Task3

1. Compile both OMP and CUDA versions of your selected benchmarks. Do you need to make any changes in Makefile?

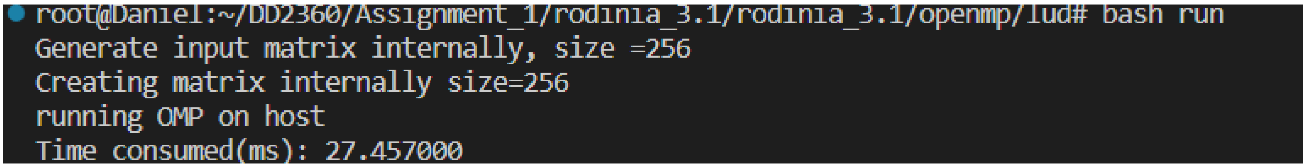
   We need to modify the rodinia_3.1/cuda/lud/cuda/Makefile if we want to run the lud.

   ```
   remove: -arch=sm_13 \
   in  NVCCFLAGS +=  section
   ```

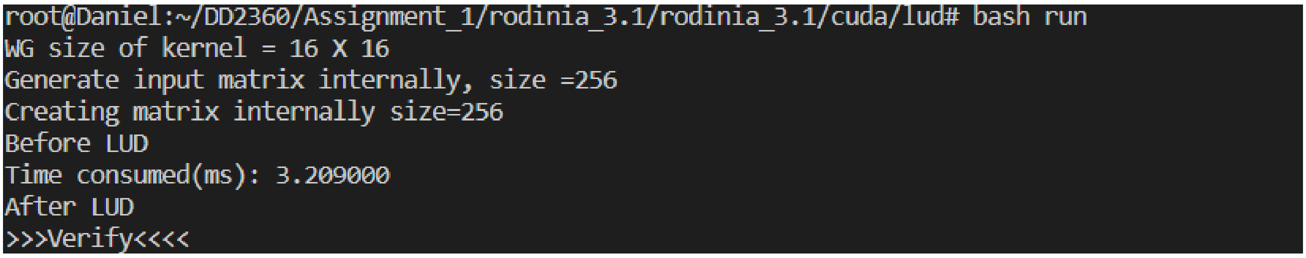
   For rodinia_3.1/cuda/b+tree/Makefile,

   ```
   remove line CUDA_FLAG = -arch sm_20 and all variables related to CUDA_FLAG
   ```

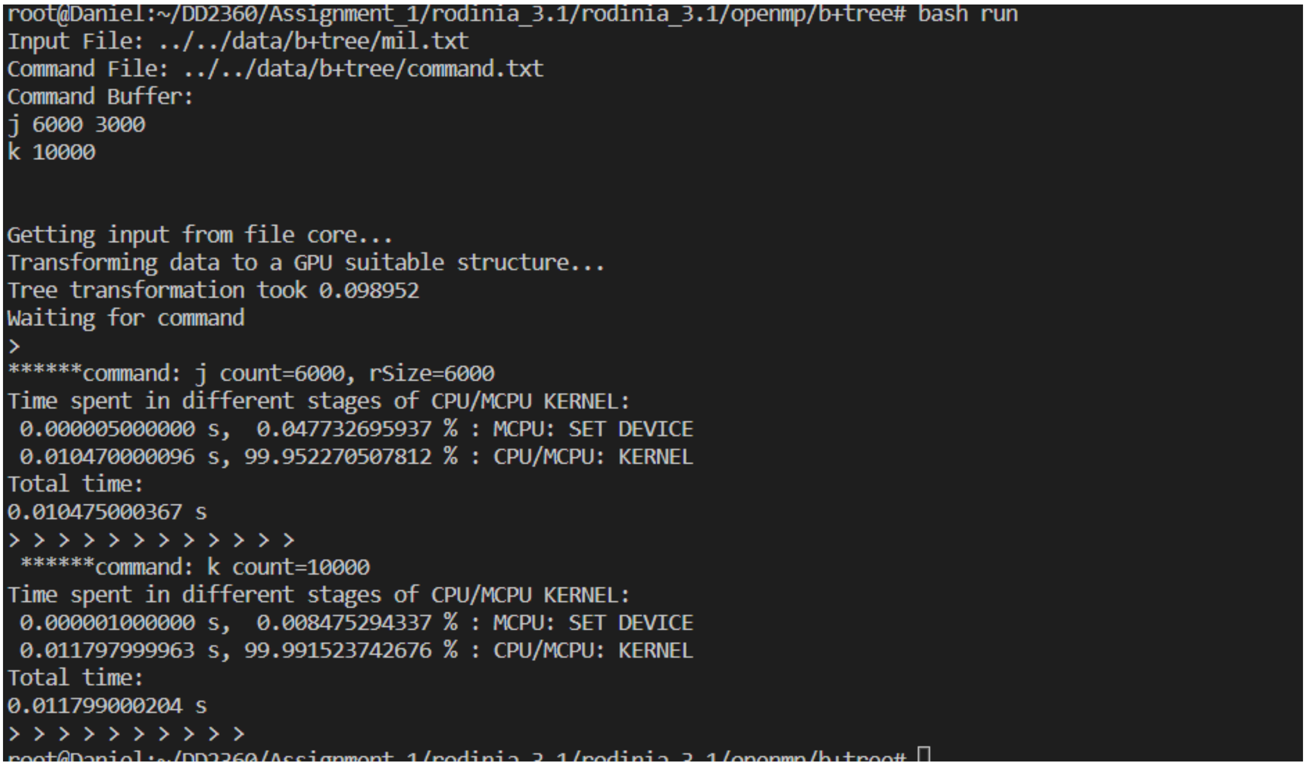
2. Ensure the same input problem is used for OMP and CUDA versions. Report and compare their execution time.



```
● root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/lud# bash run
  Generate input matrix internally, size =256
  Creating matrix internally size=256
  running OMP on host
  Time consumed(ms): 27.457000
```



```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/cuda/lud# bash run
WG size of kernel = 16 X 16
Generate input matrix internally, size =256
Creating matrix internally size=256
Before LUD
Time consumed(ms): 3.209000
After LUD
>>>Verify<<<<
```



```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/b+tree# bash run
Input File: ../../data/b+tree/mil.txt
Command File: ../../data/b+tree/command.txt
Command Buffer:
j 6000 3000
k 10000


Getting input from file core...
Transforming data to a GPU suitable structure...
Tree transformation took 0.098952
Waiting for command
>
******command: j count=6000, rSize=6000
Time spent in different stages of CPU/MCPU KERNEL:
 0.000005000000 s,  0.047732695937 % : MCPU: SET DEVICE
 0.010470000096 s, 99.952270507812 % : CPU/MCPU: KERNEL
Total time:
0.010475000367 s
> > > > > > > > > > > > >
 ******command: k count=10000
Time spent in different stages of CPU/MCPU KERNEL:
 0.000001000000 s,  0.008475294337 % : MCPU: SET DEVICE
 0.011797999963 s, 99.991523742676 % : CPU/MCPU: KERNEL
Total time:
0.011799000204 s
> > > > > > > > > >
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/openmp/b+tree# 
```

```
root@Daniel:~/DD2360/Assignment_1/rodinia_3.1/rodinia_3.1/cuda/b+tree# bash run
WG size of kernel 1 & 2  = 256
Selecting device 0
Input File: ../../data/b+tree/mil.txt
Command File: ../../data/b+tree/command.txt
Command Buffer:
j 6000 3000
k 10000
Getting input from file ../../data/b+tree/mil.txt...
Transforming data to a GPU suitable structure...
Tree transformation took 0.403506
Waiting for command
>
******command: j count=6000, rSize=6000
knodes_elem=7874, knodes_unit_mem=2068, knodes_mem=16283432
# of blocks = 6000, # of threads/block = 256 (ensure that device can handle)
Time spent in different stages of GPU_CUDA KERNEL:
 0.864696025848 s, 98.529289245605 % : GPU: SET DEVICE / DRIVER INIT
 0.002187999897 s,  0.249315470457 % : GPU MEM: ALO
 0.008689999580 s,  0.990197181702 % : GPU MEM: COPY IN
 0.000292999990 s,  0.033386394382 % : GPU: KERNEL
 0.000315000012 s,  0.035893220454 % : GPU MEM: COPY OUT
 0.001420999994 s,  0.161918312311 % : GPU MEM: FRE
Total time:
0.877602994442 s
> > > > > > > > > > > >
 ******command: k count=10000
records_elem=1000000, records_unit_mem=4, records_mem=4000000
knodes_elem=7874, knodes_unit_mem=2068, knodes_mem=16283432
# of blocks = 10000, # of threads/block = 256 (ensure that device can handle)
Time spent in different stages of GPU_CUDA KERNEL:
 0.000015000000 s,  0.278035223484 % : GPU: SET DEVICE / DRIVER INIT
 0.001618999988 s, 30.009265899658 % : GPU MEM: ALO
 0.002870999975 s, 53.215938568115 % : GPU MEM: COPY IN
 0.000209000005 s,  3.873957395554 % : GPU: KERNEL
 0.000032000000 s,  0.593141794205 % : GPU MEM: COPY OUT
 0.000648999994 s, 12.029656410217 % : GPU MEM: FRE
Total time:
0.005394999869 s
> > > > > > > > > >
```

3. Do you observe expected speedup on GPU compared to CPU? Why or Why not?

There's speedup for lud, 27.45ms is reduced to 3.21ms when running on gpu.

For b+tree, 0.877602994442s for gpu and 0.017s for cpu.

Therefore, not all algorithms could benefit from parallelism.