# Problem1:

# 1.What have you done?

在有關 MPI 變數的部分，我宣告:
(1)my_rank 紀錄目前在哪個 rank.
(2)comm_size 是 process 的數量。
(3)my_count 是從 latter process 傳送到 former process 的 solution 數量(receive_count)和 former process solution 數量的加總。

我使用樹狀結構，因此宣告:
(1)remain_process : 紀錄還有多少 process 需要傳送或接收 value，當 remain_process ==1 時結束 tree structure communication.
(2)receive_count:記錄從 latter process 收到 solution 的數量有多少，會加到 my_count 裡，之後 my_rank==0 的時候，count=my_count，並印出時間及 solution 的數量(count)。

我用 while loop 來實現 tree structure communication : 每一次迴圈會把 process 切一半，前半部 process 接收從後半部 process 傳過來的 receive_count，並與 my_count 做相加，直到 remain_process==1 時跳出迴圈。在 my_rank==0 時，把 my_count 附值給 count，並印出時間及 solution 的數量(count)。

# 2.Analysis on your result

## (1)使用 tree structure communication 時，執行速度 :

## 12 cpu(0.000579 sec)>8 cpu(0.000621 sec) >4 cpu(0.001498 sec)

4 cpu:

```
H54084078@pn1:~> mpiexec -n 4 ./problem1.out
1) 1001100111110101
1) 1001101111110101
1) 1001110111110101
2) 1001100111110110
2) 1001101111110110
2) 1001110111110110
3) 1001100111110111
3) 1001101111110111
3) 1001110111110111
Process 0 finished.

A total of 9 solutions were found.

Process 0 finished in time 0.001498 secs.
H54084078@pn1:~>
```

8 cpu:

```
H54084078@pn1:~> mpiexec -n 8 ./problem1.out
7) 1001100111110111
7) 1001101111110111
7) 1001110111110111
6) 1001100111110110
6) 1001101111110110
6) 1001110111110110
5) 1001100111110101
5) 1001101111110101
5) 1001110111110101
Process 0 finished.

A total of 9 solutions were found.

Process 0 finished in time 0.000621 secs.
H54084078@pn1:~>
```

12 cpu:

```
H54084078@pn1:~> mpiexec -n 12 ./problem1.out
10) 1001110111110110
1) 1001101111110101
2) 1001101111110110
5) 1001100111110101
3) 1001101111110111
9) 1001110111110101
7) 1001100111110111
6) 1001100111110110
11) 1001110111110111
Process 0 finished.

A total of 9 solutions were found.

Process 0 finished in time 0.000579 secs.
H54084078@pn1:~>
```

(2)serial communication vs. tree structure communication 的執行速度(8 顆 cpu): tree structure communication(0.000621 sec) > serial communication(0.001770 sec)

serial communication:

```
0) 1001100111110101
0) 1001100111110110
0) 1001100111110111
0) 1001101111110101
0) 1001101111110110
0) 1001101111110111
0) 1001110111110101
0) 1001110111110110
0) 1001110111110111
Process 0 finished in time 0.001770 secs.

A total of 9 solutions were found.

H54084078@pn1:~>
```

tree structure communication:

```
H54084078@pn1:~> mpiexec -n 8 ./problem1.out
7) 1001100111110111
7) 1001101111110111
7) 1001110111110111
6) 1001100111110110
6) 1001101111110110
6) 1001110111110110
5) 1001100111110101
5) 1001101111110101
5) 1001110111110101
Process 0 finished.

A total of 9 solutions were found.

Process 0 finished in time 0.000621 secs.
H54084078@pn1:~>
```

# 3.Any difficulites?

在看懂 mpi 運作原理就花了不少時間，然後我也花很多時間思考 tree structure communication 怎麼實作，寫完 circuit satisfiability problem 讓我更了解 mpi 的運作流程。

# 4.(optional) Feedback to TAs

我覺得助教寫的如何繳交作業到 server 寫得很好，幫我省了不少時間去摸索

如何上傳檔案到 server.

# Problem2:

# 1.What have you done?

num_of_toss 是 toss 的總數，使用者可自行輸入
num_in_cir 是在 circle 裡面的 toss 總數

在 MPI 方面，我宣告:
(1)comm_size : total number of process
(2)my_rank : 紀錄目前在哪個 rank
(3)my_num_in_circle: 是從 latter process 傳送到 former process 的 toss 數量(recv_num_in_circle)和 former process toss 數量的加總。

我使用樹狀結構，因此宣告:
(1) remain_process:紀錄還有多少 process 需要傳送或接收 value，當 remain_process==1 時結束 tree structure communication.
(2)recv_num_in_circle:記錄從 latter process 收到 toss 的數量有多少，會加到 my_num_in_circle 裡，之後 my_rank==0 的時候，
num_in_cir=my_num_in_circle，並印出時間及運算 pi 的估計值並印出。

一開始我 Broadcast ”num_of_toss”給每個 process，接著我用 while loop 來實現 tree structure communication : 每一次迴圈會把 process 切一半，前半部 process 接收從後半部 process 傳過來的 recv_num_in_circle，並與 my_num_in_circle 做相加，直到 remain_process==1 時跳出迴圈。在 my_rank==0 時，把 my_num_in_circle 附值給 num_in_cir，並印出時間及估算 pi 的值並印出。

# 2.Analysis on your result

## (1)toss 數量影響 pi 的精確度:

10,000,000 tosses(pi=3.141331)>

10,000 tosses(pi=3.176000)>

1,000 tosses(pi=3.072000)>

100 tosses(pi=3.520000)>

10 tosses(pi=4.000000)

10 tosses:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 10

PI: 4.000000
Process 0 finished in time 0.002688 secs.
```

100 tosses:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 100

PI: 3.520000
Process 0 finished in time 0.003231 secs.
```

1,000 tosses:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 1000

PI: 3.072000
Process 0 finished in time 0.002881 secs.
```

10,000 tosses:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 10000

PI: 3.176000
Process 0 finished in time 0.003333 secs.
```

10,000,000 tosses:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 10000000

PI: 3.141331
Process 0 finished in time 0.051305 secs.
```

並且發現 toss 數越多，pi 計算到小數點以下的位數會越多

位

(2) serial communication vs. tree structure communication 的執行速度(8 顆 cpu、10,000,000 tosses): tree structure communication(0.051305 sec)>serial communication(0.272327 sec)

serial communication:

```
H54084078@pn1:~> mpiexec -n 8 ./Monte_Carlo_method.out
Process finished in time 0.272327 secs.
Process finished in time 0.272337 secs.
Process finished in time 0.274181 secs.
Process finished in time 0.274616 secs.
Process finished in time 0.282490 secs.
Process finished in time 0.293015 secs.
Process finished in time 0.303159 secs.
Process finished in time 0.400139 secs.
Pi: 3.141860
Pi: 3.141860
Pi: 3.141860
Pi: 3.141860
Pi: 3.140821
Pi: 3.140821
Pi: 3.140821
Pi: 3.140821
H54084078@pn1:~>
```

Tree structure communication:

```
H54084078@pn1:~> mpiexec -n 8 ./problem2.out
Enter the number of tosses: 10000000

PI: 3.141331
Process 0 finished in time 0.051305 secs.
```

# 3.Any difficulites?

跟 problem1 一樣，因為還不熟悉 mpi，花了很多時間了解 mpi 的原理跟 tree structure communication.

# 4.(optional) Feedback to TAs

很謝謝助教開放 homework 討論區，助教也會熱心回答我們遇到的問題