并行程序设计大作业-求出曼德博集合

目的

提升对 OpenMp 和 MPI 优化的认识
理解串行与并行带来的速度差异
理解程序负载均衡的意义

问题描述

曼德博集合(或曼德博复数集合)是一种在复平面上组成分形的点的集合，以数学家本华·曼德博的名字命名。

曼德博集合可以使用复数多项式定义:

$$f_c(z) = z^2 + c$$

其中$c = a + bi$ 是一个复数。从 $z = 0$ 开始对 $f_c(z)$ 进行迭代:

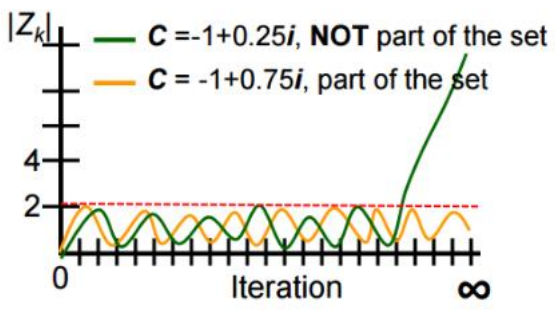
$$\boldsymbol{z_{n+1} = z_n^2 + c, n = 0, 1, 2, \dots\dots}$$
$$\boldsymbol{z_0 = 0}$$
$$\boldsymbol{z_1 = z_0^2 + c = c}$$
$$\boldsymbol{z_2 = z_1^2 + c = c^2 + c}$$

每次迭代的值依序可以按照如下序列表示

$$\left(\boldsymbol{0, f_c(0), f_c(f_c(0)), f_c\left(f_c(f_c(0))\right), \dots\dots}\right) = \{\boldsymbol{z_0, z_1, z_2, \dots}\}$$

不同的复数 c 可以使序列的绝对值$|\boldsymbol{z_k}|$逐渐发散到无限大，也可能一直收敛在有限的区域内。曼德博集合M就是使序列不延伸至无限大(用高数的话说，收敛)的所有复数c的集合。

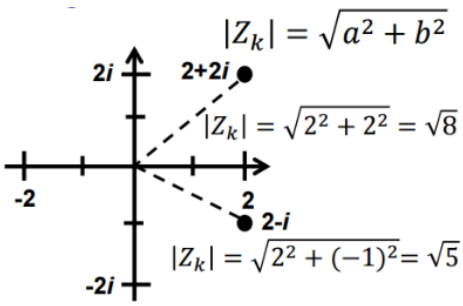
有个已证明的定理，当有个复数 c 使得$|\boldsymbol{z_n}| \leq \boldsymbol{2}, (\boldsymbol{n = 0, 1, 2, \dots\dots, k})$时，这个 c 就属于曼德博集合。**理论上**这个 $\boldsymbol{k}$ 是趋近于无穷大，在此题我们设 $\boldsymbol{k}$ 为 **10000**。

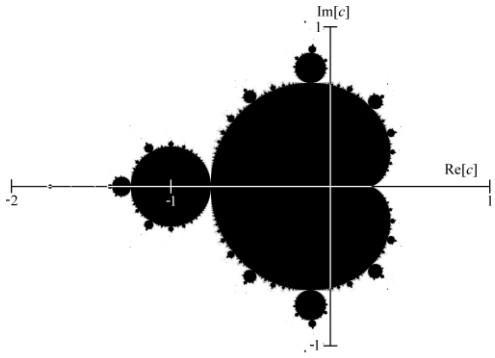


我们将利用这个理论找出特定范围内的所有曼德博集合，并使用 png.h 库把集合用描点的形式画出来。伪代码如下

```
For Each c in Complex
 repeats = 0
 z = 0
 Do
  z = z^2 + c
  repeats = repeats + 1
 Loop until abs(z) > EscapeRadius or repeats > MaxRepeats
'根据定理，EscapeRadius 可设置为 2。

 If repeats > MaxRepeats Then
  Draw c,ColorYouWant    '如果迭代次数超过 MaxRepeats=10000，就将 c 认定为属于曼德博集合，并设置为其他颜色。
 Else
  Draw c,color(z,c,repeats)                    'colo 函数用来决定色。
 End If
Next
```





其中实部 a 对应平面上的横轴，虚部 b 对应平面上的纵轴。

输入输出格式

使用命令行编译，需要加上 fopenmp, -lpng 等选项

```
mpicc ${filename.c} -O3 -pthread -lm -lpng -fopenmp -std=gnu99 -o filename
```

使用 mpirun 运行程序

```
mpirun -n $procs ./${filename} ${thread_per_proc} ${real_lower}
${real_upper} ${imag_lower} ${imag_upper} ${w} ${h} ${output_path}
```

每个参数的含义如下

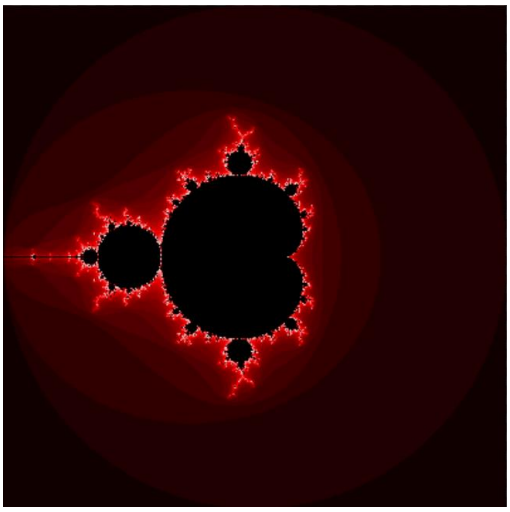
参数	类型	范围	解释
$proc	int	[1,24]	进程数，当只有 OpenMP 优化时默认为 1
$filename	string		要执行的二进制文件名
$thread_per_proc	int	[1,12]	每个进程的线程数，当只有 MPI 优化时默认为 1
$real_lower	double	[-10,10]	图像上实轴的最小值
$real_upper	double	[-10,10]	图像上实轴的最大值
$imag_lower	double	[-10,10]	图像上虚轴的最小值
$imag_upper	double	[-10,10]	图像上虚轴的最大值
$w	int	[1,16000]	输出的图像实轴上的点数量
$h	int	[1,16000]	输出的图像虚轴上的点数量
$output_path	string		输出图形的路径，方便起见最好设置为 "${filename}.png"

样例

样例 A

```
$ mpicc homework6.c -O3 -pthread -lm -lpng -fopenmp -std=gnu99 -o homework
$ mpirun -n ${proc} ./homework ${thd_per_proc} -2 2 -2 2 400 400 ./homework.png
```

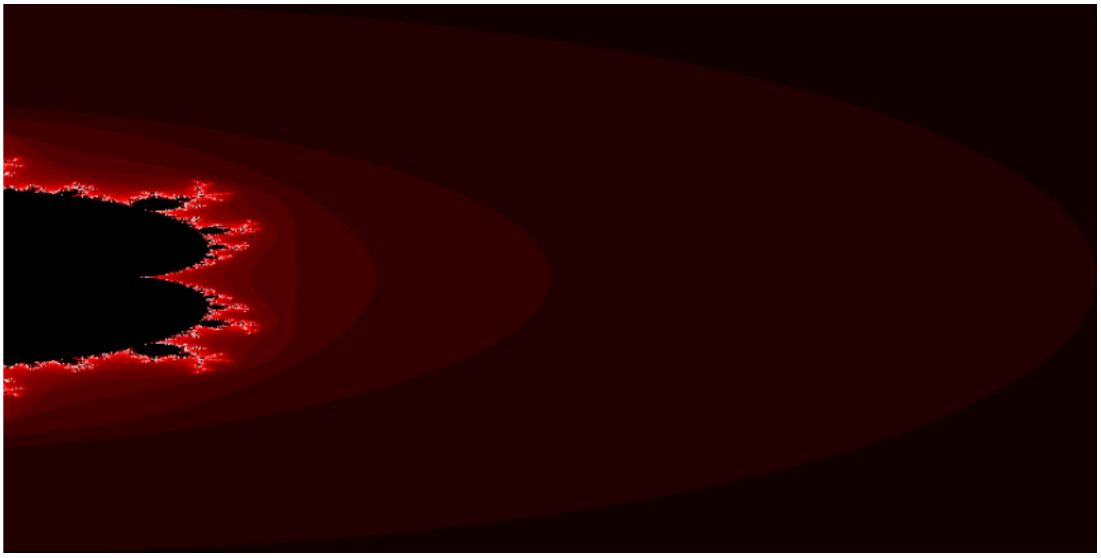
此时的图像如下



样例 B

```
$ mpicc homework6.c -O3 -pthread -lm -lpng -fopenmp -std=gnu99 -o homework
$ mpirun -n ${proc} ./homework ${thd_per_proc} 0 2 -2 2 800 400 ./homework.png
```

此时的图像如下



要求

学习 png.h 库和实现伪代码看上去比较麻烦，不用担心，**本题在附件提供了能正确运行的串行代码 sequential.c**，串行代码已完成核心功能和生成图片操作。请同学们在这个串行代码的基础上进行优化。串行代码可如下命令执行:

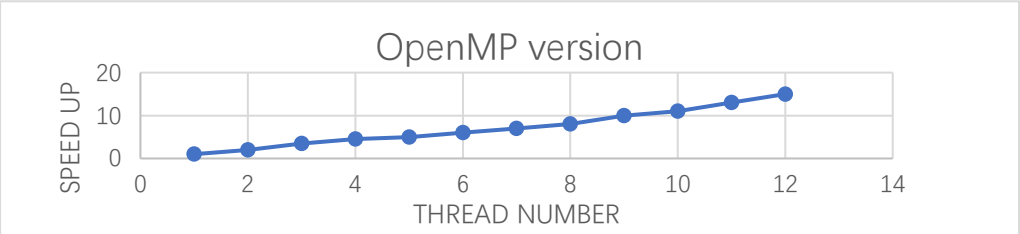
```
./${串行程序二进制文件} ${thread_per_proc} ${real_lower} ${real_upper} ${imag_lower} ${imag_upper} ${w} ${h} ${output_path}
```

本次作业需要提交**四种不同版本优化**的代码，分别是

1. **纯 MPI 静态调度(static scheduling)，命名为 mpi_static**
2. **纯 MPI 动态调度(dynamic scheduling)，命名为 mpi_dynamic**
3. **纯 OpenMP 优化，命名为 omp**
4. **MPI+OpenMP 混合优化，命名为 hybrid**
5. **并提交一份实验报告。**

实验报告需包含:

1. 学号、姓名
2. 实现
   - 遇到的问题与解决措施
   - 各种版本的优化是如何实现的
   - 是如何将任务进行分块的
   - 是如何提升运行速度与增强程序扩展性的
3. 分析与实验
   - 是如何测量程序运行时间的
   - 请用 Excel 或者其他类似工具作出比较图，比较 3 个不同的因素所带来的的**加速比**变化: **进程的数量，线程的数量和优化版本**。写清楚每张图的纵轴横轴是什么，是在什么版本运行下的。如下给了一张 OpenMP 版本的不同线程数的加速比(当然下图的数据是虚拟的，只做样例参考)，选择自己认为比较直观的图表形式即可。



   - 分析加速比随自变量变化的趋势，及发生这种变化的原因。
4. 结论
   - 通过上述比较，得出最佳的优化方案是什么，该方案的优势是什么
   - 在混合编程中，进程数量和线程数量该如何分配才能达到最佳效果

评分标准

1. 正确性(40%)
   - 能按照题目所提供的命令正确运行，并生成对应的 png 文件
   - 测试时将使用黑盒测试，每个版本的正确性须在 95%以上，可通过比较生成的 image[]数组验证
   - 运行时间最多为串行版本+30 秒
2. 性能(30%)
   - 共提供 20 个测试样例，每个样例的**分数**为
     [所有同学的所有版本在该样例的最快速度 / 你提交的四个版本在该样例的最快速度] * 1.5%
   - 最快速度是在保证正确性的情况下
3. 报告完成度(30%)

注意

1. 提交这几个文件到课程中心，**不用打包！不用打包！不用打包！**几个文件命名如下:
   - mpi_static.c 或 mpi_static.cpp
   - mpi_dynamic.c 或 mpi_dynamic.cpp
   - omp.c 或 omp.cpp
   - hybrid.c 或 hybrid.cpp
   - 如果用 makefile 编译的，可以提交一份 Makefile 文件
   - **报告命名**: 学号+姓名，word 或者 pdf 都可以

2. 务必按时提交
3. 此次将使用查重软件严查抄袭，抄袭者将给予 0 分
4. 若出现机器问题请及时上报老师或者助教
5. 一些建议:
   - 可以使用 EasyConncet+VScode Remote 连接机器，查看图片时会更方便 https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.vscode-remote-extensionpack
   - 熟悉脚本编写和 Makefile 文件编写会简化很多重复的编译运行过程。附件提供了一份运行脚本和 Makefile 作参考