 序；ND 序保留为实验路径，小图正确性由 PBT 持续守卫。改进方向：真正的小分隔符（inertial flow / flow-based refinement）+ 构建端桶合并。

六、工程质量

- 代码规模：src/ 下 40+ 包，directed 子模块 9,100+ 行。
 - 测试：全量 2,309 个测试通过；moon check 零告警；moon fmt / moon info 纳入提交前检查。
 - CI：GitHub Actions 三后端（wasm-gc / native / js）差分门禁。
 - 可复现：所有基准均附命令、种子、机器环境，结果归档于 benches/ results/。
-

七、用户体验

- 低摩擦安装：moon add Suquster/moonbit-pathfinding 一行安装（已发布 mooncakes.io v0.0.4）。
 - 交互式 Playground：网格寻路可视化，由 src/ 同一算法库编译为 wasm-gc 驱动。
 - 可执行文档：README 示例即测试，复制即可运行。
 - AI-agent 友好：后继函数式 API + docs/AI_AGENT_USAGE.md 用量指南。
-

八、对标与借鉴说明

本库 API 哲学借鉴 Rust pathfinding crate (v4.15.0, MIT OR Apache-2.0) 的「后继函数」极简设计、泛型节点类型 (N : Eq + Hash)、Option[(Array[N], W)] 返回风格；但所有算法实现独立派生自原始论文。相较原参考，本库的原创增量为：可执行 proof predicates、可执行 README、三后端一致性门禁、CH/CCH/HL 前沿加速 梯队与真实 OSM 路网基准。

针对 MoonBit 生态内已有同类包（寻路：moonpathfinding/petgraph 移植等；INFRA：sha256/flate/toml/piediff/circuit_breaker 等），本项目已完成基于 mooncakes.io 全量索引（1491 包）的逐领域筛查并出具对照与取舍说明（docs/ECOSYSTEM_COMPARISON.md）：所有实现均独立派生自原始论文 / RFC / FIPS 规范，对拍对象为权威参考实现（CPython、Go stdlib、python-lz4

等) 的输出向量而非任何生态包源码；INFRA 子包首要角色是本项目工程闭环的支撑层（基准指纹、发布校验、文档 diff、CI 韧性均有主线内真实消费方），其次才是可独立复用的生态贡献。

本申报书对应作品版本 v0.0.4，仓库与 mooncakes 包均已公开同步。