Project GUNDAM 开发指导

付文智

总述

- 首要目的：支撑本组研究、并以此为目的完善GUNDAM库
  - 熟悉GUNDAM库的使用，并以此支撑较为可靠的工作
  - 有的放矢地对GUNDAM库进行拓展、不脱离实际生产
  - 积累实践经验&数据、以支撑工程上的trade-off

- 对以后想找C++开发岗的同学：
  - 需要抓紧时间 & 机会锻炼自己
    - 多交流&合作，共同成长
      - 同学之间就别卷了，尤其是工程方面
    - 用于研究的代码做到尽可能扎实、切忌将几个月的编程经验用上一两年
  - 参与本项目可（快速）提升C++开发水平

开发路径

1. 添加可有可无的测试
2. 基于GUNDAM开发/完善非核心应用
   - 可能遇到的问题：
     a. 被编译错误困扰，尤其由模板类引起的复杂编译错误
     b. 对库不熟悉、重复实现已有功能
3. 添加/优化 GUNDAM库非核心功能
   - 可能遇到的问题：
     a. 封装无力，接口抽象程度不够、难以在其他程序中调用
     b. 更贴近底层、会引起更多复杂编译错误
     c. 难以使用库中依赖callback的基本功能
4. 对已有功能进行完整测试
   - “优秀的开发才能去作测试”
5. 优化/开发/完善 核心应用
6. 优化/开发/修改 GUNDAM核心功能

添加可有可无的测试

- 目标:
  - 通过GitHub上的文档、上手GUNAM，熟悉库中已有方法的调用方式
  - 大概了解当前库中有哪些功能、避免重复建设
  - 不指望此处能测出代码库中的错误、但也不必担心此举会引入错误

- 具体实践:
  1. 向gundam/test/test_data/test_pattern_set.h中添加更多的数据图
  2. 依照已有代码、将这些数据图应用于gundam/test/中更多测试
  3. 保证新添加的测试能够编译 & 测试通过
  4. 补全doc/中相关方法的文档

基于GUNDAM开发/完善非核心应用

- 目标：
  - 运用库中已有功能、实现新的应用
- 可能遇到的问题：
  a. 被编译错误困扰，尤其由模板类引起的复杂编译错误
  b. 对库不熟悉、重复实现已有功能
- 没什么很好的解决措施、只能遇到问题多沟通

- 具体实践：

  完善grape-gundam中：

  1. gar_to_gar_set.h
  2. gar_set_to_gar.h
  3. graph_statistic_collection.h
  4. gar_info_collection.

  或添加其他有用工具

添加/优化 GUNDAM库非核心功能

- GUNDAM的功能需要在实际应用中扩充
- 由于开发周期限制、许多功能未能提供高效实现

- 目标：进一步熟悉GUNDAM，同时完善&优化GUNDAM

- 可能遇到的问题：
  a. 封装无力，接口抽象程度不够、难以在其他程序中调用
  b. 更贴近底层、会引起更多复杂编译错误
  c. 难以使用库中依赖callback函数的基本功能

添加/优化 GUNDAM库非核心功能

- 具体实践
  - 对所需功能进行提炼，对接口进行抽象&标准化
  - 实现&测试自己、或同学提出的需求

添加/优化 GUNDAM库非核

```cpp
/* ######################################## *
 * ##  wenzhi: todo:                     ## *
 * ##    decompose the input graph into  ## *
 * ##    set of connected components     ## *
 * ######################################## */
template <bool bidirectional = true,
          typename GraphType>
std::vector<GraphType>  // OK to return vector after C11
     ConnectedComponent(GraphType& graph) {
  assert(false);
  std::vector<GraphType> connected_components;
  assert(!connected_components.empty());
  /* ################################################################## *
   * ##  wenzhi: hint                                                ## *
   * ##    Call the method ConnectedComponent(graph, vertex_handle)  ## *
   * ##    above here                                                ## *
   * ################################################################## */
  return  connected_components;
}
```

总结需求，抽象&规范接口 by 文智

```cpp
/* ######################################## *
 * ##    decompose the input graph into  ## *
 * ##    set of connected components     ## *
 * ######################################## */
template <bool bidirectional = true,
          typename GraphType>
std::vector<GraphType>  // OK to return vector after C11
     ConnectedComponent(const GraphType& graph) {

  using VertexHandleType = typename VertexHandle<GraphType>::type;
  using VertexIDType = typename GraphType::VertexType::IDType;
  std::vector<GraphType> connected_components;
  connected_components.clear();

  /* ######################################## *
   * ##  wenzhi: optimize me                 ## *
   * ##    to support const GraphType in     ## *
   * ##    ConnectedComponent(graph, vertex_handle  ## *
   * ######################################## */
  GraphType temp_graph(graph);
  GraphType k_hop_subgraph;

  std::unordered_set<VertexIDType> processed_nodes;

  for (auto vertex_it = temp_graph.VertexBegin();
                !vertex_it.IsDone();
                vertex_it++) {
    if (processed_nodes.find(vertex_it->id())
                  != processed_nodes.end()) {
      continue;
    }

    k_hop_subgraph = GUNDAM::ConnectedComponent(temp_graph, vertex_it);
    connected_components.emplace_back(k_hop_subgraph);

    for (auto kh_vertex_it = k_hop_subgraph.VertexBegin();
                  !kh_vertex_it.IsDone();
                  kh_vertex_it++) {
      processed_nodes.insert(kh_vertex_it->id());
    }
  }


  assert(!connected_components.empty());
  return  connected_components;
}
```

开发 & 测试 by 沐阳

添加/优化 GUNDAM库非核心功能

- 具体实践
  - 对所需功能进行提炼，对接口进行抽象&标准化
  - 实现&测试自己、或同学提出的需求
  - 对已有低效实现进行优化

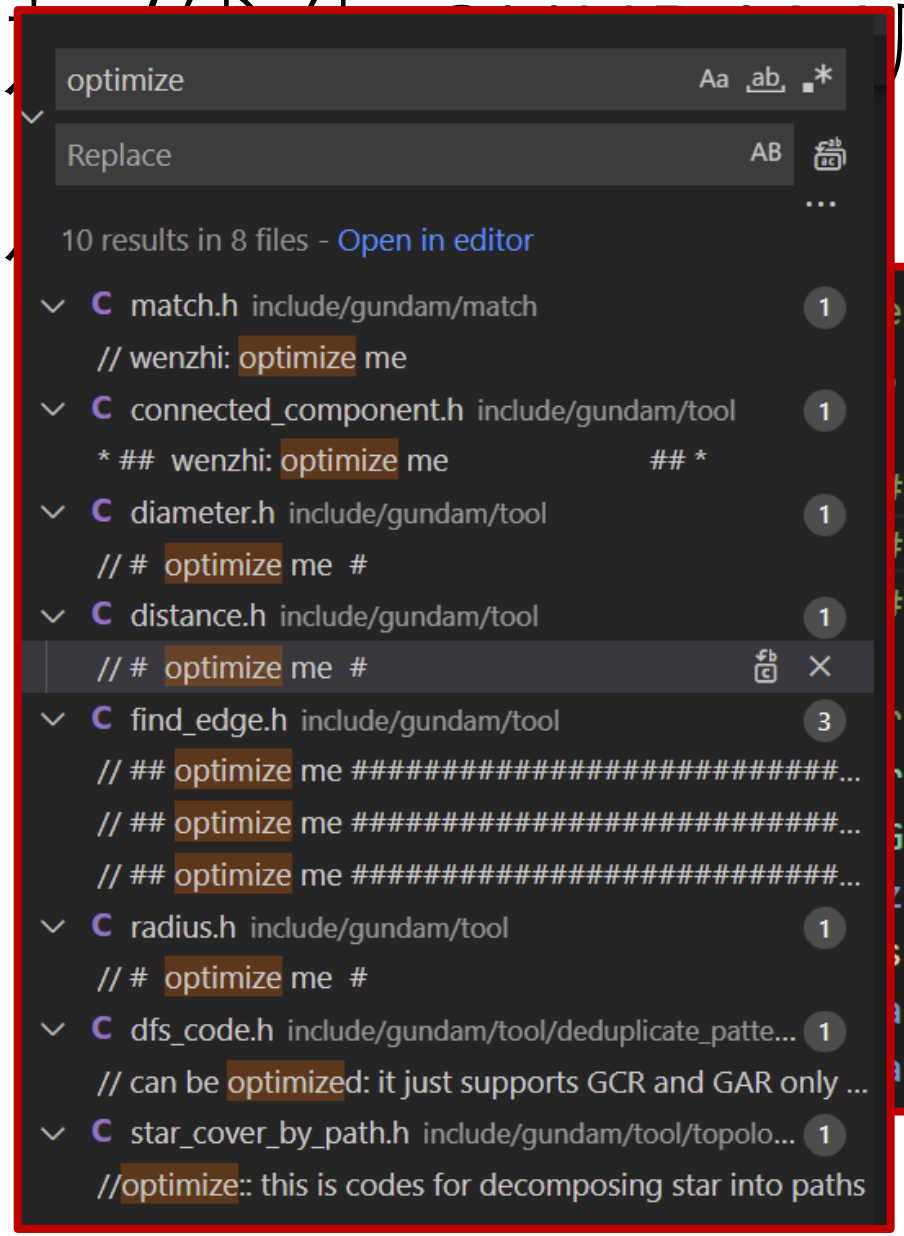
添加/优化 GUNDAM库非核心功能

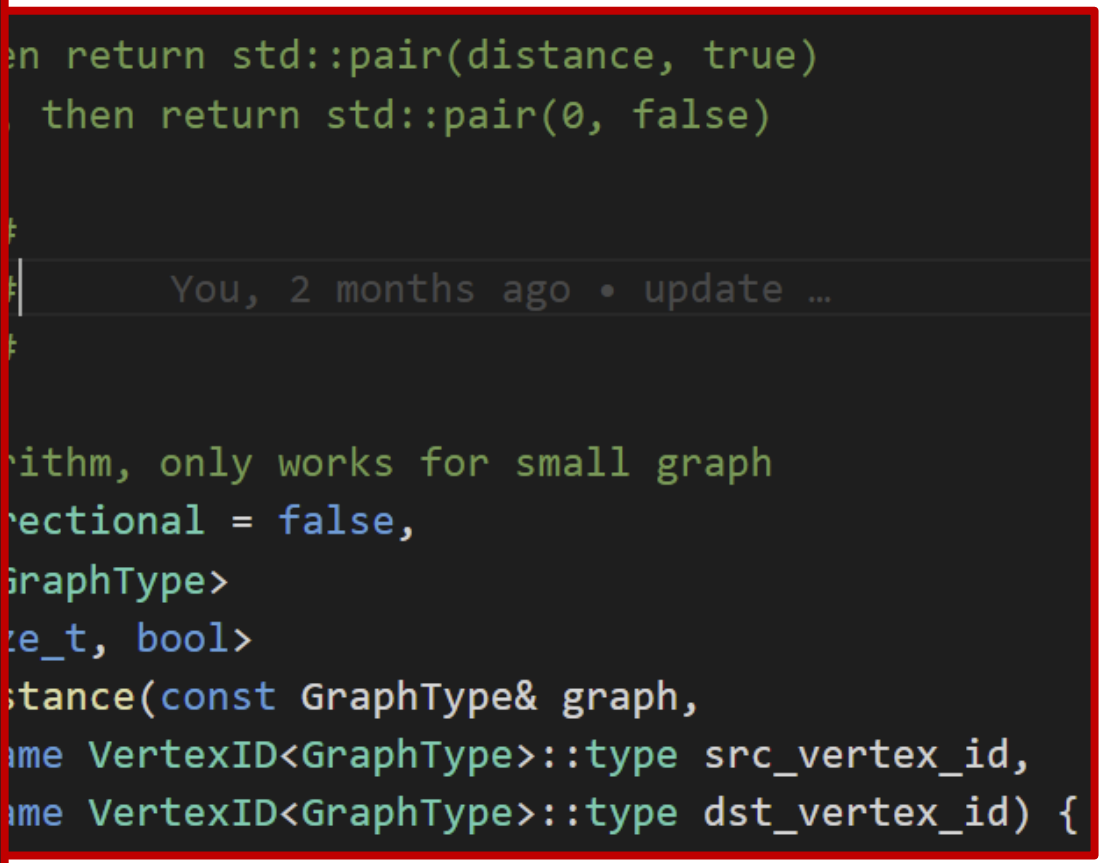
- 具体实践
  - 对所需功能进
  - 实现&测试自
  - 对已有低效实

```
// if connected, then return std::pair(distance, true)
// if not connected, then return std::pair(0, false)
//
// ##################
// #  optimize me   #      You, 2 months ago • update …
// ##################
//
// using floyd algorithm, only works for small graph
template <bool bidirectional = false,
          typename GraphType>
inline std::pair<size_t, bool>
                 Distance(const GraphType& graph,
                typename VertexID<GraphType>::type src_vertex_id,
                typename VertexID<GraphType>::type dst_vertex_id) {
```

添加/优化 库非核心功能

- 具





对已有功能进行完整测试

- 目标：对已有方法添加完整测试，保证在各种边缘情况下也能正常执行

优化/开发/完善 核心应用

- gar_discover，gar_supp等较为核心的应用中缺乏部分功能
  - 如gar_discover中：
    - 不支持分辨限制rhs、lhs的literal类型
    - 不支持输出日志
    - 不支持配置confidence bound等
  - gar_supp中：
    - support & confidence 文件路径命名混乱
- gar_imply 的yaml接口较为复杂、难用
- 长期无人维护的应用，如 dpiso/distribute_match 无法在当前GUNDAM版本中编译通过
- 随着研究的进展需要更多功能，如key等

优化/开发/修改 GUNDAM核心功能

- 实现GPU-Graph以支持GPU计算
- 实现有连续ID (or inner id) 的单增图、以提升匹配效率
- 静态attribute
- ……

总结

1. 添加可有可无的测试

   向gundam/test/test_data/中添加更多数据图、并将其添加到已有测试中
2. 基于GUNDAM开发非核心应用

   grape-gundam，tools中：
   1. gar_to_gar_set.h
   2. gar_set_to_gar.h
   3. graph_statistic_collection.h
   4. gar_info_collection.
3. 向GUNDAM库添加非核心功能

   （提炼研究过程中实现的）
4. 对已有功能进行完整的测试
5. 优化/添加核心应用
   1. 重整gar_supp中confidence
   2. 向gar_discover中添加日志
   3. 在当前GUNDAM版中编译通过grape-gundam上的dpiso/distribute_match
6. 优化/添加GUNDAM核心功能
   1. 优化match
   2. 实现更多baseline的match算法

Thanks!