
OS HW4

mutex & semaphore

Prof. 蔡文錦

TA. 林孟學 王彥珽 姚淨云 張皓雲

Deadline: 2022/1/8 (Sat) PM11:55

Tasks

1. Series

1-1. Series - 1 (30%)

1-2. Series - 2 (20%)

2. Pi (30%)

3. Report (20%)

1-1. Series -1

- Calculate the total number of occurrences of each number in the series.
- Values in series: [0, 1, 2]
- Use multi-thread(1~4) and mutex/semaphore

Ex.

Input : 4
 15
 120201202011021

Output :

0: 5
1: 5
2: 5

1-2. Series -2

- Calculate the total number of occurrences of each number in the series.
- Values in series: [0, 1, 2]
- Use multi-thread(**3**) and mutex/semaphore.

Ex.

Input :

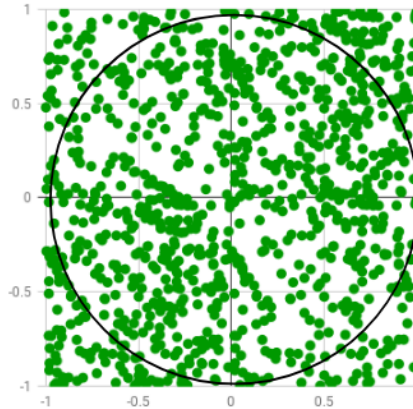
15
120201202011021

Output :

0: 5
1: 5
2: 5

2. Pi

- Estimate the value of Pi using Monte Carlo.
- The "Monte Carlo Method" is a method of solving problems using statistics.



- Algorithm reference:

<https://www.geeksforgeeks.org/estimating-value-pi-using-monte-carlo/>

Synchronization - mutex lock

- Header
`#include <pthread.h>`
- Declare: (global variable)
`pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;`
- Functions:
 - `pthread_mutex_lock()`
 - `pthread_mutex_unlock()`
 - `pthread_mutex_trylock()`

Synchronization - semaphore

- Header
`#include <pthread.h>`
 - Declare: (global variable)
`pthread_cond_t cond = PTHREAD_COND_INITIALIZER;`
 - Functions:
 - `pthread_cond_wait()`
 - `pthread_cond_signal()`
 - `pthread_cond_broadcast()`
- Header
`#include <semaphore.h>`
 - Declare: (global variable)
`sem_t sem;`
 - Functions:
 - `sem_post()`
 - `sem_wait()`
 - `sem_close()`

Series - 1 (30%) - Restrictions:

1. You should use only **ONE** global array of size 3 to update number counts.

```
long long counts[3] = {}; // datatype and variable name can be any.
```

2. You should update the counts values each time you process the string.

NOT count the numbers by threads, and update it to global variables at the end of thread.

```
(O) for(int i=begin; i<end; i++) counts[arr[i] - '0'] += 1;
```

```
(X) for(int i=begin; i<end; i++) localc[arr[i] - '0'] += 1;  
    for(int i=0; i<3; i++)      counts[i] += localc[i];
```

of course you can use **mutex** or **semaphore** to prevent race condition.

Series - 1 (30%) - input / output:

- Input

```
Number of threads ( 1 ~ 4 ) -> 3
Series length ( 1 ~ 10^8 ) -> 15
The series -> 120201202011021
```

12020 12020 11021
↓ ↓ ↓
Thread1 Thread2 Thread3

Each thread is responsible for $1/3$ of the series.

- Output

```
0:(space)counts[0](newline)
1:(space)
...(newline)
```

```
0: 1
1: 2
2: 1
```

```
Compile with:
$ g++ -o hw4_1_1 hw4_1_1.c -lpthread
Test with:
$ ./hw4_1_1 < input.txt > my_ans.txt
Check answer:
$ ./diff -w -b -B answer.txt my_ans.txt
```

Series - 2 (20%) - Restrictions:

1. You should use only **ONE** global array of size 3 to update number counts. (same as above)

```
long long counts[3] = {}; // datatype and variable name can be any.
```

2. You should update the counts values each time you process the string.

NOT count the numbers by threads, and update it to global variables at the end of thread.

```
(O) for(int i=begin; i<end; i++) counts[arr[i] - '0'] += 1; (same as above)
```

```
(X) for(int i=begin; i<end; i++) localc[arr[i] - '0'] += 1;  
    for(int i=0; i<3; i++)      counts[i] += localc[i];
```

of course you can use **mutex** or **semaphore** to prevent race condition.

3. You should output the result **by each thread in order**. (new)

Output "0: %d\n" by thread 0, "1: %d\n" by thread 1, "2: %d\n" by thread 2.

NOT output it in main region(all thread end).

Series - 2 (20%) - input / output:

- Input

Series length ($1 \sim 10^8$) ->
The series ->

```
15  
120201202011021
```

12020 12020 11021
↓ ↓ ↓
Thread1 Thread2 Thread3

Each thread is responsible for $1/3$ of the series.

- Output

```
0:(space)counts[0](newline)  
1:(space)  
...(newline)
```

```
0: 1  
1: 2  
2: 1
```

```
Compile with:  
$ g++ -o hw4_1_2 hw4_1_2.c -lpthread  
Test with:  
$ ./hw4_1_2 < input.txt > my_ans.txt  
Check answer:  
$ ./diff -w -b -B answer.txt my_ans.txt
```

Pi (30%) - Restrictions:

1. You should use only **ONE** global variable to update counts

```
long long counts = 0; // datatype and variable name can be any.
```

2. You should update the counts values each time you process the string.

NOT count the numbers by threads, and update it to global variables at the end of thread.

```
(O) for(int i=begin; i<end; i++) if(point in circle) counts += 1;
```

```
(X) for(int i=begin; i<end; i++) if(point in circle) localc += 1;  
    counts += localc;
```

of course you can use **mutex** or **semaphore** to prevent race condition.

Pi (30%) - input / output:

- Input

```
Number of threads ( 1 ~ 4 ) -> 4  
Number of points ( 1 ~ 10^8 ) -> 100000
```

```
#include <stdlib.h>  
// # include <cstdlib> in c++  
  
// srand(), rand(), RAND_MAX  
  
// ----- or  
  
# include <random>  
// Reference: https://wwwcplusplus.com/reference/random/uniform\_real\_distribution/
```

- Output

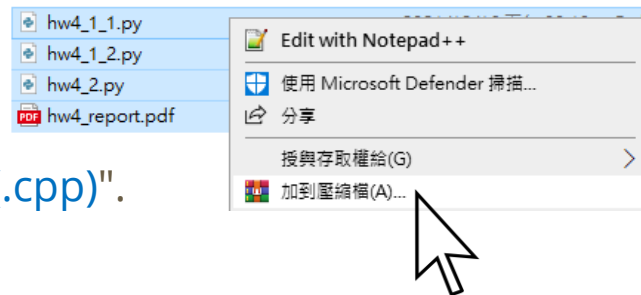
```
get:(space)counts(newline)  
Pi:(space)result(newline)
```

```
get: 78513  
Pi: 3.14052
```

```
Compile with:  
$ g++ -o hw4_2 hw4_2.c -lpthread  
Test with:  
$ ./hw4_2 < input.txt > my_ans.txt  
Check answer:  
$ no.
```

Requirements

1. You should write codes in `c/c++`.
2. Put all `*.c(.cpp)` source files and report(*.pdf) into same compressed file.
The type of compressed file must be `"zip"`.
3. The name of your compressed file must have the form of `"studentID_OS_hw4.zip"` and submit it **WITHOUT FOLDER**.
4. The name of `.*c/*.cpp` file must in the form of `"hw4_1_1.c(.cpp)"` & `"hw4_1_2.c(.cpp)"` & `"hw4_2.c(.cpp)"`.
5. Report: format is in `hw4_report.docx`.
And please **export to PDF** file(`hw4_report.pdf`) before submitting.



Grade

Total score: 100pts. **COPY WILL GET 0 POINT!**

- HW4 - 1 - 1 | HW4 - 1 - 2 | HW4 - 2 | report
30 pts 20 pts 30 pts 20 pts
(not meet the requirements(restrictions) will also **get 0 pts in that question.**
- Incorrect file form: -20 pts
(Including the names of compressed file, .c(.cpp) files, report file type)
- Deadline: 2022/1/8 (Sat) PM11:55
Late submission will get a **-20% point per day**
e.g. write HW4 - 1 - 1 and HW4 - 1- 2 only, and submit it 2 days late will get:
 $(30 + 20) * (100\% - 2 * 20\%) = 30 \text{ pts}$