# 图计算实验部分

# 实验一 YiTu安装与配置实验

【实验目的】

通过本实验，学习如何在Linux环境下配置图计算执行引擎YiTu\_XGraph的安装与使用，为后续的图数据处理和分析工作做好准备。

【实验内容】

1. 安装Linux系统。
2. 安装docker (可选)
3. 安装YiTu图计算编程框架

【实验环境】

1. 操作系统：Linux（Ubuntu）
2. 软件环境：conda, python
3. 硬件要求：至少1台计算机或虚拟机，建议配置至少4GB内存和100GB的硬盘空间。
4. 网络连接：互联网连接，用于下载所需的软件和文档

【实验步骤】

1. 用浏览器访问https://github.com/CGCL-codes/YiTu，下载.zip源代码文件。
2. 编译安装YiTu：

通过cmake安装：

rm -rf ~/temp

version=3.18

build=0

mkdir ~/temp

cd ~/temp

wget https://cmake.org/files/v$version/cmake-$version.$build-Linux-x86\_64.sh

sudo mkdir /opt/cmake

sudo sh cmake-$version.$build-Linux-x86\_64.sh --prefix=/opt/cmake --skip-license

sudo ln -s /opt/cmake/bin/cmake /usr/local/bin/cmake

cd ~

rm -rf ~/temp

\*\*(仅安装图计算部分)

定位到源码/examples/YiTu\_GNN/NDP目录下：

删除原有pybind11目录，直接下载pybind11源码

git clone <https://github.com/pybind/pybind11.git>

cmake .

（如果报错，定位到cmake报错代码处，将py修改成pybind11即可）

make

cd nondp

make

# 图计算实验部分

## 实验二 DFS实验

【实验目的】

本实验旨在通过编写和执行基于YiTu\_XGraph的DFS程序，帮助学生深入理解图计算系统的工作原理，并学会使用YiTu\_XGraph进行大规模图数据分析和处理。通过此实验，学生将能够掌握图计算的基本概念、编写简单的图算法程序以及运行它们在YiTu\_XGraph系统中。

【实验内容】

1. 基于YiTu\_XGraph实现DFS算法，掌握DFS算法和函数式编程思想。
2. 使用命令行执行DFS程序。
3. 查看程序执行结果。

【实验环境】

1. 操作系统：Linux（Ubuntu）
2. 软件环境：conda, python
3. 硬件要求：至少1台计算机或虚拟机，建议配置至少4GB内存和100GB的硬盘空间用于安装Hadoop。
4. 网络连接：互联网连接，用于下载所需的软件和文档。

【实验步骤】

1. 接下来，将以YiTu\_XGraph中的BFS算法为例，讲解如何在YiTu\_XGraph实现BFS算法并运行。

以BFS算法为例的实现细节：执行python demo.py --YiTu\_GNN 0 --method bfs –input Graph.bcsr --source 0

后调用<root>/YiTu/examples/YiTu\_GNN/NDP/ndp/bfs\_sig\_async.cu代码，需要注意，输入只支持.bcsr和.el格式的数据集

迭代方式的BFS算法的伪代码：

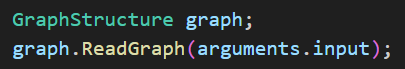
|  |  |
| --- | --- |
| **算法：**BFS() | |
| **输入：**图，指定源顶点 | |
| **输出：**图所有顶点与的深度向量 | |
|  | 初始化活跃顶点队列和下一轮的活跃顶点队列; |
|  | 初始化深度向量; |
|  | 将顶点加入; |
|  | **while**(不为空) **do** |
|  | Vertex .front(); |
|  | 出队; |
|  | **while**(&& +1) **do** |
|  | +1;//遍历到未被访问的邻居时，使邻居的深度加1 |
|  | 顶点加入队列; |
|  | **end while** |
|  | 交换队列和; |
|  | **end while** |
|  | **return** ; |

1. 初始化：初始化包括分析指令、初始化图结构、初始化图状态三个阶段。

分析指令时创建ArgumentParser对象分析Linux指令的输入参数；



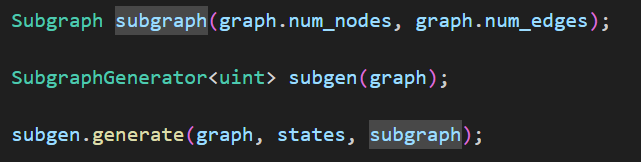
初始化图结构创建GraphStructrue对象，并根据输入的图获取图的数据，如顶点数、边数以及分布情况。并根据输入图计算各顶点的度数，以CSR格式存储。



初始化图状态根据图顶点数和算法需求创建GraphStates对象，并初始化活跃顶点

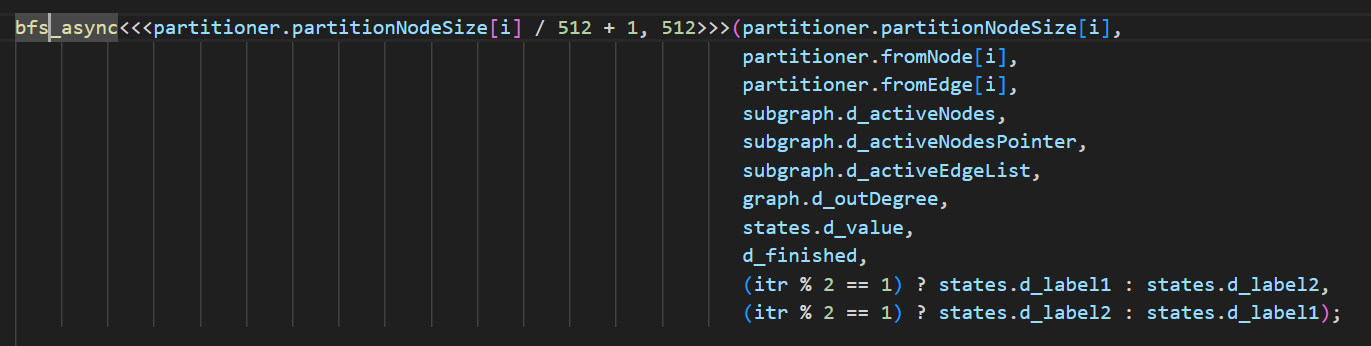


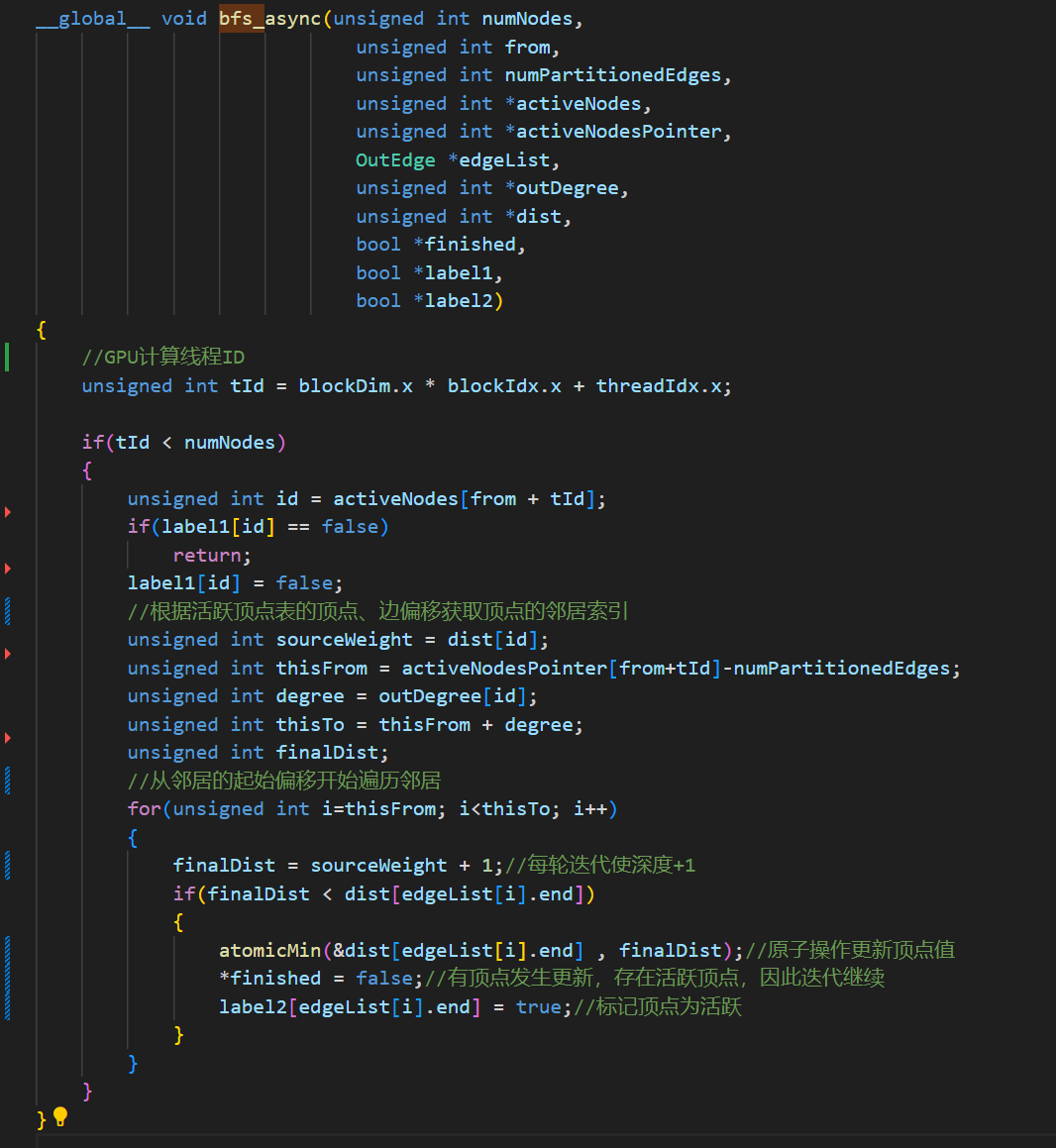
1. 子图划分：创建Sugraph、SubgraphGenerator、Partitioner等对象





1. 迭代：根据当前迭代是否存在活跃顶点进行迭代。每轮迭代执行CUDA核函数。对应核函数代码在<root>/YiTu/examples/YiTu\_GNN/NDP/shared/gpu\_kernels.cu：





运行指令和输出结果：

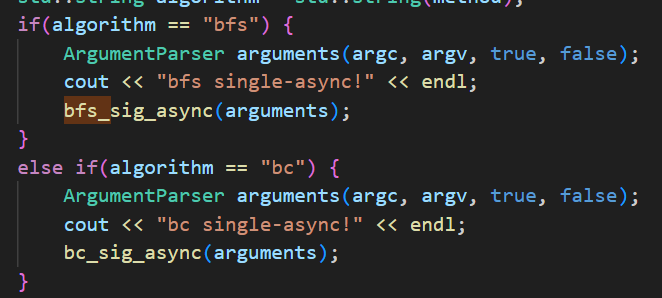
编译完成后执行Python demo.py –YiTu\_GNN 0 –method <app-name> --input <输入数据集> --source <源顶点>



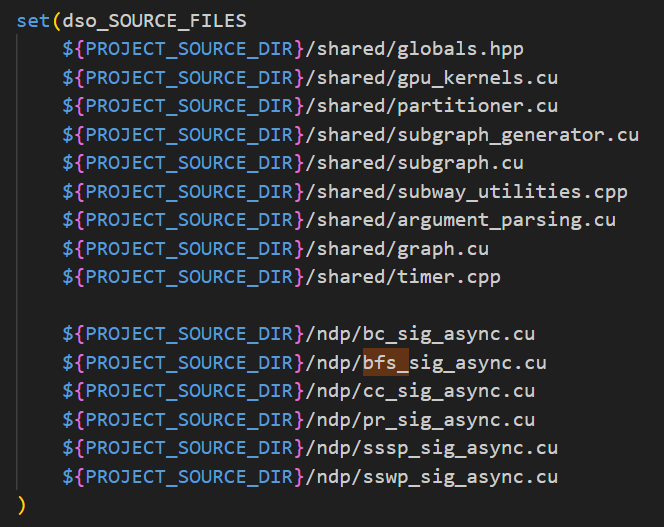
运行bfs的输出结果

接下来，请同学们将以YiTu\_XGraph中的BFS算法为例，在YiTu\_XGraph实现DFS，并成功正确地跑出结果。

（注意事项）



新添加算法后需要在<root>/YiTu/examples/YiTu\_GNN/NDP/YiTu\_GP.cpp中算法的if else中添加新增加的算法，使其定位到新添加的算法中。并在<root>/YiTu/examples/YiTu\_GNN/NDP/CMakeLists.txt中的set中添加新增的算法文件，并重新进行编译



# 图计算实验部分

## 实验三 SCC实验

【实验目的】

本实验旨在通过编写和执行基于YiTu\_XGraph的SCC程序，帮助学生深入理解图计算系统的工作原理，并学会使用YiTu\_XGraph进行大规模图数据分析和处理。通过此实验，学生将能够掌握图计算的基本概念、编写简单的图算法程序以及运行它们在YiTu\_XGraph系统中。

【实验内容】

1. 基于YiTu\_XGraph实现SCC算法，掌握SCC算法和函数式编程思想。
2. 使用命令行执行SCC程序。
3. 查看程序执行结果。

【实验环境】

1. 操作系统：Linux（Ubuntu）
2. 软件环境：conda, python
3. 硬件要求：至少1台计算机或虚拟机，建议配置至少4GB内存和100GB的硬盘空间用于安装Hadoop。
4. 网络连接：互联网连接，用于下载所需的软件和文档。

【实验步骤】

1. 接下来，将以YiTu\_XGraph中的CC算法为例，讲解如何在YiTu\_XGraph实现CC算法并运行。

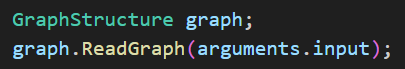
2.以CC算法为例的实现细节：执行python demo.py --YiTu\_GNN 0 --method CC–input Graph.bcsr --source 0。与BFS算法的主要区别在于核函数的实现方式的不同

1）初始化：初始化包括分析指令、初始化图结构、初始化图状态三个阶段。

分析指令时创建ArgumentParser对象分析Linux指令的输入参数；



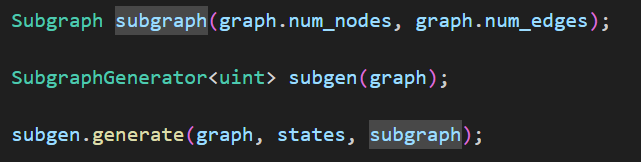
初始化图结构创建GraphStructrue对象，并根据输入的图获取图的数据，如顶点数、边数以及分布情况。并根据输入图计算各顶点的度数，以CSR格式存储。



初始化图状态根据图顶点数和算法需求创建GraphStates对象，并初始化活跃顶点

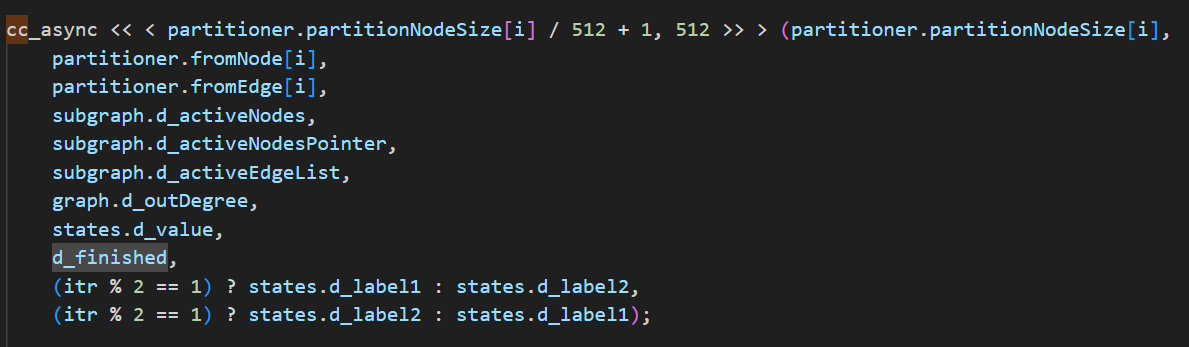


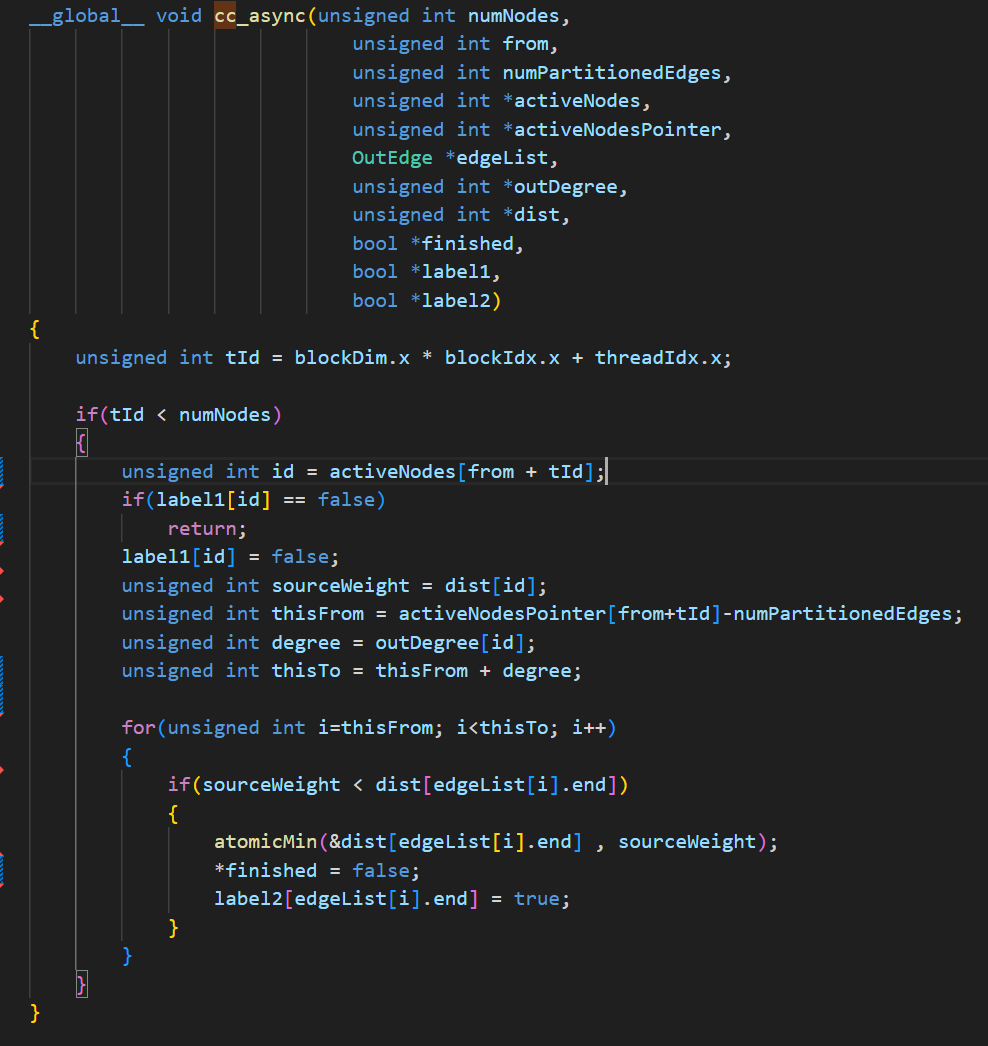
1. 子图划分：创建Sugraph、SubgraphGenerator、Partitioner等对象





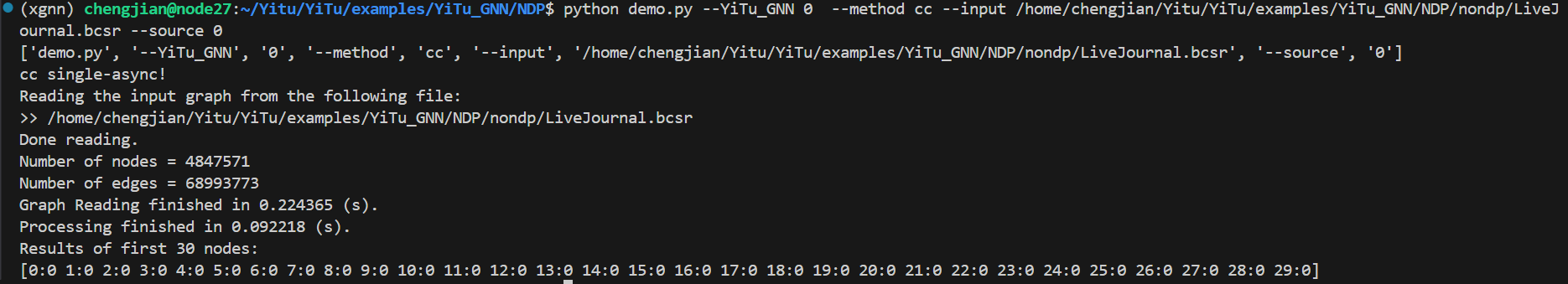
1. 迭代：根据当前迭代是否存在活跃顶点进行迭代。每轮迭代执行CUDA核函数。CC执行的核函数如下，对应代码在<root>/YiTu/examples/YiTu\_GNN/NDP/shared/gpu\_kernels.cu：





1. 运行指令和输出结果：

编译完成后执行Python demo.py –YiTu\_GNN 0 –method <app-name> --input <输入数据集> --source <源顶点>



运行CC算法的输出结果

接下来，请同学们将以YiTu\_XGraph中的CC算法为例，在YiTu\_XGraph实现SCC，并成功正确地跑出结果。