并行程序第二次作业

姓名	学号
刘冠麟	20319045

p1

正常输出：

```
(base) liuglin@WIN-6G3ESV0SHI5:/mnt/d/Paper/Communication-theory/并行程序设计与算法/理论作业/hw2$ mpirun ex1
Greetings from process 0 of 6!
Greetings from process 1 of 6!
Greetings from process 2 of 6!
Greetings from process 3 of 6!
Greetings from process 4 of 6!
Greetings from process 5 of 6!
```

更改代码后：

```
(base) liuglin@WIN-6G3ESV0SHI5:/mnt/d/Paper/Communication-theory/并行程序设计与算法/理论作业/hw2$ mpic++ ex1.c -o ex1
(base) liuglin@WIN-6G3ESV0SHI5:/mnt/d/Paper/Communication-theory/并行程序设计与算法/理论作业/hw2$ mpirun ex1
Greetings from process 0 of 6!
Greetings from process 1 of 6!
Greetings from process 2 of 6!
Greetings from process 3 of 6!
Greetings from process 4 of 6!
Greetings from process 5 of 6!
```

这是因为`strlen(greeting)+1`是为了字符串末尾的 '\0'（空字符）。如果用`strlen(greeting)`而不是`strlen(greeting)+1`那么表示字符串末尾的空字符不会被传输，不过对于print输出来说并没有影响。

p2

可以使用Barrier即屏障机制来完成顺序输出。即使用`MPI_Barrier()`函数来同步进程。`MPI_Barrier()`函数会阻塞调用它的进程直到所有进程都调用了该函数。

这题中可以在每个进程打印输出之前设置一个Barrier，这样只有当所有进程都到达Barrier时，它们才会继续执行并按顺序打印输出，代码如下：

```
    MPI_Barrier(MPI_COMM_WORLD); // 同步所有进程
    printf("Proc %d of %d > Does anyone have a toothpick ?\n", my_rank, comm_sz);
```

更改前，可以看到是乱序的：

```
(base) liuglin@WIN-6G3ESV0SHI5:/mnt/d/Paper/Communication-theory/并行程序设计与算法/理论作业/hw2$ mpirun ex2
Proc 3 of 6 > Does anyone have a toothpick ?
Proc 5 of 6 > Does anyone have a toothpick ?
Proc 0 of 6 > Does anyone have a toothpick ?
Proc 1 of 6 > Does anyone have a toothpick ?
Proc 2 of 6 > Does anyone have a toothpick ?
Proc 4 of 6 > Does anyone have a toothpick ?
```

更改后，可以看到按照顺序输出：

```
(base) liuglin@WIN-6G3ESV0SHI5:/mnt/d/Paper/Communication-theory/并行程序设计与算法/理论作业/hw2$ mpirun ex2
Proc 0 of 6 > Does anyone have a toothpick ?
Proc 1 of 6 > Does anyone have a toothpick ?
Proc 2 of 6 > Does anyone have a toothpick ?
Proc 3 of 6 > Does anyone have a toothpick ?
Proc 4 of 6 > Does anyone have a toothpick ?
Proc 5 of 6 > Does anyone have a toothpick ?
```

p3

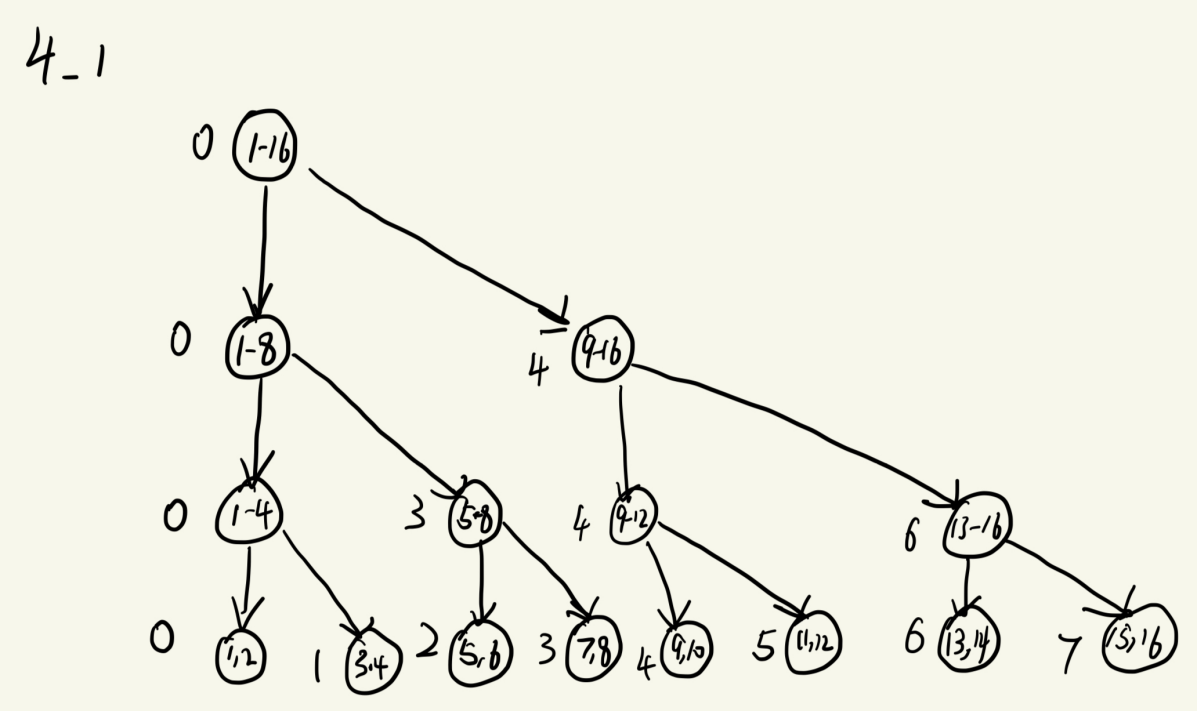
对于集合通信函数，如果通信子中只有一个进程，那么通常会**退化为串行操作**，即仅在当前进程内部进行数据操作，而不涉及进程间的通信。

具体几个函数的行为：

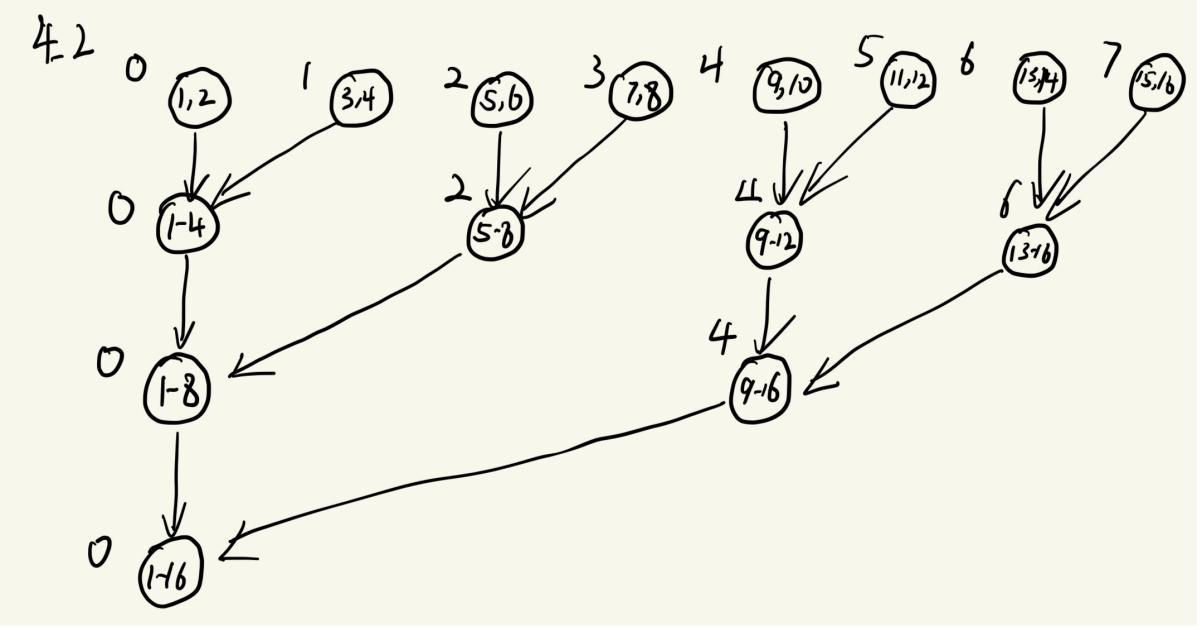
- **MPI_Bcast**：将等同于串行的广播操作。即根进程将数据广播给自己，然后继续执行。
- **MPI_Scatter**：等同于串行的分散操作。根进程将数据分散给自己，然后继续执行。
- **MPI_Gather**：等同于串行。每个进程将数据发送给根进程然后根进程收集所有数据。
- **MPI_Allgather**：等同于串行。每个进程将数据发送给自己然后继续执行。
- **MPI_Reduce**：等同于串行。根进程将数据规约到自己然后继续执行。

p4

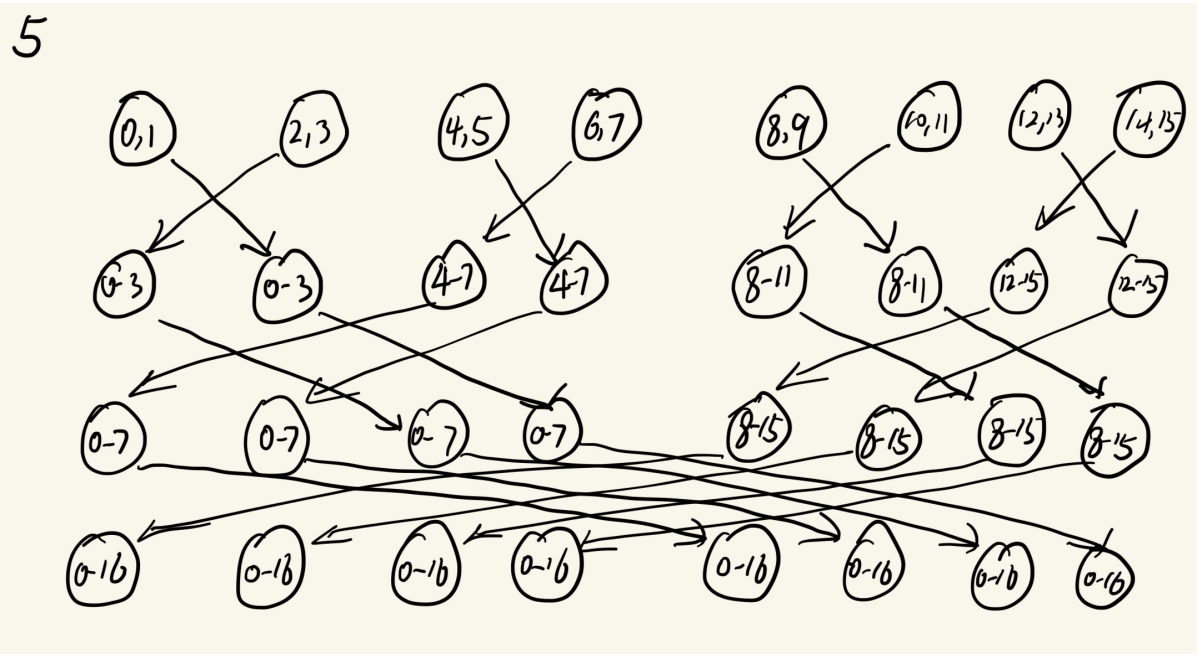
(1)



(2)


4 2
0
1,2
1
3,4
2
5,6
3
7,8
4
9,10
5
11,12
6
13,14
7
15,16
0
1-4
2
5-8
4
9-12
6
13-16
0
1-8
4
9-16
0
1-16

p5


5
0,1
2,3
4,5
6,7
8,9
10,11
12,13
14,15
0-3
0-3
4-7
4-7
8-11
8-11
12-15
12-15
0-7
0-7
0-7
0-7
8-15
8-15
8-15
8-15
0-16
0-16
0-16
0-16
0-16
0-16
0-16
0-16