# الاسم : احمد عزت جابر على

# الفرقة الرابعة علوم حاسب

# سكشن 1

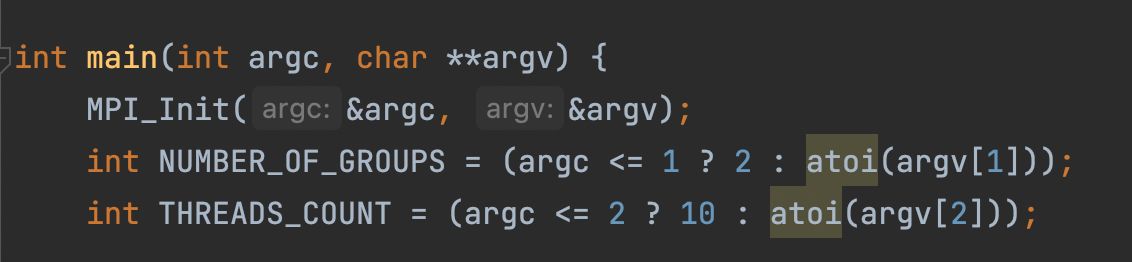
# How to Run the code

I am using “mpirun” in MAC environment, use this command, with variables:

mpirun -np {mpi\_processes} main.o {number\_of\_groups} {threads\_count}

* number\_of\_groups: the variable used in the code to divide the MPI nodes, have a default value equal to **2**.

Threads\_count: represent number of threads used in each node, have a default value **10**.



# Split the nodesGraphical user interface, text Description automatically generated

Assume we have 6 nodes, and number\_of\_groups is 2.

I am split the nodes according to its reminder (%number\_of\_groups) like this table.

|  |  |  |
| --- | --- | --- |
| 0 | 2 | 4 |
| 1 | 3 | 5 |

Then for each row we will have a communicator (row\_comm), and for each column will have another communicator (col\_comm). Groups:

* 0, 2, 4 -> row\_comm
* 1, 3, 5 -> row\_comm
* 0, 1 -> col\_comm
* 2, 3 -> col\_comm
* 4, 5 -> col\_comm

Each row\_comm will have the whole database, and a part of query. And inside each node in the communicator will have a part of database. In out example it will be like this:

|  |  |  |
| --- | --- | --- |
| 0 (database\_1/3), (query\_1/2) | 2 (database\_2/3), (query\_1/2) | 4 (database\_3/3), (query\_1/2) |
| 1 (database\_1/3), (query\_2/2) | 3 (database\_2/3), (query\_2/2) | 5 (database\_3/3), (query\_2/2) |

# Generate and split the data: read\_data function

* First, I generate database and queries as random strings in the node 0, then broadcast the whole database to in the first column communicator, now the data will be like:

|  |  |  |
| --- | --- | --- |
| 0 (all database, and half of queries) | 2 | 4 |
| 1 (all database, and second half of queries) | 3 | 5 |

* Second, I Scatter the database in each row, so the data will be like:

|  |  |  |
| --- | --- | --- |
| 0 (database\_1/3), (query\_1/2) | 2 (database\_2/3) | 4 (database\_3/3) |
| 1 (database\_1/3), (query\_2/2) | 3 (database\_2/3) | 5 (database\_3/3) |

* Then, I Broadcast the queries in each row, so the data will be like:

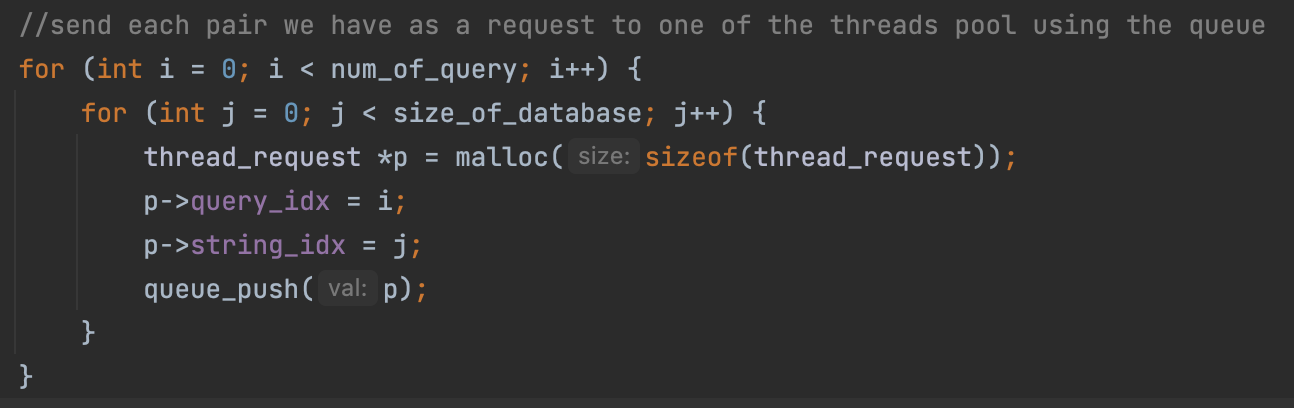
|  |  |  |
| --- | --- | --- |
| 0 (database\_1/3), (query\_1/2) | 2 (database\_2/3), (query\_1/2) | 4 (database\_3/3), (query\_1/2) |
| 1 (database\_1/3), (query\_2/2) | 3 (database\_2/3), (query\_2/2) | 5 (database\_3/3), (query\_2/2) |

# myqueue.h

this file conations a queue implantation using a linked list

# Calculation in each process

* First, each node creates a thread pool of size Threads\_Count, which we have taken from the argument, and use thread\_work function in it.
* The main thread pushes all pairs to the queue (in myqueue.h) to process it in one of the threads pools.



* Now each thread will start processing the pairs from the queue, calculate the edit distance, and the store the result in output array to print it later.

Text

Description automatically generated

## Why I’m using this approach?

the work will be balanced between each CPU core, the working threads will process the requests we need, if any thread are waiting in the scheduler, then will be another thread to handle the request.

# Gather output data from nodes

After the Calculation, each node has its part of output, as shown below.

|  |  |  |
| --- | --- | --- |
| 0 output\_0 | 2 output\_2 | 4 output\_4 |
| 1 output\_1 | 3 output\_3 | 5 output\_5 |

So, for each query we gather the output from each row to the first node in row

Graphical user interface, text

Description automatically generatedNow the data will be like:

|  |  |  |
| --- | --- | --- |
| 0 output\_(0,2,4) | 2 | 4 |
| 1 output\_(1,3,4) | 3 | 5 |

After that we gather using the col\_comm for the first columnA picture containing text

Description automatically generatedNow we have the whole output in the node 0 and we can print it.