High Performance Computing
Sultan Asiri
Final Project

1. Accept arbitrary matrix sizes. You can assume that both matrices will fit in global memory.

To do this task we need to consider three things:

1. Reading .mtx file
2. Get number of rows.
3. Get number of columns.

I used the fstream library for c++. With it, I can read an external file and get its beginning and its ending. This step can help me to store the number of rows and columns in the struct matrix. By knowing the rows and columns, we can see the size of the file matrix (see fig.1). I also add a condition to test the matrix size. If they can't do multiplication, it will return an error and stop the program.

```
if(argc != 4) {//there should be four arguments
    printf("missing argumnet");
    return 1; //exit and return an error
}
time_t reading_start=time(NULL);

ifstream infile_A, infile_B;    //reading the input matrices

// ******************************************************************************
//                                   Matrix A                                  //
//******************************************************************************

infile_A.open(argv[1],ios::binary|ios::in|ios::ate);

//getting end and beginning of the file
infile_A.seekg(0,ios::end);
infile_A.seekg(0,ios::beg);

//memory allocation
matrix M_A;
infile_A.read(reinterpret_cast<char*>(&M_A),2*sizeof(unsigned int));


float* array_A=(float*)malloc(M_A.rows*M_A.cols*sizeof(float)); //column major
infile_A.read(reinterpret_cast<char*>(array_A),M_A.rows*M_A.cols);

infile_A.close();
print(array_A,M_A.rows,M_A.cols);
```

Fig.1 sample code of reading an external file

2. Implement the matrix multiplication using the function in cuBLAS library (i.e., SGEMM)

   a. Profile this implementation on PantaRhei GPU node and output the performance in GFLOPS

All cuda programming need the seven steps which is:

1. Setup inputs on the host (CPU-accessible memory):

   b. In this step my program will get the input from previous task and allocate it in the CPU, Then allocate it in the host with size of the array.

```
float* array_A=(float*)malloc(M_A.rows*M_A.cols*sizeof(float)); //column major
float* array_B=(float*)malloc(M_B.rows*M_B.cols*sizeof(float));
```

2. Allocate memory for outputs on the host

   c. In this step I initialize the output array and allocate it in CPU then in the host with size of input array a rows * array b columns.

```
float* array_D=(float*)malloc(M_A.rows*M_B.cols*sizeof(float));
```

3. Allocate memory for inputs on the GPU

   d. In this step first I create new arrays to allocate it in GPU. This array will take the size of the array in the CPU.

```
// ***************************************************************************
//                              allocate to the GPU                                                              //
//***************************************************************************
float *array_A_gpu, *array_B_gpu,  *array_D_gpu;   //gpu arrays declared

cudaMalloc(&array_A_gpu,M_A.rows*M_A.cols*sizeof(float)); //allocate space to store arrayA

cudaMalloc(&array_B_gpu,M_B.rows*M_B.cols*sizeof(float)); //allocate space to store arrayB

cudaMalloc(&array_D_gpu,M_A.rows*M_B.cols*sizeof(float)); //allocate space to store cublas result
```

4. Allocate memory for outputs on the GPU

   i. As previous , I create a new array and copy it from the host to the GPU.

```
cudaMemcpy(array_D_gpu, array_D, M_A.rows*M_B.cols*sizeof(float), cudaMemcpyHostToDevice);//copy
arrayD to gpu
```

5. Copy inputs from host to GPU

   e. In this step I copy the array from the host to GPU

```
//COPY TO GPU MEMORY
cudaMemcpy(array_A_gpu, array_A, M_A.rows*M_A.cols*sizeof(float), cudaMemcpyHostToDevice);//copy
arrayA to gpu

cudaMemcpy(array_B_gpu, array_B, M_B.rows*M_B.cols*sizeof(float), cudaMemcpyHostToDevice);//copy
arrayB to gpu

cudaMemcpy(array_D_gpu, array_D, M_A.rows*M_B.cols*sizeof(float), cudaMemcpyHostToDevice);//copy
arrayD to gpu
```

6. Start GPU kernel (function that executed on gpu)

   f. In this step I start work on the cublas. Since our data is 32 bit float,I used SGEMM.

```
//Creating handle for CUBLAS
   float milliseconds2 = 0;
   cublasHandle_t handle;
   cublasCreate(&handle);

   //parameter declaration for cublas implementation
   float alpha = 1.0;
   float beta = 0.0;

   //cublas time measurement
   cudaEvent_t start2, stop2;

   cudaEventCreate(&start2);
   cudaEventCreate(&stop2);

   //MATRIX MULTIPLICATION USING CUBLAS

   cudaEventRecord(start2);
   for( int s=0; s<20;s++){
      cublasSgemm(handle, CUBLAS_OP_N, CUBLAS_OP_N, M_A.rows, M_B.cols, M_A.cols, &alpha,
array_A_gpu, M_A.rows, array_B_gpu, M_B.rows, &beta, array_D_gpu, M_A.rows);
   }
}
```

   g. Create a the cublas Handle_t value to execute cuBLASfunctions
      i. Use some optional data such as apha 1, beta 0.
      ii. I put SGEMM method in iteration to warm it and to get the best result.
      iii. Call SGEMM method.

7. Copy output from GPU to host

   iv. This. The last step in cuda programming which will be copy the output array into the host.

```
cudaMemcpy(array_D, array_D_gpu, M_A.rows*M_B.cols*sizeof(float), cudaMemcpyDeviceToHost);//copy result of multiplication using CUBLAS from gpu to cpu
```

   v. Then Thread Synchronize, and stop the event and get the time and GFLOPS.

```
cudaEventRecord(stop2);
cudaEventSynchronize(stop2);
cudaEventElapsedTime(&milliseconds2, start2, stop2);//get the time in milliseconds
float msecPerMatrixMul = milliseconds2 / 20;
double flopsPerMatrixMul = 2.0 * (double) M_A.rows *(double) M_B.cols *(double) M_A.cols;
double gigaFlops = (flopsPerMatrixMul * 1.0e-9f) / (msecPerMatrixMul / 1000.0f);
printf("Performance= %.2f GFlop/s, Time= %.3f msec, Size= %.0f Ops\n",gigaFlops,msecPerMatrixMul,
flopsPerMatrixMul);
```

Result of cublas:

Matrix size		Gflops	Time (ms)
Matrix 1	**Matrix 2**		
3*3	**3*2**	**0**	**0.035**
32*32	**32*32**	**1.28**	**0.051**
1024*1024	**1024*1024**	**8601.94**	**0.250**
2048*2048	**2048*2048**	**11433.26**	**1.503**

3. Implement the matrix multiplication using shared memory

- **Profile this implementation on PantaRhei GPU node and output the performance in GFLOPS**
- **Compare your result to the result of cuBLAS gemm function**

- For this task I used the input and the first five steps in cuda programing previous task.
- The different will be in step 6 which is Start GPU kernel (function that executed on gpu)

```
__global__ void matrix_mult(float* array1, unsigned int rows1, unsigned int cols1, float* array2, unsigned int
rows2, unsigned int cols2, float* array3)
{
   //shared memory takes one tile at a time
   __shared__ float S1[TILE_WIDTH][TILE_HEIGHT];  //to store tiles for array 1
   __shared__ float S2[TILE_HEIGHT][TILE_WIDTH];  //to store tiles for array 2

   //threads x and y index for the current block
   unsigned int tx=threadIdx.x;
   unsigned int ty=threadIdx.y;

   unsigned int c=blockIdx.x*blockDim.x + threadIdx.x;    //row value using x-index of current thread
   unsigned int r=blockIdx.y*blockDim.y + threadIdx.y;    //column value using y-index of current thread

   unsigned int idx=c*rows1+r;            //column major index, using row and column value

   float val=0;      //register to store multiplication result initialized to zero

   for(int m=0; m<1+((rows2-1)/TILE_WIDTH);m++)    //going over all tiles one by one, with each m
   {

       int var1=m*TILE_WIDTH+tx ;    //x thread value for current tile
       int var2=m*TILE_WIDTH+ty ;    //y thread value for current tile

       //copying a tile from array1
       if (r < rows1 && var1 < rows2)    //if the value is associated to a valid matrix coordinate in array1 then store it
to shared memory S1
          S1[ty][tx]=array1[r + var1*rows1];//storing a "valid" value from array to shared memory
       else
          S1[ty][tx]=0;             //storing zero, since there is no valid value
       __syncthreads();                 //syncing all threads once shared memory S1 is stored

       //copying a tile from array2
       if(c < cols2 && var2 < rows2)  //if value is associates to a valid matrix coordinate in array2 then store it to
shared memory S2
          S2[ty][tx]=array2[var2+rows2*c];   //storing the valid value
       else
          S2[ty][tx]=0;     //storing zero, since no valid value
       __syncthreads();      //synchronizing threads


       for(int i=0; i<TILE_WIDTH;i++) //going over entire tile, ty row in S1 and tx column in S2
          val+=S1[ty][i]*S2[i][tx];  //and multiplying elements
       __syncthreads();      //synchronizing threads
```

```
  }

  if(r < rows1 && c< cols2)  //removing degenerate cases
     array3[idx]=val;   //saving multiplication result to global memory
}
```

a. In this step I used same function as provide from the GEMM slide with adding some important requirement . Which is :
   i. BX= The block id of x ,
   ii. BY= The block id of x ,
   iii. TX= Thread id x
   iv. Ty= Thread id x
   v. rows= will calculate the value of the block size x and thread size x multiplying with the TILE_width
   vi. cols= will calculate the value of the block size y and thread size y multiplying with the TILE_width
   vii. There is also some conditions as shown in the code that need to be consider.
b. Before I called the function I need to create dimension for the grid and for the block in the main function

```
cudaDeviceProp prop;
cudaGetDeviceProperties(&prop, 0); //using GPU0

//BLOCK AND GRID SIZE DECLARATION
float thread_block=sqrt(prop.maxThreadsPerBlock);  //2D blocks used
dim3 DimGrid(ceil(M_B.cols/thread_block),ceil(M_A.rows/thread_block),1); //image saved as a 2D grid
dim3 DimBlock(thread_block,thread_block,1);

size_t Sbytes = 2* DimBlock.x * DimBlock.y ;   //2 arrays used in the calculation, hence 2 * DimBlock.x *
DimBlock.y
```

c. Then, I called the function.

```
matrix_mult<<<DimGrid,
DimBlock,Sbytes>>>(array_A_gpu,M_A.rows,M_A.cols,array_B_gpu,M_B.rows,M_B.cols,array_C_gpu);//calling
the kernel
```

After that I did the last step with coda programing steps which is copy the output to the hots and than stop the event and calculate the time and GFLOPS.

```
cudaMemcpy(array_D, d_c, sizeof(float)*M_A.rows*M_B.cols, cudaMemcpyDeviceToHost);
cudaThreadSynchronize();
// time counting terminate
cudaEventRecord(stop, 0);

float milliseconds1 = 0;//storing the execution time in milliseconds

cudaEventElapsedTime(&milliseconds1, start1, stop1);//get the time in milliseconds
float msecPerMatrixMul = milliseconds1;
double flopsPerMatrixMul = 2.0 * (double) M_A.rows *(double) M_B.cols *(double) M_A.cols;
```

```
double gigaFlops = (flopsPerMatrixMul * 1.0e-9f) / (msecPerMatrixMul / 1000.0f);
printf("Performance= %.2f GFlop/s, Time= %.3f msec",gigaFlops,msecPerMatrixMul);
```

d. To evaluate the result of my programming I compare it with CPU implementation and see if they have same result
   i. This step is successes with different size of TILE_WIDTH

e. Result of cuda shared memory:

   i. Performance based on time (second)

Performance by Time					
Matrix size		TILE_WIDTH			
Matrix 1	**Matrix 2**	**8**	**16**	**32**	**64**
3*3	**3*2**	0.059	0.062	0.080	0.068
32*32	**32*32**	0.061	0.060	0.060	0.062
1024*1024	**1024*1024**	2.205	1.442	0.954	1.014
2048*2048	**2048*2048**	16.211	10.626	7.416	7.433

   ii. Performance based on Gflops.

Performance by GFLOPS					
Matrix size		**Block Size**			
Matrix 1	**Matrix 2**	**8**	**16**	**32**	**64**
3*3	**3*2**	0.000	0.000	0.000	0.000
32*32	**32*32**	1.07	1.10	1.09	1.06
1024*1024	**1024*1024**	974.10	1489.55	2249.86	2117.93
2048*2048	**2048*2048**	1059.78	1616.79	2316.74	2311.39

The result shows that my implementation can’t achieve higher than 1 teraflop. At the same time, Cublas shows can make more than that can reach until ten teraflops. So, it is clear that Cublas show a better result than my implementation using shared memory.

4. Compare the performance of your GPU implementation with the CPU code that you implemented in Project 2.

For this comparison I used the best algorithm performance in my project 2 which is kji. The result of comparison as shown in next two tables.

GPU Performance by Time					
Matrix size		Block Size			
Matrix 1	Matrix 2	8	16	32	64
3*3	3*2	0.059	0.062	0.080	0.068
32*32	32*32	0.061	0.060	0.060	0.062
1024*1024	1024*1024	2.205	1.442	0.954	1.014
2048*2048	2048*2048	16.211	10.626	7.416	7.433
CPU Performance by Time					
Matrix size		Block Size			
Matrix 1	Matrix 2	8	16	32	64
3*3	3*2	0.000003	0.000017	0.000146	0.001058
32*32	32*32	6.060332	9.914468	13.650993	13.342011
1024*1024	1024*1024	0.000137	0.000131	0.000136	0.001069
2048*2048	2048*2048	46.578765625	80.240548	114.025109	847946

GPU Performance by GFLOPS					
Matrix size		Block Size			
Matrix 1	Matrix 2	8	16	32	64
3*3	3*2	0.000	0.000	0.000	0.000
32*32	32*32	1.07	1.10	1.09	1.06
1024*1024	1024*1024	974.10	1489.55	2249.86	2117.93
2048*2048	2048*2048	1059.78	1616.79	2316.74	2311.39
CPU Performance by GFLOPS					
Matrix size		Block Size			
Matrix 1	Matrix 2	8	16	32	64
3*3	3*2	0.004693	0.000726	0.000082	0. 000011
32*32	32*32	0.000346	0.000212	0.000154	0.000157
1024*1024	1024*1024	0.014924	0.015658	0.574661	0.001915
2048*2048	2048*2048	0.000180	0.000105	0.000074	0.000093

In this part, I implemented my code in project 2 in the four datasets. In the previous table, I chose the highest gflops from all implementation, which is kji with blocking. So, based on this table, it is clear that the GPU shows better performance than the CPU. However, some of the datasets show that the CPU is faster than the GPU.

Note: All implementation was on the DMC cluster with one GPU, 1 GP, and. Volta.

5. Run my cude:

1. My folder has three file all these files can be run using the next command with different of the file name.
    a. ./multi.cublas a_1024_1024.mtx b_1024_1024.mtx c.mtx

6. Project 2 reuslt

---3*2---

```
-----------------------------|ijk algorithm done no block|--------------------------------------
B= 8
runTime = 0.000002 s   2180 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.005505
-----------------------------|ijk algorithm done|-----------------------------------------
runTime = 0.000002 s   2251 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.005331
-----------------------------|jik algorithm done|-----------------------------------------
runTime = 0.000002 s   2499 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.004802
-----------------------------|kij algorithm done|-----------------------------------------
runTime = 0.000003 s   2557 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.004693
-----------------------------|ikj algorithm done|-----------------------------------------
runTime = 0.000003 s   2601 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.004614
-----------------------------|jki algorithm done|-----------------------------------------
runTime = 0.000003 s   2557 ns   number of FloatingPoint Operations = 12    GFLOPS = 0.004693
-----------------------------|kji algorithm done|-----------------------------------------
B= 16
runTime = 0.000013 s   13189 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000910
-----------------------------|ijk algorithm done|-----------------------------------------
runTime = 0.000013 s   13362 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000898
-----------------------------|jik algorithm done|-----------------------------------------
runTime = 0.000016 s   16107 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000745
-----------------------------|kij algorithm done|-----------------------------------------
runTime = 0.000016 s   16231 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000739
-----------------------------|ikj algorithm done|-----------------------------------------
runTime = 0.000017 s   16586 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000724
-----------------------------|jki algorithm done|-----------------------------------------
runTime = 0.000017 s   16540 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000726
-----------------------------|kji algorithm done|-----------------------------------------
B= 32
runTime = 0.000100 s   99983 ns  number of FloatingPoint Operations = 12    GFLOPS = 0.000120
-----------------------------|ijk algorithm done|-----------------------------------------
runTime = 0.000106 s   106267 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000113
-----------------------------|jik algorithm done|-----------------------------------------
runTime = 0.000132 s   132048 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000091
-----------------------------|kij algorithm done|-----------------------------------------
runTime = 0.000131 s   130996 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000092
-----------------------------|ikj algorithm done|-----------------------------------------
runTime = 0.000157 s   157443 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000076
-----------------------------|jki algorithm done|-----------------------------------------
runTime = 0.000146 s   146217 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000082
-----------------------------|kji algorithm done|-----------------------------------------
B= 64
runTime = 0.000826 s   826173 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000015
-----------------------------|ijk algorithm done|-----------------------------------------
runTime = 0.000799 s   799462 ns number of FloatingPoint Operations = 12    GFLOPS = 0.000015
-----------------------------|jik algorithm done|-----------------------------------------
runTime = 0.001017 s   1016541 ns    number of FloatingPoint Operations = 12    GFLOPS = 0.000012
-----------------------------|kij algorithm done|-----------------------------------------
runTime = 0.002114 s   2114094 ns    number of FloatingPoint Operations = 12    GFLOPS = 0.000006
-----------------------------|ikj algorithm done|-----------------------------------------
runTime = 0.001074 s   1074202 ns    number of FloatingPoint Operations = 12    GFLOPS = 0.000011
-----------------------------|jki algorithm done|-----------------------------------------
runTime = 0.001058 s   1058306 ns    number of FloatingPoint Operations = 12    GFLOPS = 0.000011
-----------------------------|kji algorithm done|-----------------------------------------
```

```
 runTime = 6.069261 s
s  number of FloatingPoint Operations = 2097152   GFLOPS = 0.000346
```

--32*32--

```
-----------------------------|ijk algorithm done no block|------------------------------------------------------------------------
 B= 8
 runTime = 4.226036 s   4226036127 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000496
-----------------------------|ijk algorithm done|-------------------------------------
 runTime = 4.696806 s   4696806497 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000447
-----------------------------|jik algorithm done|-------------------------------------
 runTime = 4.788136 s   4788135504 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000438
-----------------------------|kij algorithm done|-------------------------------------
 runTime = 4.796283 s   4796282739 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000437
-----------------------------|ikj algorithm done|-------------------------------------
 runTime = 6.260349 s   6260348675 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000335
-----------------------------|jki algorithm done|-------------------------------------
 runTime = 6.060332 s   6060332402 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000346
-----------------------------|kji algorithm done|-------------------------------------
 B= 16
 runTime = 4.356163 s   4356162830 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000481
-----------------------------|ijk algorithm done|-------------------------------------
 runTime = 4.529889 s   4529889110 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000463
-----------------------------|jik algorithm done|-------------------------------------
 runTime = 4.600116 s   4600115591 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000456
-----------------------------|kij algorithm done|-------------------------------------
 runTime = 4.591667 s   4591666649 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000457
-----------------------------|ikj algorithm done|-------------------------------------
 runTime = 12.214085 s      12214084799 ns    number of FloatingPoint Operations = 2097152   GFLOPS =
0.000172
-----------------------------|jki algorithm done|-------------------------------------
 runTime = 9.914468 s   9914467557 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000212
-----------------------------|kji algorithm done|-------------------------------------
 B= 32
 runTime = 3.918784 s   3918784136 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000535
-----------------------------|ijk algorithm done|-------------------------------------
 runTime = 4.402054 s   4402054161 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000476
-----------------------------|jik algorithm done|-------------------------------------
 runTime = 4.284045 s   4284044783 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000490
-----------------------------|kij algorithm done|-------------------------------------
 runTime = 4.292766 s   4292766132 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000489
-----------------------------|ikj algorithm done|-------------------------------------
 runTime = 13.636589 s      13636588755 ns    number of FloatingPoint Operations = 2097152   GFLOPS =
0.000154
-----------------------------|jki algorithm done|-------------------------------------
 runTime = 13.650993 s      13650993382 ns    number of FloatingPoint Operations = 2097152   GFLOPS =
0.000154
-----------------------------|kji algorithm done|-------------------------------------
 B= 64
 runTime = 4.870623 s   4870622828 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000431
-----------------------------|ijk algorithm done|-------------------------------------
 runTime = 5.533583 s   5533582612 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000379
-----------------------------|jik algorithm done|-------------------------------------
 runTime = 4.395311 s   4395310725 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000477
```

Final Project

```
------------------------------|kij algorithm done|---------------------------------------
 runTime = 4.354882 s   4354881797 ns number of FloatingPoint Operations = 2097152   GFLOPS = 0.000482
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 12.984173 s      12984173351 ns    number of FloatingPoint Operations = 2097152   GFLOPS =
0.000162
------------------------------|jki algorithm done|---------------------------------------
 runTime = 13.342011 s      13342011077 ns    number of FloatingPoint Operations = 2097152   GFLOPS =
0.000157
------------------------------|kji algorithm done|---------------------------------------
```

---1024*1024--

```
------------------------------|ijk algorithm done no block|---------------------------------------
 B= 8
 runTime = 0.000119 s   119337 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.017161
------------------------------|ijk algorithm done|---------------------------------------
 runTime = 0.000131 s   130936 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015641
------------------------------|jik algorithm done|---------------------------------------
 runTime = 0.000133 s   132542 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015452
------------------------------|kij algorithm done|---------------------------------------
 runTime = 0.000132 s   132202 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015491
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 0.000137 s   136882 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.014962
------------------------------|jki algorithm done|---------------------------------------
 runTime = 0.000137 s   137232 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.014924
------------------------------|kji algorithm done|---------------------------------------
 B= 16
 runTime = 0.000103 s   103241 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.019837
------------------------------|ijk algorithm done|---------------------------------------
 runTime = 0.000127 s   126881 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.016141
------------------------------|jik algorithm done|---------------------------------------
 runTime = 0.000136 s   135773 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015084
------------------------------|kij algorithm done|---------------------------------------
 runTime = 0.000146 s   146283 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.014000
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 0.000131 s   131079 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015624
------------------------------|jki algorithm done|---------------------------------------
 runTime = 0.000131 s   130795 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015658
------------------------------|kji algorithm done|---------------------------------------
 B= 32
 runTime = 0.000099 s   99398 ns  number of FloatingPoint Operations = 2048  GFLOPS = 0.020604
------------------------------|ijk algorithm done|---------------------------------------
 runTime = 0.000106 s   106209 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.019283
------------------------------|jik algorithm done|---------------------------------------
 runTime = 0.000132 s   132283 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015482
------------------------------|kij algorithm done|---------------------------------------
 runTime = 0.000136 s   136457 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015008
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 0.000129 s   128645 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015920
------------------------------|jki algorithm done|---------------------------------------
 runTime = 0.000136 s   135985 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.015060
------------------------------|kji algorithm done|---------------------------------------
 B= 64
 runTime = 0.000865 s   865242 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.002367
------------------------------|ijk algorithm done|---------------------------------------
 runTime = 0.000853 s   853073 ns number of FloatingPoint Operations = 2048  GFLOPS = 0.002401
```

Final Project

```
-----------------------------|ijk algorithm done|--------------------------------------
 runTime = 0.001044 s   1043527 ns    number of FloatingPoint Operations = 2048  GFLOPS = 0.001963
-----------------------------|kij algorithm done|--------------------------------------
 runTime = 0.001033 s   1032793 ns    number of FloatingPoint Operations = 2048  GFLOPS = 0.001983
-----------------------------|ikj algorithm done|--------------------------------------
 runTime = 0.001067 s   1066934 ns    number of FloatingPoint Operations = 2048  GFLOPS = 0.001920
-----------------------------|jki algorithm done|--------------------------------------
 runTime = 0.001069 s   1069183 ns    number of FloatingPoint Operations = 2048  GFLOPS = 0.001915
-----------------------------|kji algorithm done|--------------------------------------
 runTime = 45.535954 s  number of FloatingPoint Operations = 8388608   GFLOPS = 0.000184
```

---2048*2048---

```
-----------------------------|ijk algorithm done no block|--------------------------------------
 B= 8
 runTime = 32.433923 s      32433922697 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000259
-----------------------------|ijk algorithm done|--------------------------------------
 runTime = 36.517633 s      36517632824 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000230
-----------------------------|jik algorithm done|--------------------------------------
 runTime = 36.972438 s      36972437924 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000227
-----------------------------|kij algorithm done|--------------------------------------
 runTime = 38.612746 s      38612746134 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000217
-----------------------------|ikj algorithm done|--------------------------------------
 runTime = 48.140140 s      48140139733 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000174
-----------------------------|jki algorithm done|--------------------------------------
 runTime = 46.687330 s      46687330076 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000180
-----------------------------|kji algorithm done|--------------------------------------
 B= 16
 runTime = 34.165167 s      34165166566 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000246
-----------------------------|ijk algorithm done|--------------------------------------
 runTime = 39.122779 s      39122778619 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000214
-----------------------------|jik algorithm done|--------------------------------------
 runTime = 35.674037 s      35674037496 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000235
-----------------------------|kij algorithm done|--------------------------------------
 runTime = 35.337689 s      35337688791 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000237
-----------------------------|ikj algorithm done|--------------------------------------
 runTime = 97.794302 s      97794302221 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000086
-----------------------------|jki algorithm done|--------------------------------------
 runTime = 80.240548 s      80240548163 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000105
-----------------------------|kji algorithm done|--------------------------------------
 B= 32
 runTime = 31.391975 s      31391975356 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000267
-----------------------------|ijk algorithm done|--------------------------------------
```

```
 runTime = 45.858611 s      45858610681 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000183
------------------------------|jik algorithm done|---------------------------------------
 runTime = 34.187858 s      34187858301 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000245
------------------------------|kij algorithm done|---------------------------------------
 runTime = 35.945998 s      35945997962 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000233
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 112.004129 s     112004129025 ns   number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000075
------------------------------|jki algorithm done|---------------------------------------
 runTime = 114.025109 s     114025109043 ns   number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000074
------------------------------|kji algorithm done|---------------------------------------
 B= 64
 runTime = 48.982161 s      48982161391 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000171
------------------------------|ijk algorithm done|---------------------------------------
 runTime = 50.262453 s      50262453427 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000167
------------------------------|jik algorithm done|---------------------------------------
 runTime = 38.106988 s      38106988343 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000220
------------------------------|kij algorithm done|---------------------------------------
 runTime = 36.821491 s      36821491017 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000228
------------------------------|ikj algorithm done|---------------------------------------
 runTime = 90.307067 s      90307066786 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000093
------------------------------|jki algorithm done|---------------------------------------
 runTime = 89.847946 s      89847946485 ns    number of FloatingPoint Operations = 8388608   GFLOPS = 
0.000093
------------------------------|kji algorithm done|---------------------------------------
```