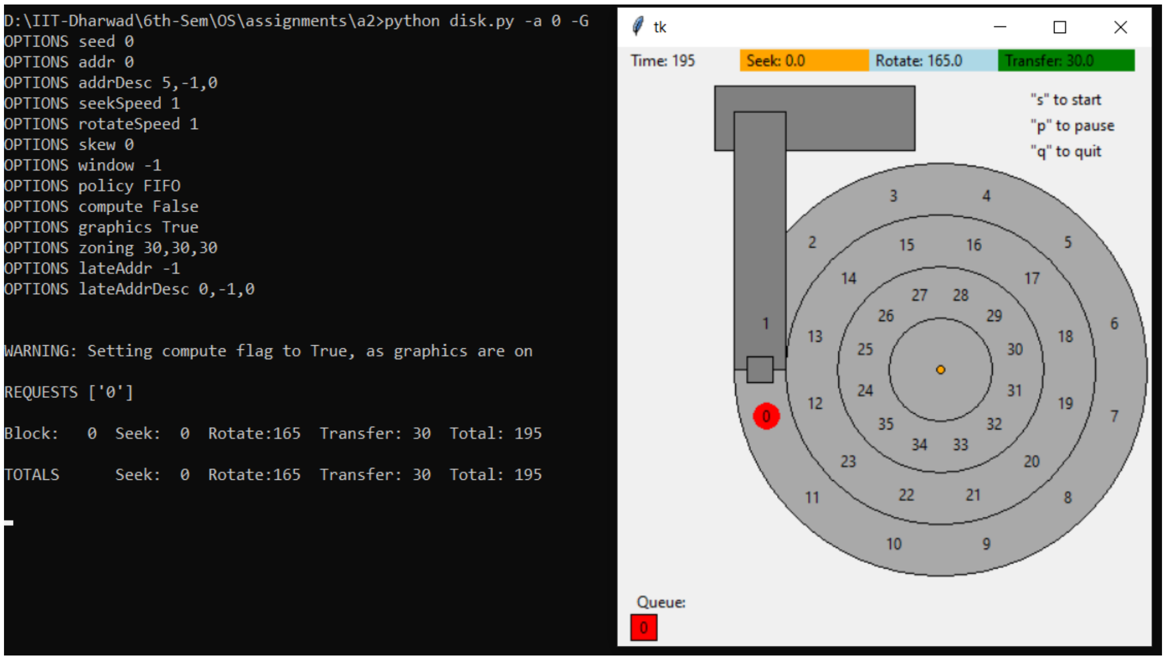
CS304 Assignment 2

- Atharva Swami (190010008)

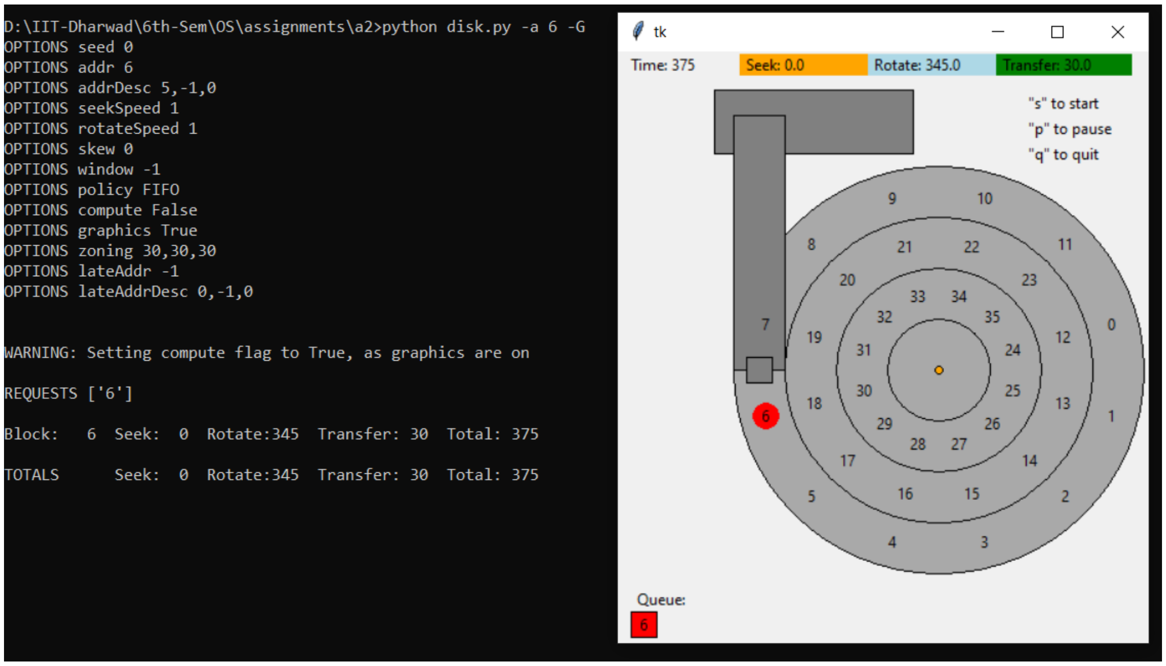
disk.py

(1) `python disk.py -a 0 -G`



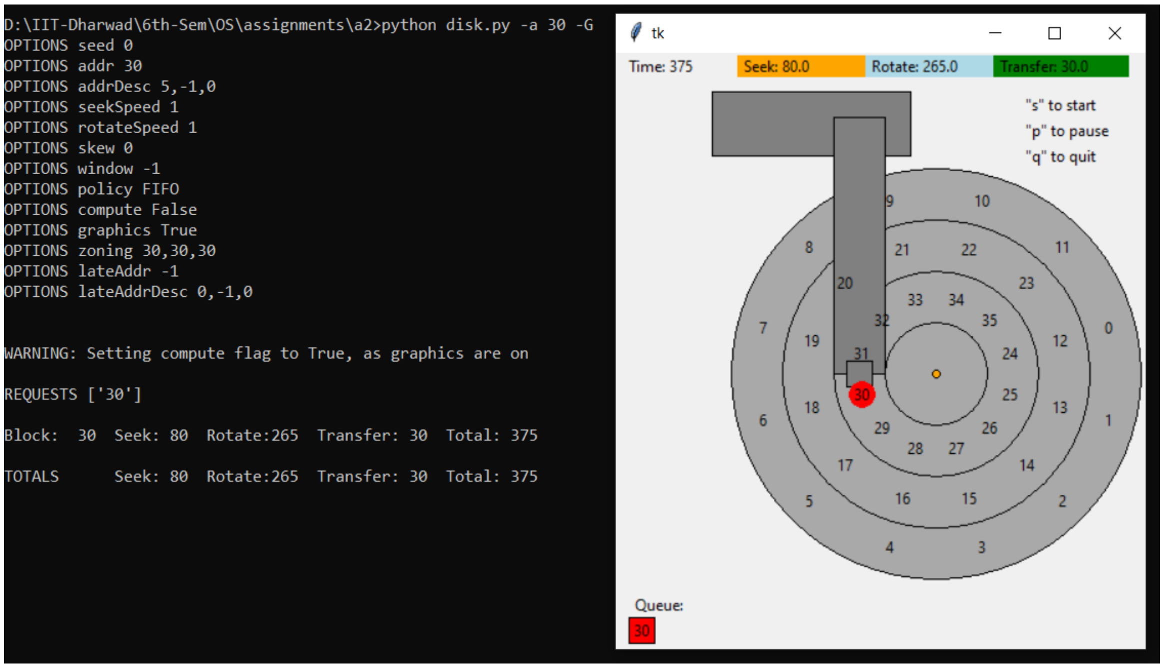
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 0 -G
OPTIONS seed 0
OPTIONS addr 0
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['0']

Block:   0  Seek:  0  Rotate:165  Transfer: 30  Total: 195

TOTALS      Seek:  0  Rotate:165  Transfer: 30  Total: 195
```

`python disk.py -a 6 -G`



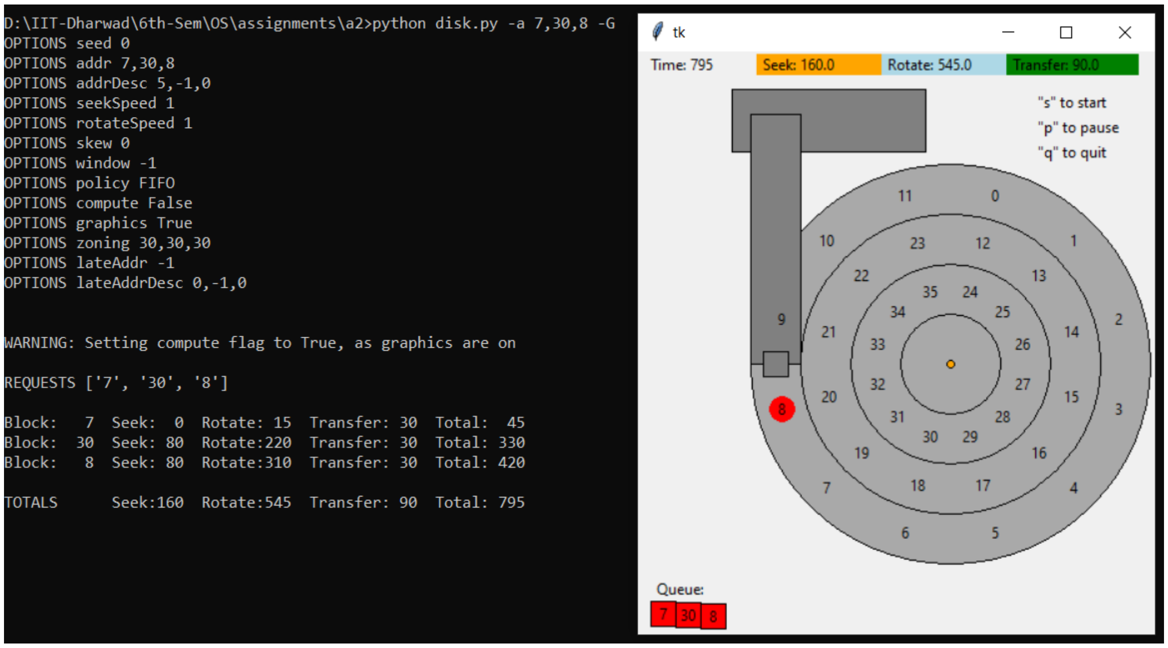
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 6 -G
OPTIONS seed 0
OPTIONS addr 6
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['6']

Block:   6  Seek:  0  Rotate:345  Transfer: 30  Total: 375

TOTALS      Seek:  0  Rotate:345  Transfer: 30  Total: 375
```

python disk.py -a 30 -G



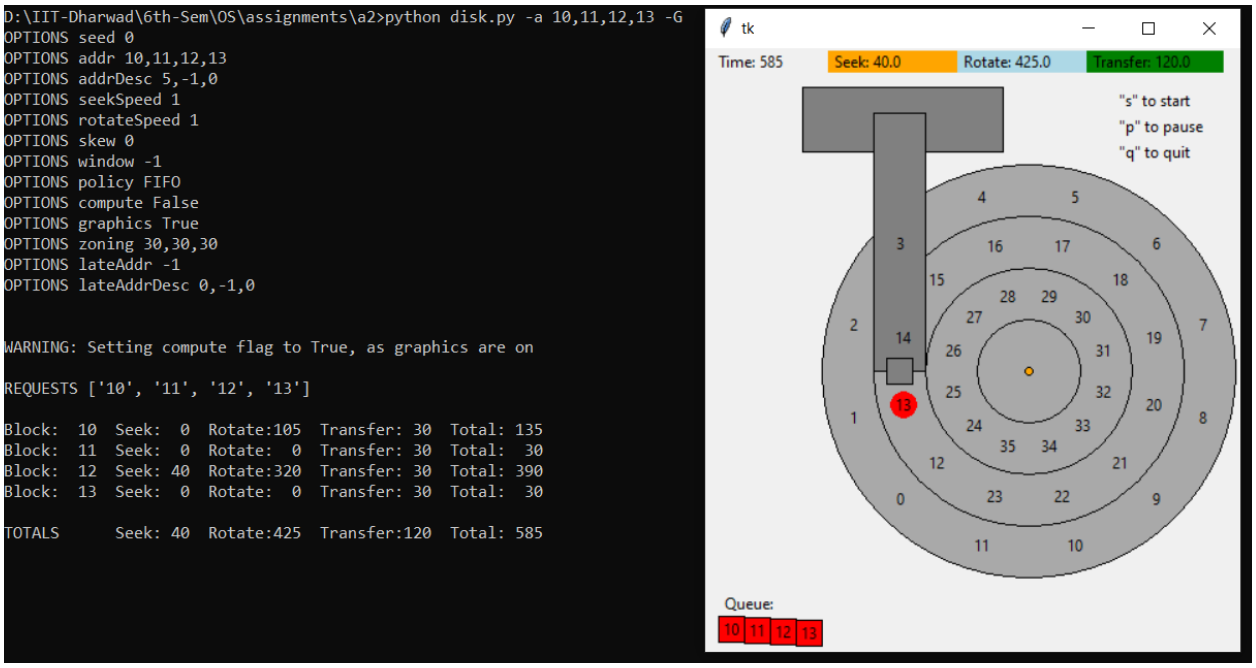
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 30 -G
OPTIONS seed 0
OPTIONS addr 30
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['30']

Block:  30  Seek: 80  Rotate:265  Transfer: 30  Total: 375

TOTALS      Seek: 80  Rotate:265  Transfer: 30  Total: 375
```

python disk.py -a 7,30,8 -G



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795
```

python disk.py -a 10,11,12,13 -G



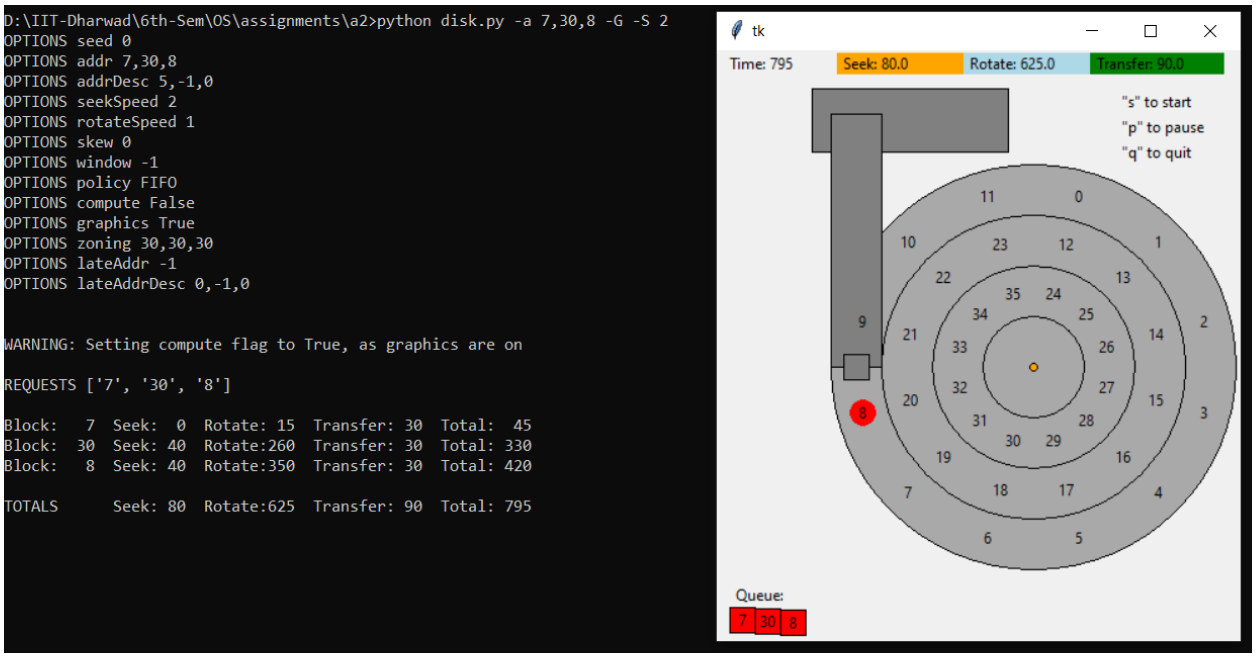
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 10,11,12,13 -G
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate:320  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:425  Transfer:120  Total: 585
```

(2) Here ‘S’ flag is used to change the seek speed which has a default value of 1. One issue here is that the rotation keeps going on while seeking is being done. That causes unnecessary rotations sometimes. A higher seek rate helps only when it prevents such unnecessary rotation. Lower seek rate causes more of these unnecessary rotations, and thus decreases overall performance. If the rotation stopped during seeking, seek rate would have had a direct impact on performance.

For -a 0, -a 6 the output will be the same as in question 1 but for the other 3 requests the seek times will change.

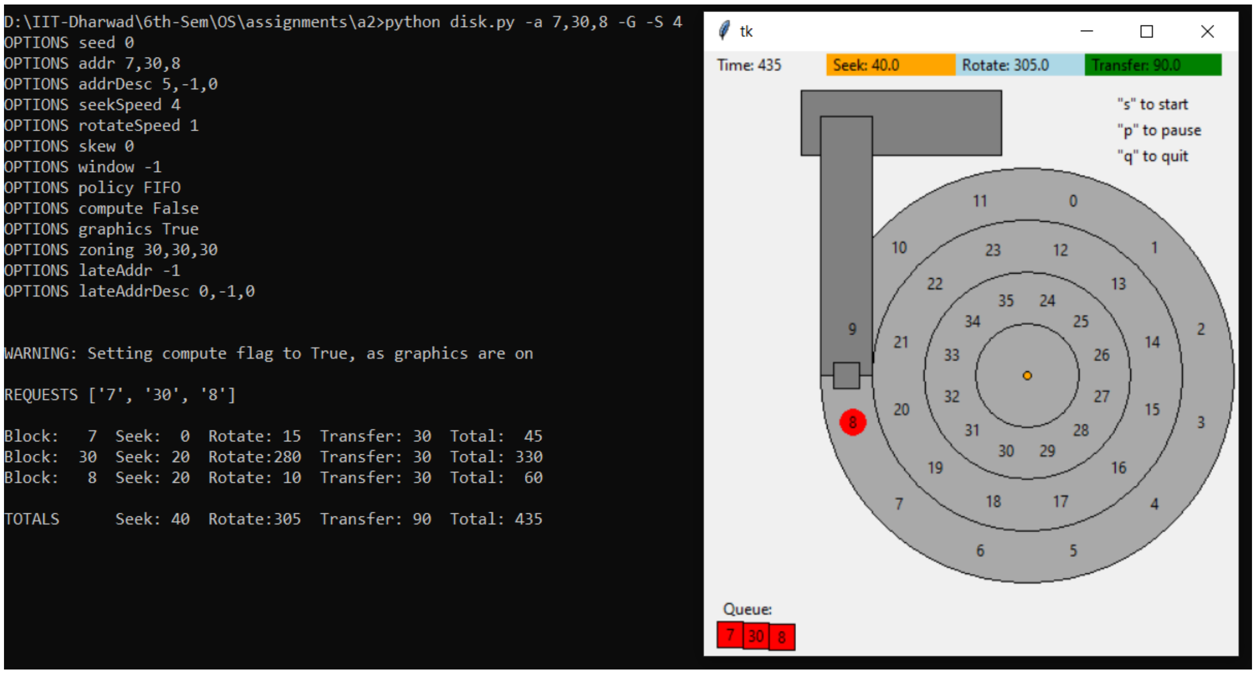
For -a 30, -a 7,30,8, and -a 10,11,12,13: Only seek, rotation times are the only ones changing; whereas the transfer time remains the same in all cases. The seek times have decreased as we increase the seek speed and if we decrease the speed the seek time is increased. The changes in the rotation time are due to the changed position of the heads when seek is done to the track because the platter will be rotating even though we are seeking and that might cause unnecessary rotations.

python disk.py -a 7,30,8 -G -S 2



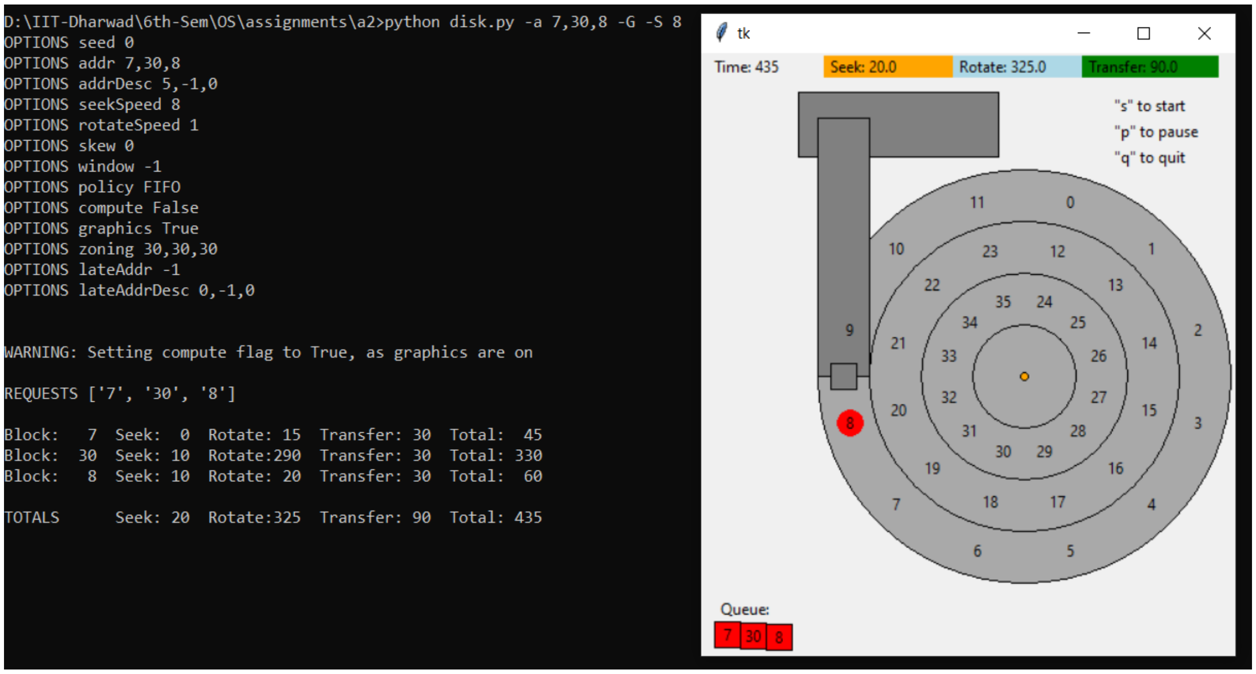
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 2
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 2
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 40  Rotate:260  Transfer: 30  Total: 330
Block:   8  Seek: 40  Rotate:350  Transfer: 30  Total: 420

TOTALS      Seek: 80  Rotate:625  Transfer: 90  Total: 795
```

python disk.py -a 7,30,8 -G -S 4



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 4
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 4
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 20  Rotate:280  Transfer: 30  Total: 330
Block:   8  Seek: 20  Rotate: 10  Transfer: 30  Total:  60

TOTALS      Seek: 40  Rotate:305  Transfer: 90  Total: 435
```

python disk.py -a 7,30,8 -G -S 8



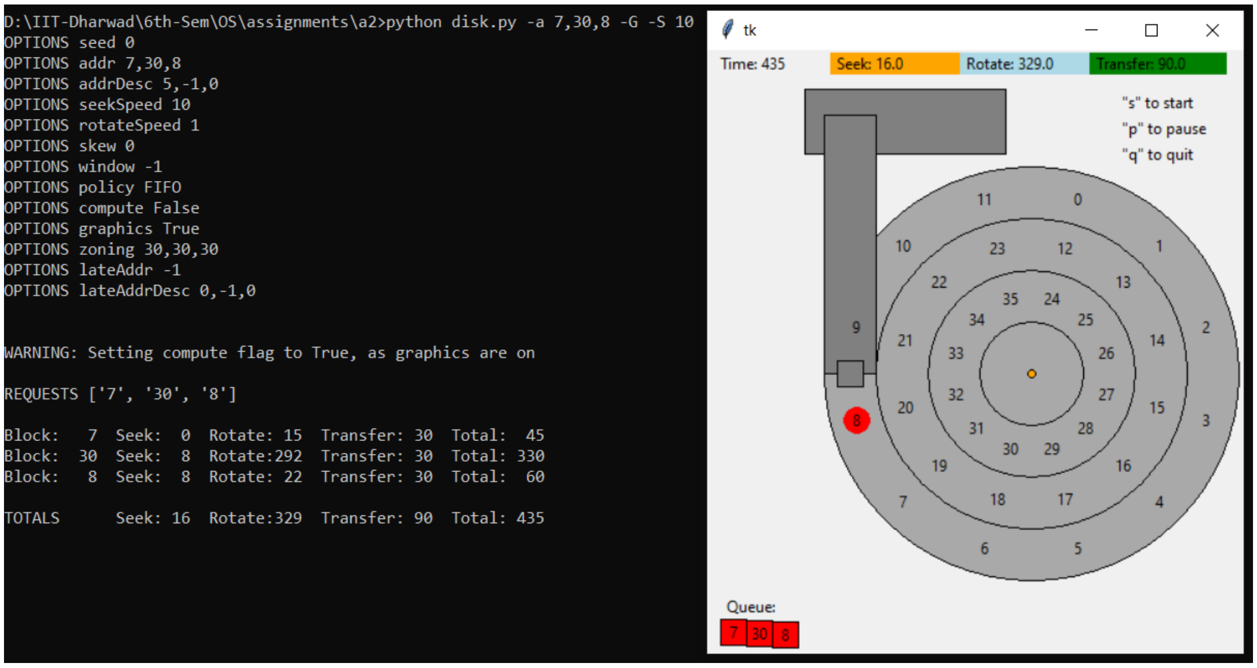
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 8
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 8
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 10  Rotate:290  Transfer: 30  Total: 330
Block:   8  Seek: 10  Rotate: 20  Transfer: 30  Total:  60

TOTALS      Seek: 20  Rotate:325  Transfer: 90  Total: 435
```

Time: 435 Seek: 20.0 Rotate: 325.0 Transfer: 90.0

"s" to start
"p" to pause
"q" to quit

Queue: 7 30 8

python disk.py -a 7,30,8 -G -S 10



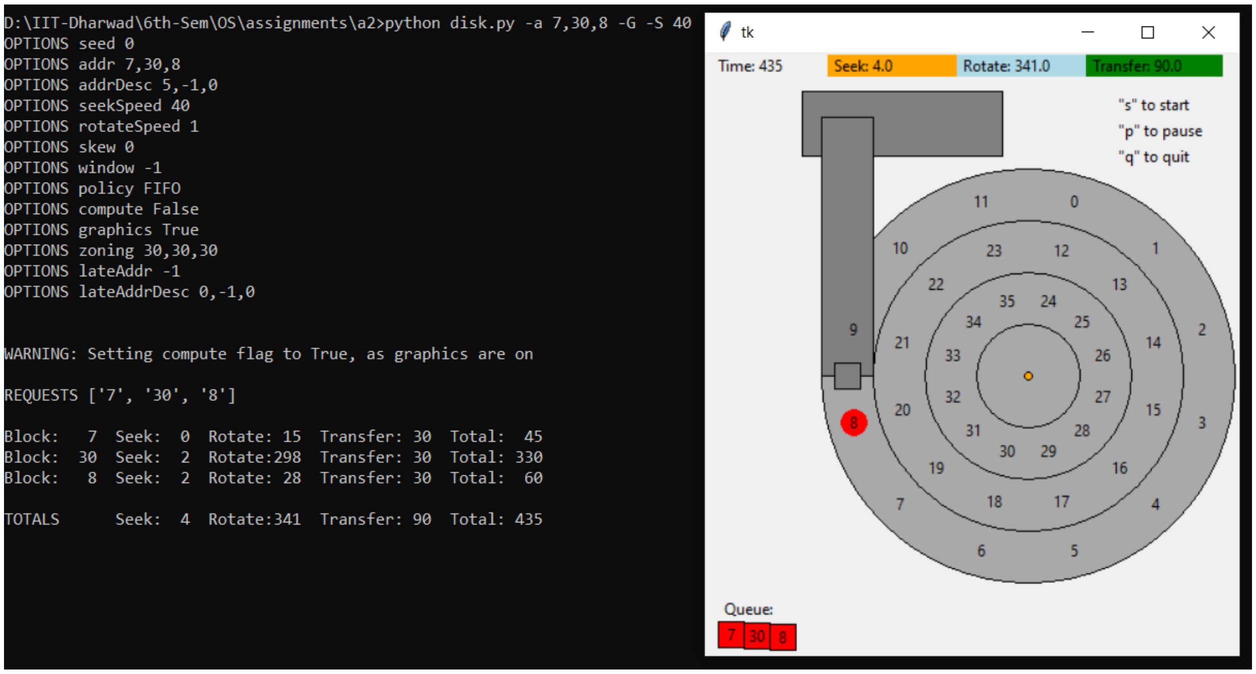
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 10
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 10
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek:  8  Rotate:292  Transfer: 30  Total: 330
Block:   8  Seek:  8  Rotate: 22  Transfer: 30  Total:  60

TOTALS      Seek: 16  Rotate:329  Transfer: 90  Total: 435
```

Time: 435 Seek: 16.0 Rotate: 329.0 Transfer: 90.0

"s" to start
"p" to pause
"q" to quit

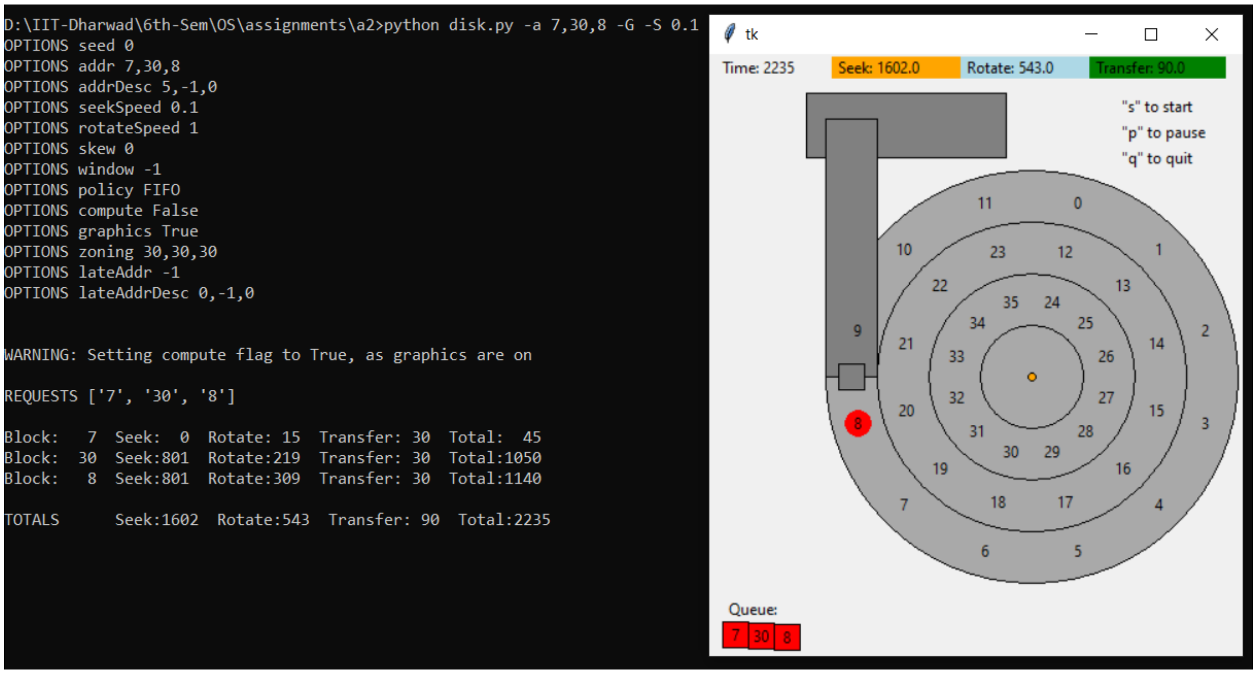
Queue: 7 30 8

python disk.py -a 7,30,8 -G -S 40



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 40
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 40
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek:  2  Rotate:298  Transfer: 30  Total: 330
Block:   8  Seek:  2  Rotate: 28  Transfer: 30  Total:  60

TOTALS      Seek:  4  Rotate:341  Transfer: 90  Total: 435
```

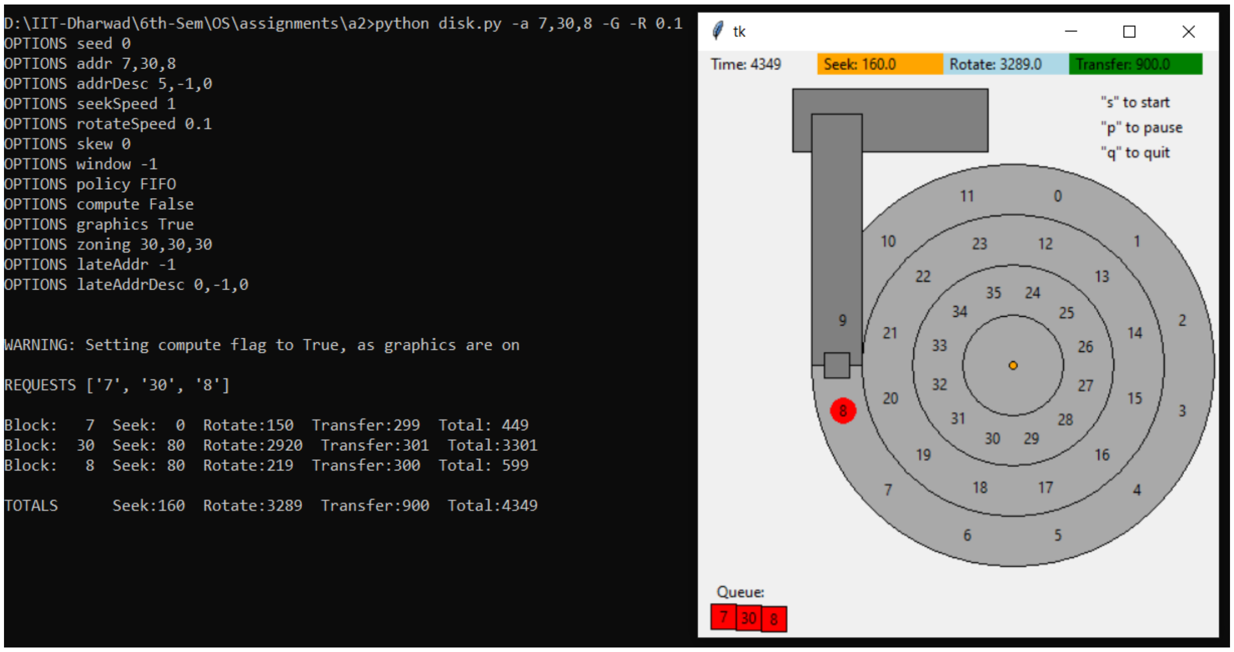
python disk.py -a 7,30,8 -G -S 0.1



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -S 0.1
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 0.1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek:801  Rotate:219  Transfer: 30  Total:1050
Block:   8  Seek:801  Rotate:309  Transfer: 30  Total:1140

TOTALS      Seek:1602  Rotate:543  Transfer: 90  Total:2235
```

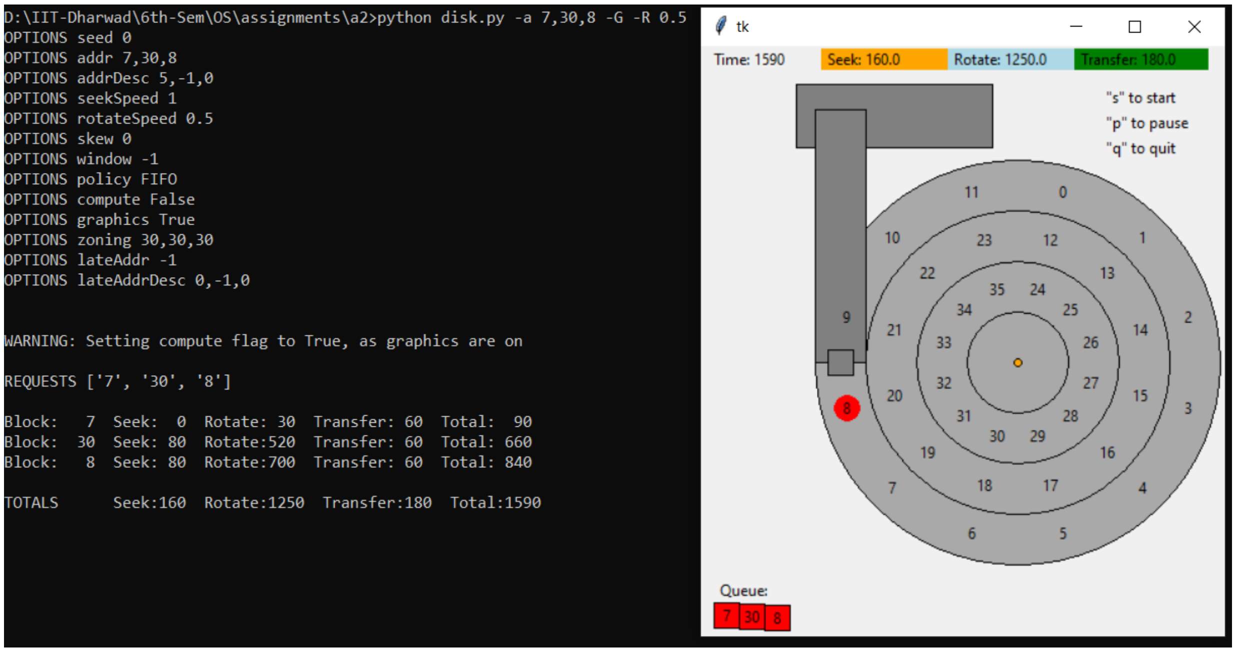
(3) The ‘R’ flag is used to change the rotation speed which is 1 by default. Rotate time and transfer time are longer. Rotation rate always has a direct relation to performance. But sometimes fast rotations can cause extra rotation and decrease performance.

`python disk.py -a 7,30,8 -G -R 0.1`



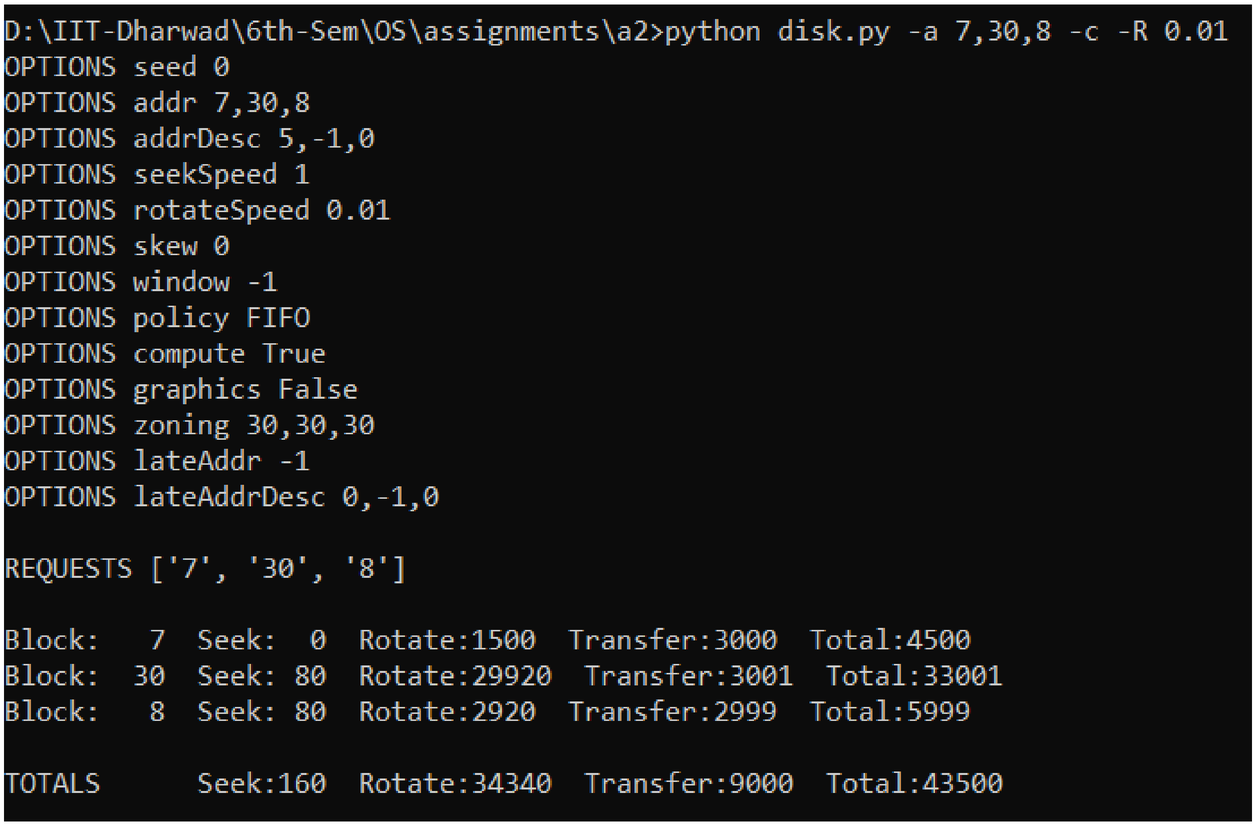
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -R 0.1
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 0.1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate:150  Transfer:299  Total: 449
Block:  30  Seek: 80  Rotate:2920  Transfer:301  Total:3301
Block:   8  Seek: 80  Rotate:219  Transfer:300  Total: 599

TOTALS      Seek:160  Rotate:3289  Transfer:900  Total:4349
```

`python disk.py -a 7,30,8 -G -R 0.5`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -R 0.5
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 0.5
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 30  Transfer: 60  Total:  90
Block:  30  Seek: 80  Rotate:520  Transfer: 60  Total: 660
Block:   8  Seek: 80  Rotate:700  Transfer: 60  Total: 840

TOTALS      Seek:160  Rotate:1250  Transfer:180  Total:1590
```

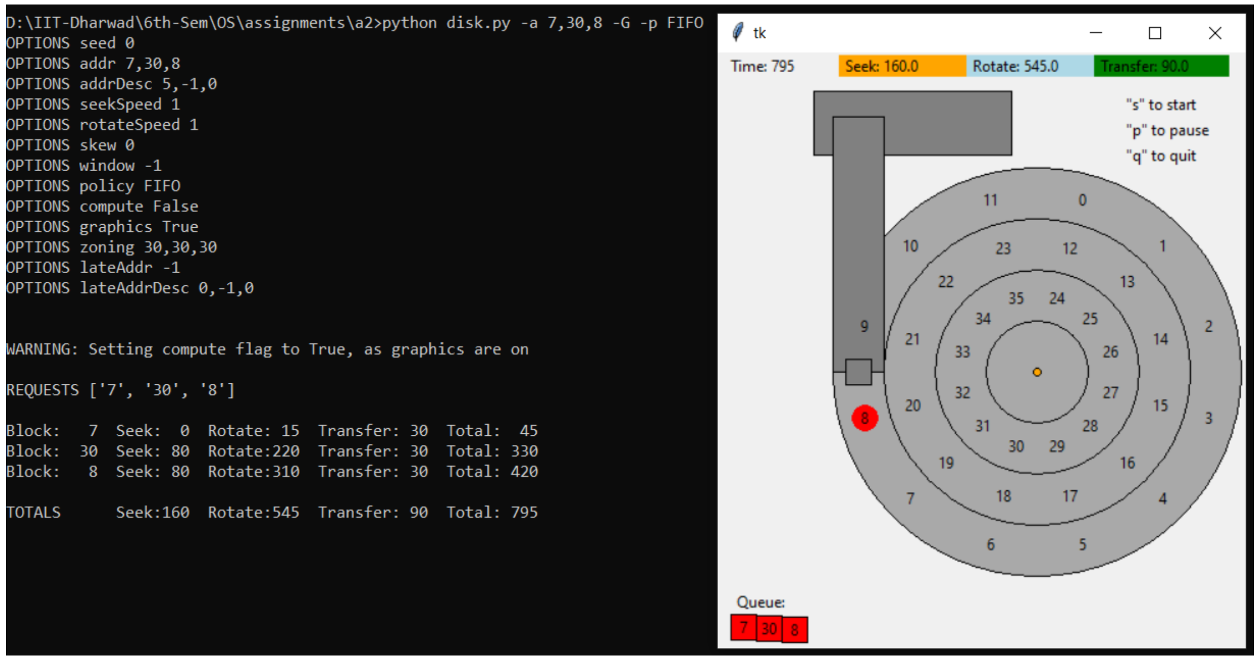
python disk.py -a 7,30,8 -c -R 0.01



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -c -R 0.01
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 0.01
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate:1500  Transfer:3000  Total:4500
Block:  30  Seek: 80  Rotate:29920  Transfer:3001  Total:33001
Block:   8  Seek: 80  Rotate:2920  Transfer:2999  Total:5999

TOTALS      Seek:160  Rotate:34340  Transfer:9000  Total:43500
```

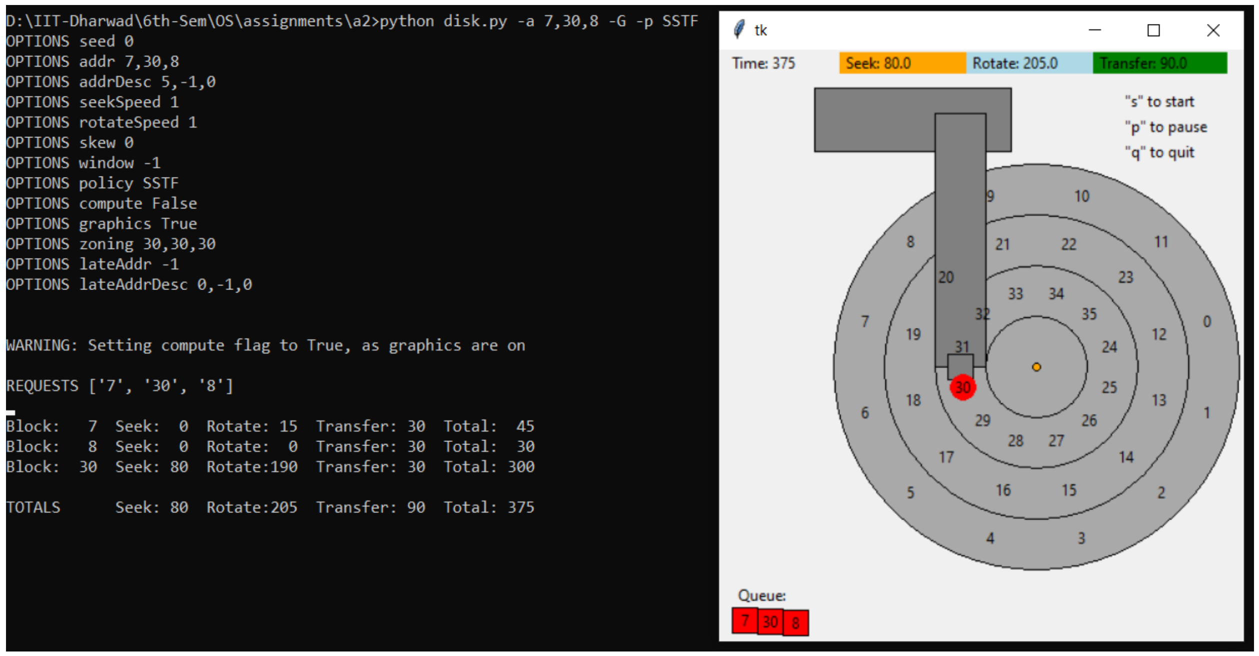
(4) FIFO order: 7, 30, 8
SSTF order: 7, 8, 30
FIFO time: 795
SSTF time: 375
From the screenshots below, we can see that the seek time is the same as in Q1 but the transfer time and rotation time are lesser that makes the total access time lesser. So, we can tell that SSTF performs better than FIFO.

`python disk.py -a 7,30,8 -G -p FIFO`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -p FIFO
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  30  Seek: 80  Rotate:220  Transfer: 30  Total: 330
Block:   8  Seek: 80  Rotate:310  Transfer: 30  Total: 420

TOTALS      Seek:160  Rotate:545  Transfer: 90  Total: 795
```

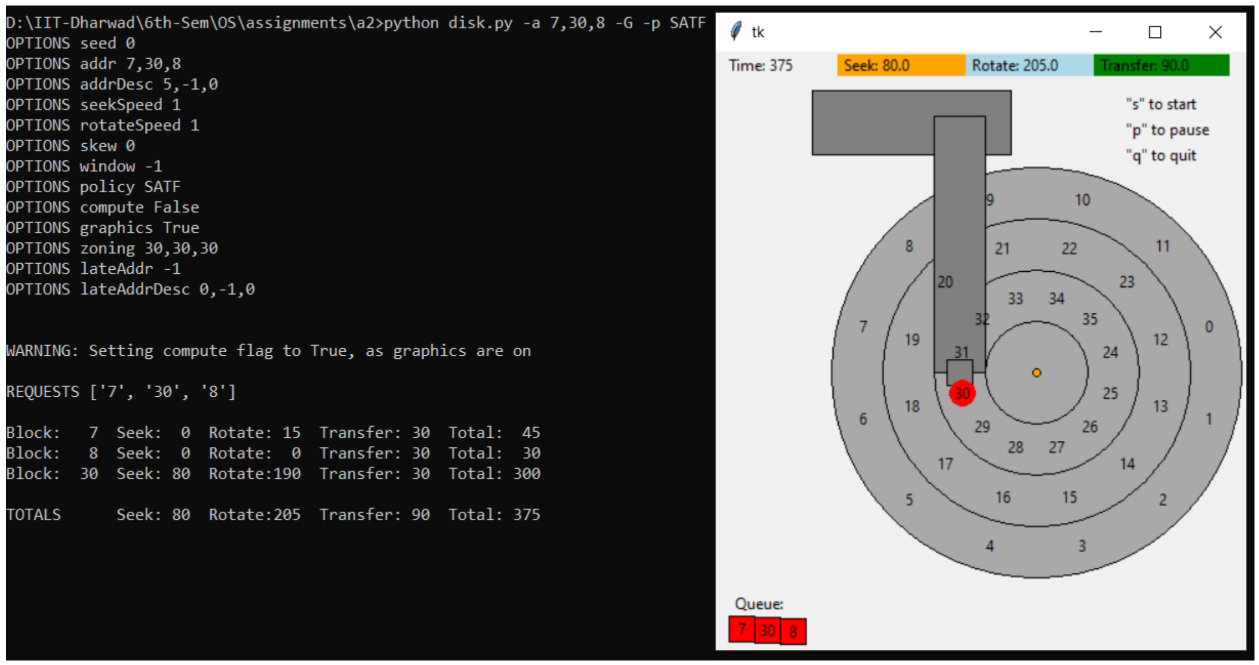
`python disk.py -a 7,30,8 -G -p SSTF`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -p SSTF
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SSTF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

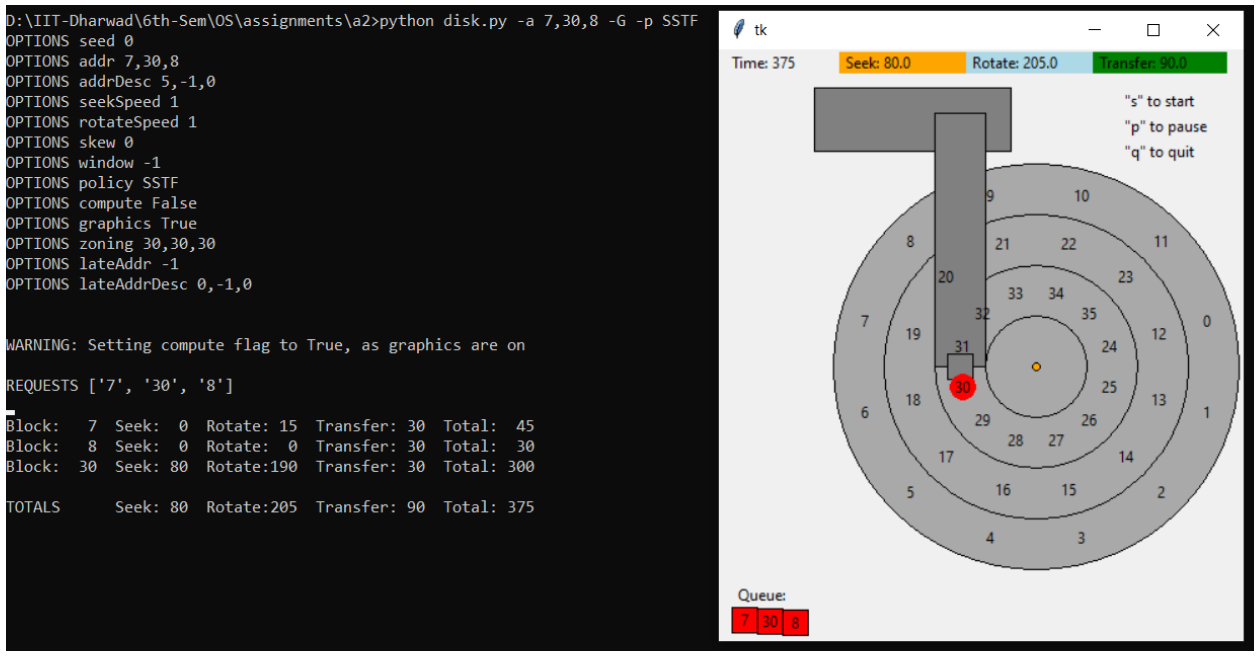
(5) From the screenshot below, we can see that the workload request -a 7,30,8 has the same output for both the schedulers - SSTF and SATF. There is no change in the outputs here because in SATF, after reading 7 the algorithm chooses the next position closest to the disk head i.e. it reads 8 and then reads 30. So, the outputs will be the same as in the SSTF case.

`python disk.py -a 7,30,8 -G -p SATF`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -p SATF
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SATF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

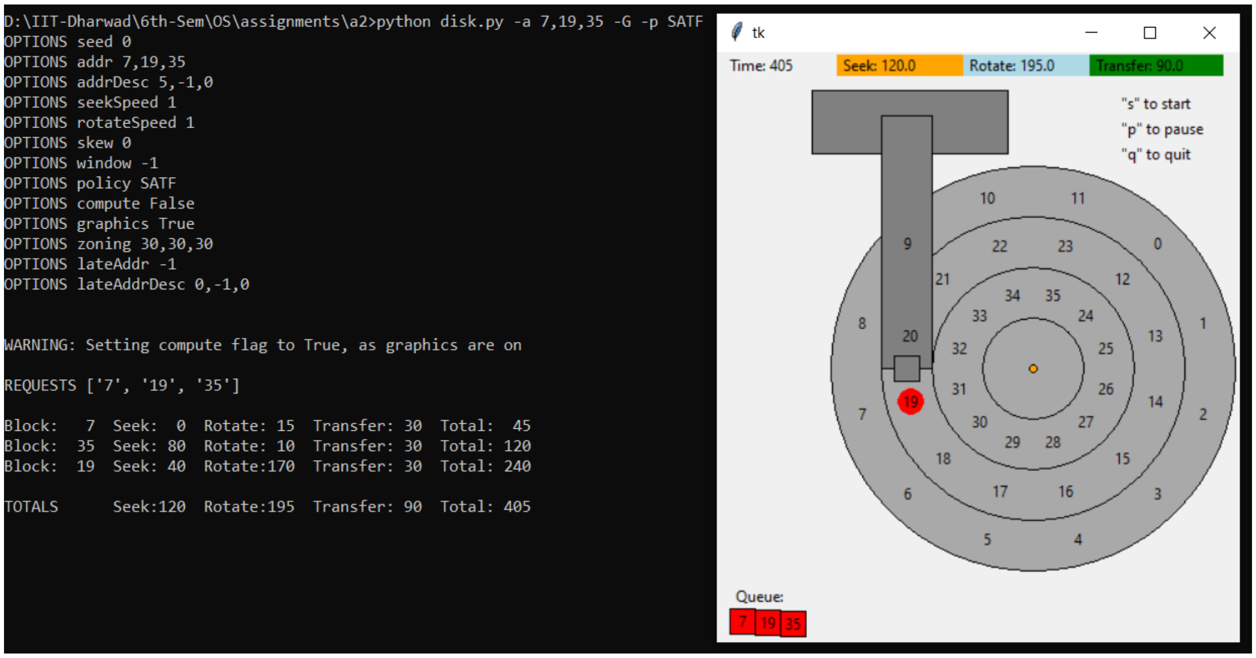
`python disk.py -a 7,30,8 -G -p SSTF`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,30,8 -G -p SSTF
OPTIONS seed 0
OPTIONS addr 7,30,8
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SSTF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '30', '8']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  30  Seek: 80  Rotate:190  Transfer: 30  Total: 300

TOTALS      Seek: 80  Rotate:205  Transfer: 90  Total: 375
```

python disk.py -a 7,19,35 -G -p SATF is faster than **python disk.py -a 7,19,35 -G -p SSTF**. In general, when there is a request to a sector that is close to seek, but requires more rotation, SSTF is slow.

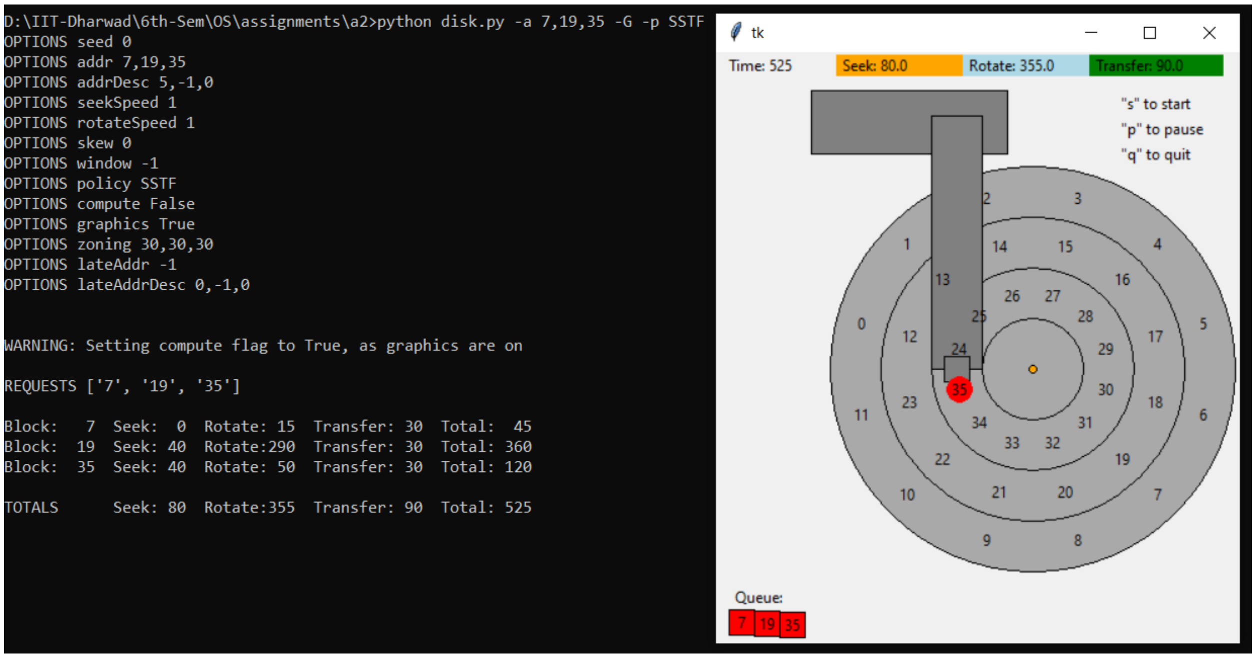
python disk.py -a 7,19,35 -G -p SATF



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,19,35 -G -p SATF
OPTIONS seed 0
OPTIONS addr 7,19,35
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SATF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '19', '35']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  35  Seek: 80  Rotate: 10  Transfer: 30  Total: 120
Block:  19  Seek: 40  Rotate:170  Transfer: 30  Total: 240

TOTALS      Seek:120  Rotate:195  Transfer: 90  Total: 405
```

```
Time: 405   Seek: 120.0   Rotate: 195.0   Transfer: 90.0
```

python disk.py -a 7,19,35 -G -p SSTF



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 7,19,35 -G -p SSTF
OPTIONS seed 0
OPTIONS addr 7,19,35
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SSTF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['7', '19', '35']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:  19  Seek: 40  Rotate:290  Transfer: 30  Total: 360
Block:  35  Seek: 40  Rotate: 50  Transfer: 30  Total: 120

TOTALS      Seek: 80  Rotate:355  Transfer: 90  Total: 525
```

```
Time: 525   Seek: 80.0   Rotate: 355.0   Transfer: 90.0
```

(6) When we try to run the command **`python disk.py -a 10,11,12,13 -G`**, we see that the blocks are consecutive after 11, and we need to seek to the middle track to access 12,13 but due to arrangement of the sectors the head just misses the blocks and so, we have to make an extra rotation to reach them again which is the reason for its poor performance. Generally we can control this disorientation via skew offset. Skewing is the rearrangement of sectors on a disc, so that by the time the computer has read and processed one sector, the next will be in the right position for the disc controller to read. Otherwise the controller has to wait for the disc to make a full revolution before the right sector appears. For the given workload request skew of 2 will maximize the performance.

More generally from the definition of skew we can deduce that skew is influenced by the track width, seek speed, rotation speed and the number of sectors. Considering all these factors the formula to calculate slew can be written as:

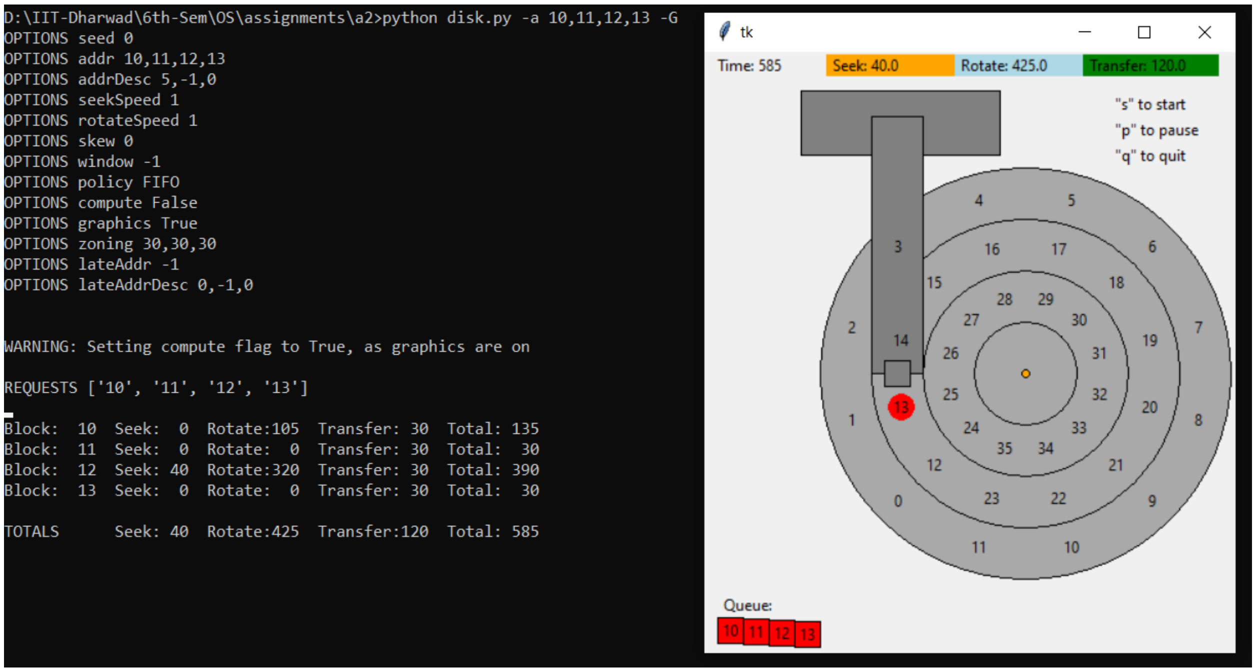
$$\begin{aligned} skew &= math.ceil(track\ width\ /\ seek\ speed)\ *\ (rotation\ speed\ /\ rotational\ space\ degrees\ *\ (360\ /\ 12)) \\ &= math.ceil((40\ /\ 1)\ *\ 1\ /\ 30) \\ &= 2 \end{aligned}$$

For different seek rates:

$- S\,2 \;=>\; skew = math.ceil((40\ /\ 2)\ *\ 1\ /\ 30) = 1$

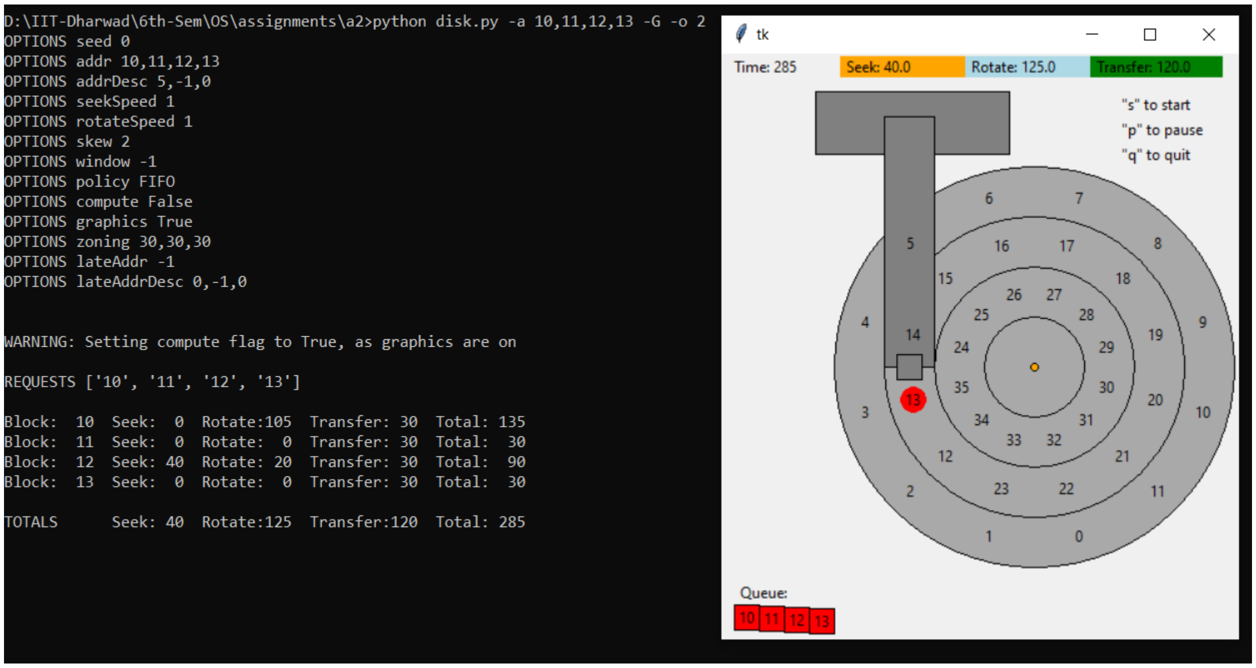
$- S\,4 \;=>\; skew = math.ceil((40\ /\ 4)\ *\ 1\ /\ 30) = 1$

python disk.py -a 10,11,12,13 -G



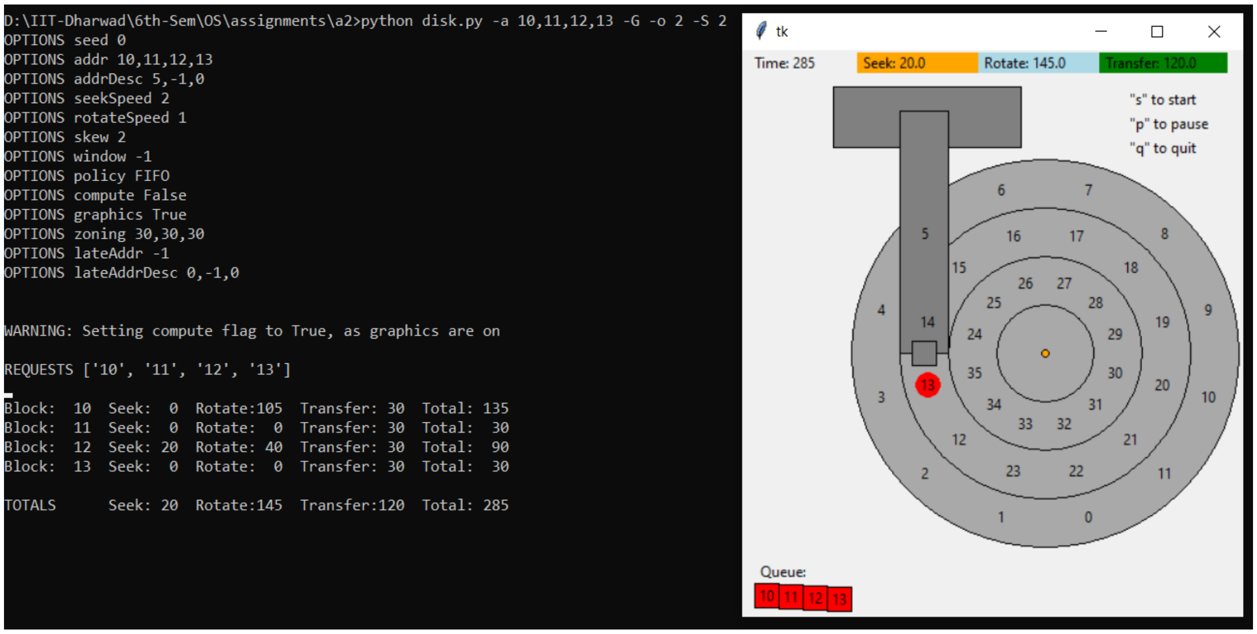
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 10,11,12,13 -G
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate:320  Transfer: 30  Total: 390
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:425  Transfer:120  Total: 585
```

python disk.py -a 10,11,12,13 -G -o 2



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 10,11,12,13 -G -o 2
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 2
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate: 20  Transfer: 30  Total:  90
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 40  Rotate:125  Transfer:120  Total: 285
```

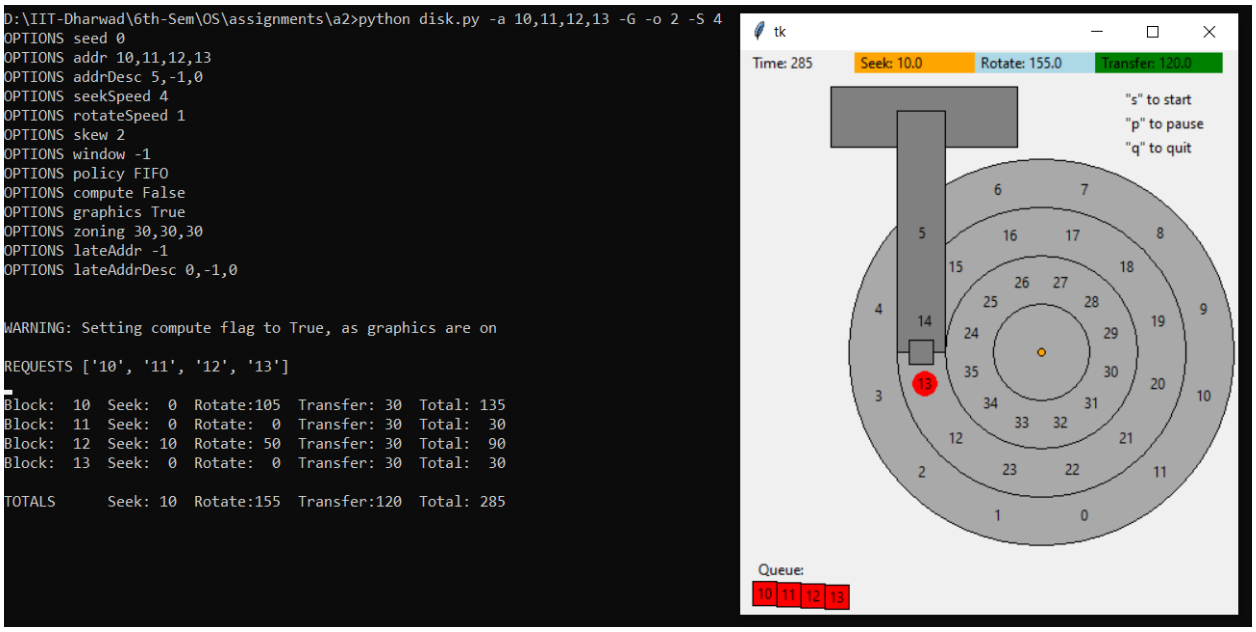
python disk.py -a 10,11,12,13 -G -o 2 -S 2



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 10,11,12,13 -G -o 2 -S 2
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 2
OPTIONS rotateSpeed 1
OPTIONS skew 2
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 20  Rotate: 40  Transfer: 30  Total:  90
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 20  Rotate:145  Transfer:120  Total: 285
```

```
Time: 285   Seek: 20.0   Rotate: 145.0   Transfer: 120.0
"s" to start
"p" to pause
"q" to quit
Queue:
10 11 12 13
```

python disk.py -a 10,11,12,13 -G -o 2 -S 4

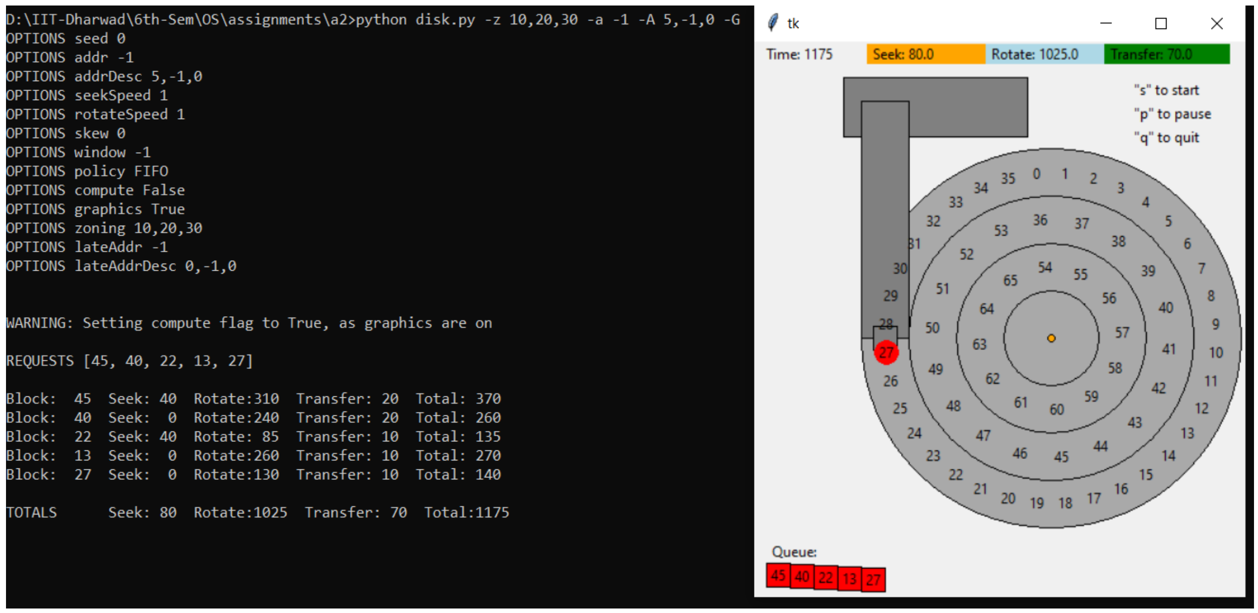


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 10,11,12,13 -G -o 2 -S 4
OPTIONS seed 0
OPTIONS addr 10,11,12,13
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 4
OPTIONS rotateSpeed 1
OPTIONS skew 2
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['10', '11', '12', '13']

Block:  10  Seek:  0  Rotate:105  Transfer: 30  Total: 135
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 10  Rotate: 50  Transfer: 30  Total:  90
Block:  13  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 10  Rotate:155  Transfer:120  Total: 285
```

```
Time: 285   Seek: 10.0   Rotate: 155.0   Transfer: 120.0
"s" to start
"p" to pause
"q" to quit
Queue:
10 11 12 13
```

(7) After running the command **`python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G`** the workload requests generated for seed 0 are 45,40,22,13,27. From the disk figure below we can see that all the blocks requested are in the outer and middle regions and none are in the inner regions. As we know Bandwidth is the total number of bytes transferred directly between the first request for service and the completion of the last transfer. Using this definition we can see the outer blocks are 3 and the inner blocks are 2. We are using sectors since the answer is asked in sectors per unit time.

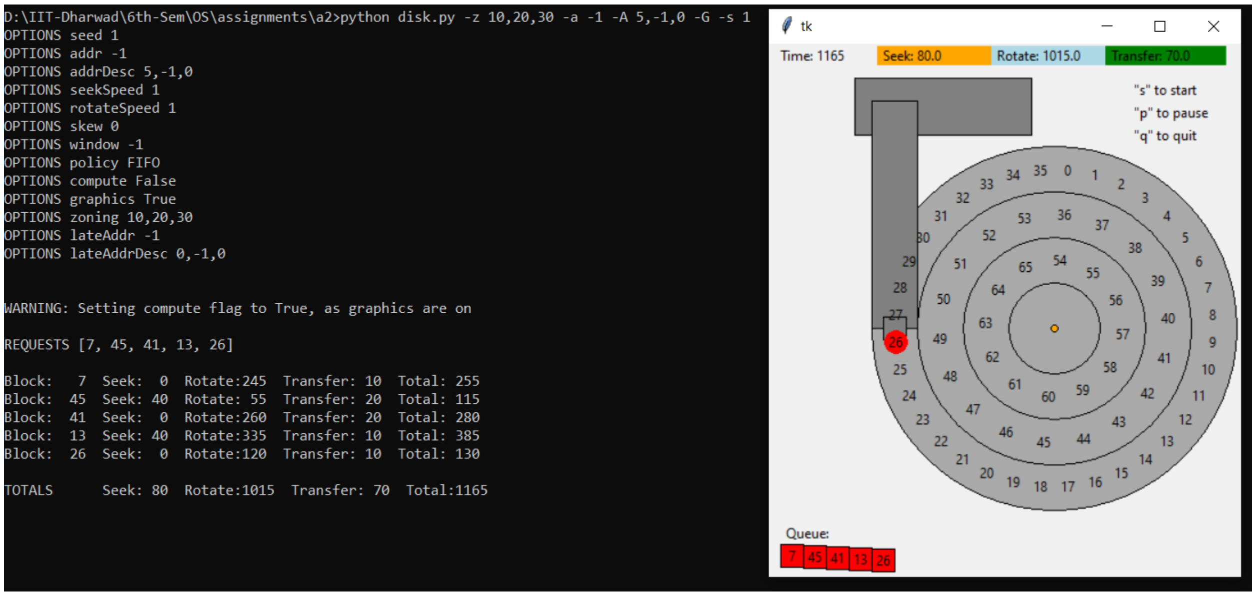
`python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G
OPTIONS seed 0
OPTIONS addr -1
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 10,20,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS [45, 40, 22, 13, 27]

Block:  45  Seek: 40  Rotate:310  Transfer: 20  Total: 370
Block:  40  Seek:  0  Rotate:240  Transfer: 20  Total: 260
Block:  22  Seek: 40  Rotate: 85  Transfer: 10  Total: 135
Block:  13  Seek:  0  Rotate:260  Transfer: 10  Total: 270
Block:  27  Seek:  0  Rotate:130  Transfer: 10  Total: 140

TOTALS      Seek: 80  Rotate:1025  Transfer: 70  Total:1175
```

Bandwidth (in sectors per unit time):
outer: 3/(135+270+140)=0.0055
middle: 2/(370+260)=0.0032

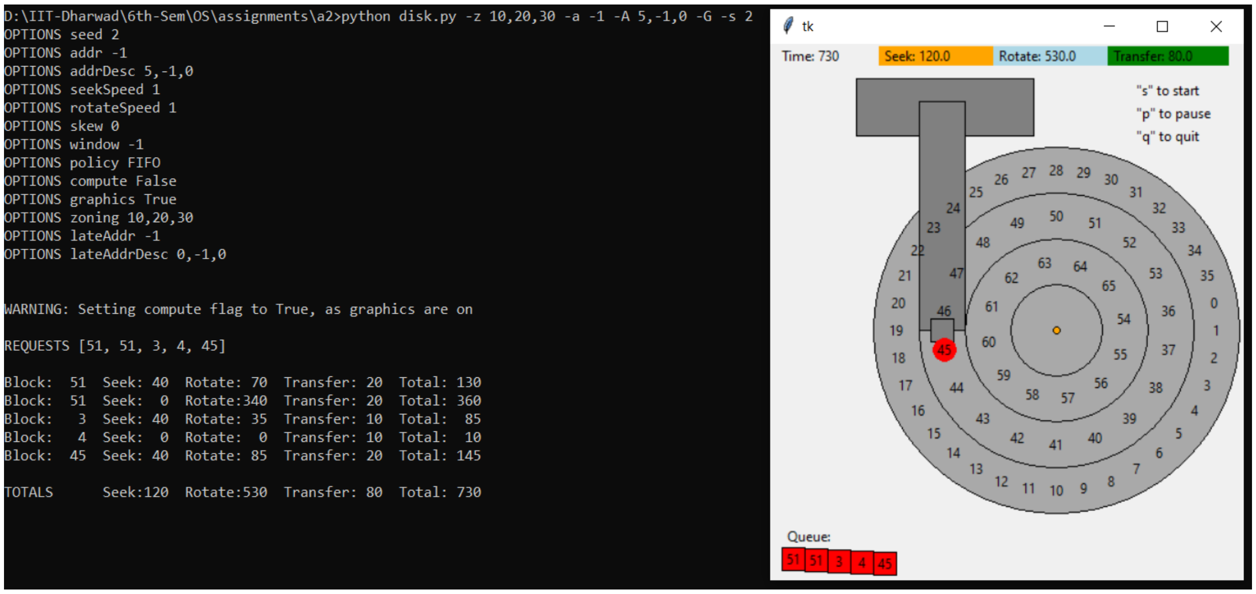
python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 1



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 1
OPTIONS seed 1
OPTIONS addr -1
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 10,20,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS [7, 45, 41, 13, 26]

Block:   7  Seek:  0  Rotate:245  Transfer: 10  Total: 255
Block:  45  Seek: 40  Rotate: 55  Transfer: 20  Total: 115
Block:  41  Seek:  0  Rotate:260  Transfer: 20  Total: 280
Block:  13  Seek: 40  Rotate:335  Transfer: 10  Total: 385
Block:  26  Seek:  0  Rotate:120  Transfer: 10  Total: 130

TOTALS      Seek: 80  Rotate:1015  Transfer: 70  Total:1165
```

Bandwidth (in sectors per unit time):
outer: 3/(255+385+130)=0.0039
middle: 2/(115+280)=0.0051

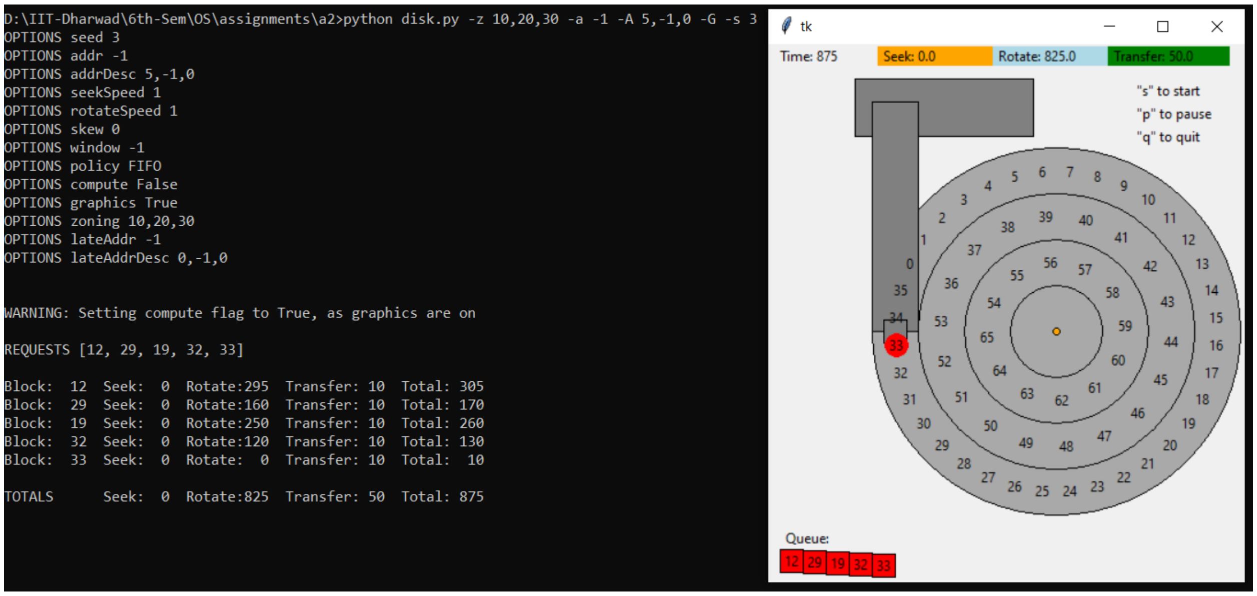
python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 2



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 2
OPTIONS seed 2
OPTIONS addr -1
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 10,20,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS [51, 51, 3, 4, 45]

Block:  51  Seek: 40  Rotate: 70  Transfer: 20  Total: 130
Block:  51  Seek:  0  Rotate:340  Transfer: 20  Total: 360
Block:   3  Seek: 40  Rotate: 35  Transfer: 10  Total:  85
Block:   4  Seek:  0  Rotate:  0  Transfer: 10  Total:  10
Block:  45  Seek: 40  Rotate: 85  Transfer: 20  Total: 145

TOTALS      Seek:120  Rotate:530  Transfer: 80  Total: 730
```

Bandwidth (in sectors per unit time):
outer: 2/(85+10)=0.0211
middle: 3/(130+360+145)=0.0047

python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 3



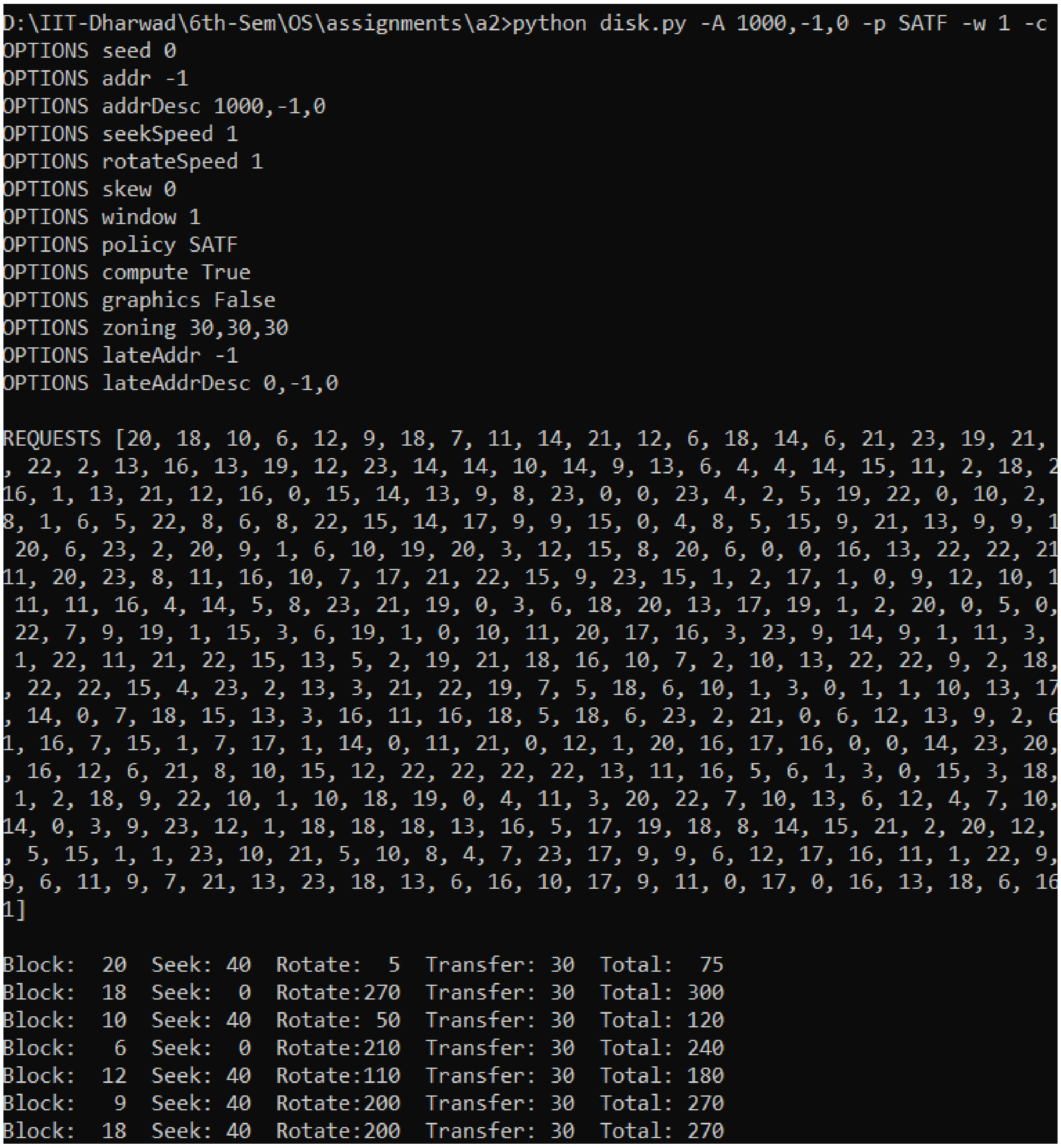
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -z 10,20,30 -a -1 -A 5,-1,0 -G -s 3
OPTIONS seed 3
OPTIONS addr -1
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 10,20,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS [12, 29, 19, 32, 33]

Block:  12  Seek:  0  Rotate:295  Transfer: 10  Total: 305
Block:  29  Seek:  0  Rotate:160  Transfer: 10  Total: 170
Block:  19  Seek:  0  Rotate:250  Transfer: 10  Total: 260
Block:  32  Seek:  0  Rotate:120  Transfer: 10  Total: 130
Block:  33  Seek:  0  Rotate:  0  Transfer: 10  Total:  10

TOTALS      Seek:  0  Rotate:825  Transfer: 50  Total: 875
```

Bandwidth (in sectors per unit time):
outer: 5/875=0.0057

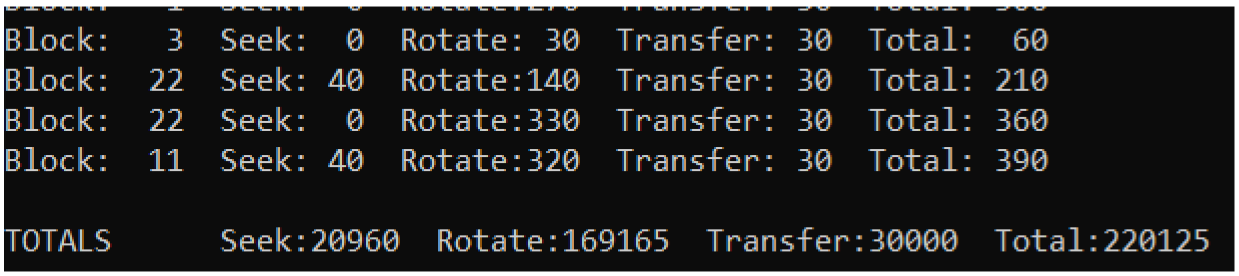
(8) The window needs to be the size of the number of requests to maximize performance. Window size of 1 makes all policies equal. From the below runs we can see that to maximize performance we need a window of size equal to the disk size. If we set the window size to 1 then we get the same output/total time for all the scheduling policies which is equal to 220125. So, the policy we are using doesn't matter in this case.

Command	**Time**
`python disk.py -A 1000,-1,0 -p SATF -w 1 -c`	220125
`python disk.py -A 1000,-1,0 -p FIFO -w 1 -c`	220125
`python disk.py -A 1000,-1,0 -p SSTF -w 1 -c`	220125
`python disk.py -A 1000,-1,0 -p BSATF -w 1 -c`	220125
`python disk.py -A 1000,-1,0 -p SATF -w 1000 -c`	35475

python disk.py -A 1000,-1,0 -p SATF -w 1 -c

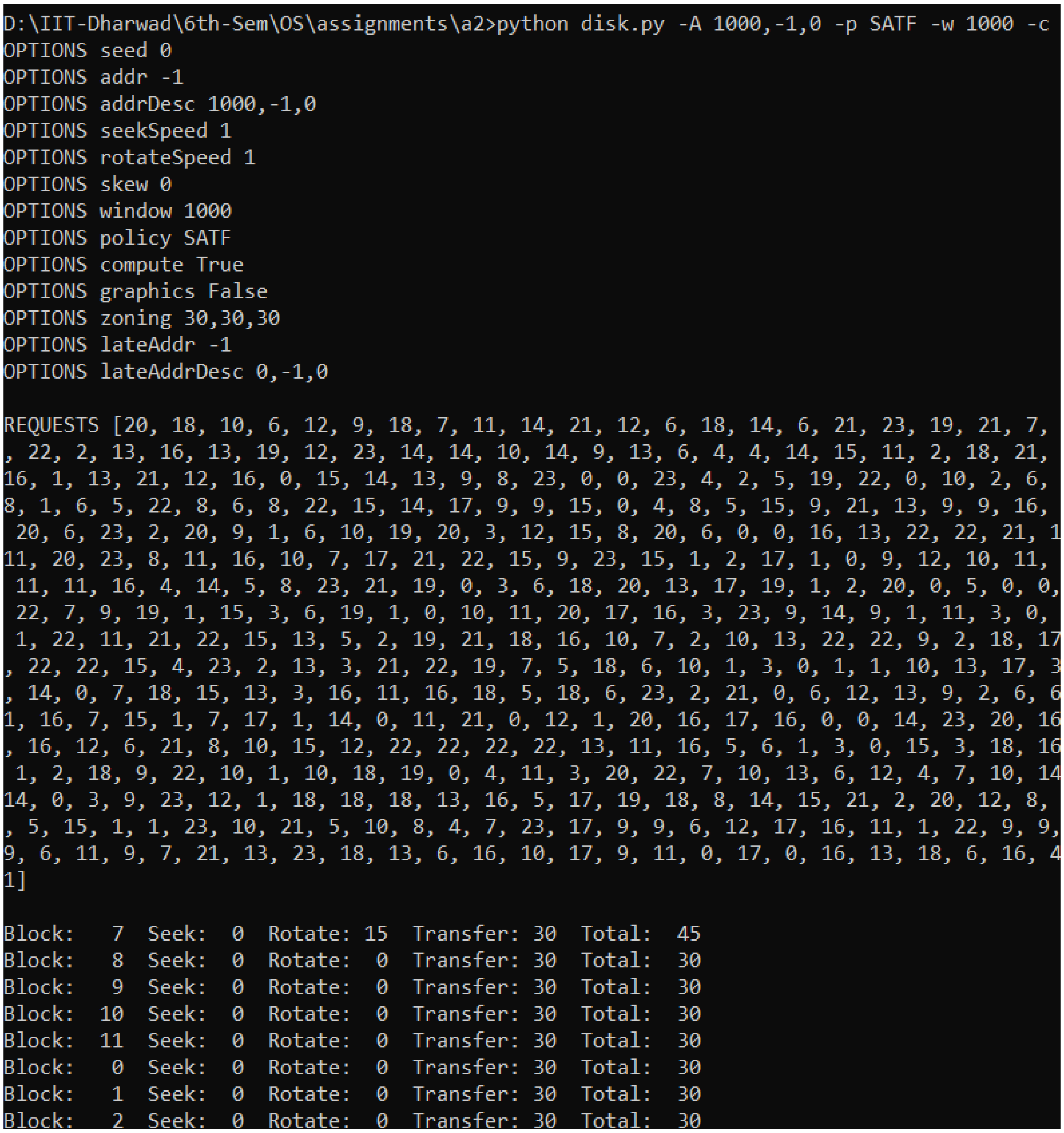


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -A 1000,-1,0 -p SATF -w 1 -c
OPTIONS seed 0
OPTIONS addr -1
OPTIONS addrDesc 1000,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window 1
OPTIONS policy SATF
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS [20, 18, 10, 6, 12, 9, 18, 7, 11, 14, 21, 12, 6, 18, 14, 6, 21, 23, 19, 21,
, 22, 2, 13, 16, 13, 19, 12, 23, 14, 14, 10, 14, 9, 13, 6, 4, 4, 14, 15, 11, 2, 18, 2
16, 1, 13, 21, 12, 16, 0, 15, 14, 13, 9, 8, 23, 0, 0, 23, 4, 2, 5, 19, 22, 0, 10, 2,
8, 1, 6, 5, 22, 8, 6, 8, 22, 15, 14, 17, 9, 9, 15, 0, 4, 8, 5, 15, 9, 21, 13, 9, 9, 1
 20, 6, 23, 2, 20, 9, 1, 6, 10, 19, 20, 3, 12, 15, 8, 20, 6, 0, 0, 16, 13, 22, 22, 21
11, 20, 23, 8, 11, 16, 10, 7, 17, 21, 22, 15, 9, 23, 15, 1, 2, 17, 1, 0, 9, 12, 10, 1
 11, 11, 16, 4, 14, 5, 8, 23, 21, 19, 0, 3, 6, 18, 20, 13, 17, 19, 1, 2, 20, 0, 5, 0,
 22, 7, 9, 19, 1, 15, 3, 6, 19, 1, 0, 10, 11, 20, 17, 16, 3, 23, 9, 14, 9, 1, 11, 3,
 1, 22, 11, 21, 22, 15, 13, 5, 2, 19, 21, 18, 16, 10, 7, 2, 10, 13, 22, 22, 9, 2, 18,
, 22, 22, 15, 4, 23, 2, 13, 3, 21, 22, 19, 7, 5, 18, 6, 10, 1, 3, 0, 1, 1, 10, 13, 17
, 14, 0, 7, 18, 15, 13, 3, 16, 11, 16, 18, 5, 18, 6, 23, 2, 21, 0, 6, 12, 13, 9, 2, 6
1, 16, 7, 15, 1, 7, 17, 1, 14, 0, 11, 21, 0, 12, 1, 20, 16, 17, 16, 0, 0, 14, 23, 20,
, 16, 12, 6, 21, 8, 10, 15, 12, 22, 22, 22, 22, 13, 11, 16, 5, 6, 1, 3, 0, 15, 3, 18,
 1, 2, 18, 9, 22, 10, 1, 10, 18, 19, 0, 4, 11, 3, 20, 22, 7, 10, 13, 6, 12, 4, 7, 10,
14, 0, 3, 9, 23, 12, 1, 18, 18, 18, 13, 16, 5, 17, 19, 18, 8, 14, 15, 21, 2, 20, 12,
, 5, 15, 1, 1, 23, 10, 21, 5, 10, 8, 4, 7, 23, 17, 9, 9, 6, 12, 17, 16, 11, 1, 22, 9,
9, 6, 11, 9, 7, 21, 13, 23, 18, 13, 6, 16, 10, 17, 9, 11, 0, 17, 0, 16, 13, 18, 6, 16
1]

Block:  20  Seek: 40  Rotate:  5  Transfer: 30  Total:  75
Block:  18  Seek:  0  Rotate:270  Transfer: 30  Total: 300
Block:  10  Seek: 40  Rotate: 50  Transfer: 30  Total: 120
Block:   6  Seek:  0  Rotate:210  Transfer: 30  Total: 240
Block:  12  Seek: 40  Rotate:110  Transfer: 30  Total: 180
Block:   9  Seek: 40  Rotate:200  Transfer: 30  Total: 270
Block:  18  Seek: 40  Rotate:200  Transfer: 30  Total: 270
```

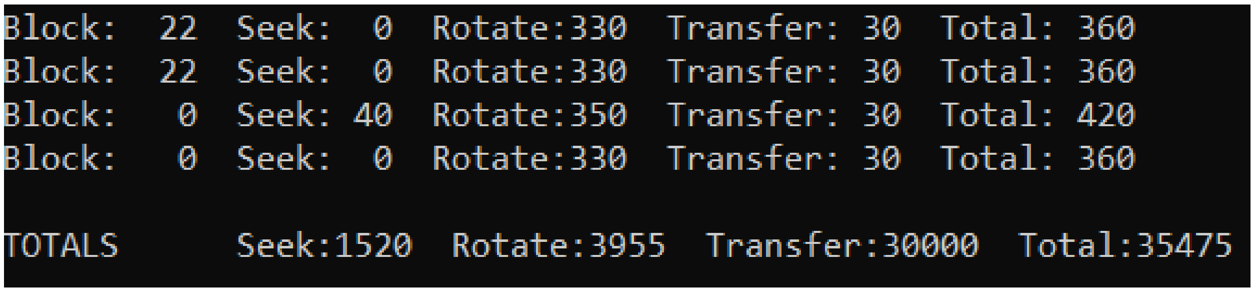


```
Block:   3  Seek:  0  Rotate: 30  Transfer: 30  Total:  60
Block:  22  Seek: 40  Rotate:140  Transfer: 30  Total: 210
Block:  22  Seek:  0  Rotate:330  Transfer: 30  Total: 360
Block:  11  Seek: 40  Rotate:320  Transfer: 30  Total: 390

TOTALS      Seek:20960  Rotate:169165  Transfer:30000  Total:220125
```

python disk.py -A 1000,-1,0 -p SATF -w 1000 -c



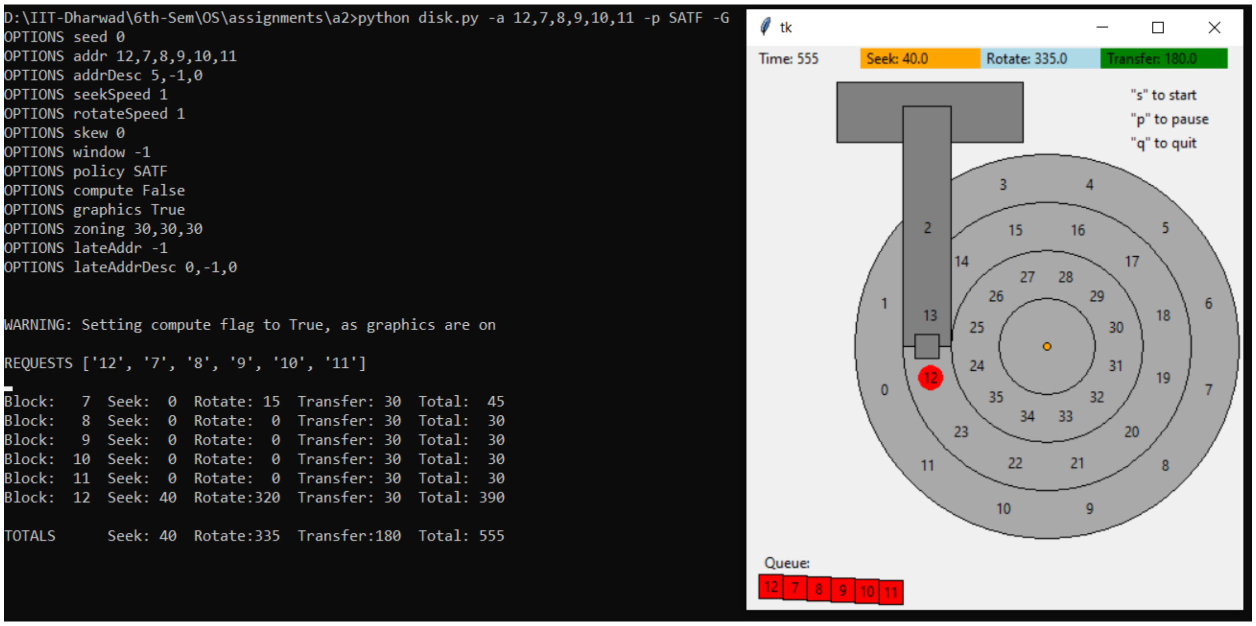
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -A 1000,-1,0 -p SATF -w 1000 -c
OPTIONS seed 0
OPTIONS addr -1
OPTIONS addrDesc 1000,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window 1000
OPTIONS policy SATF
OPTIONS compute True
OPTIONS graphics False
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0

REQUESTS [20, 18, 10, 6, 12, 9, 18, 7, 11, 14, 21, 12, 6, 18, 14, 6, 21, 23, 19, 21, 7,
, 22, 2, 13, 16, 13, 19, 12, 23, 14, 14, 10, 14, 9, 13, 6, 4, 4, 14, 15, 11, 2, 18, 21,
16, 1, 13, 21, 12, 16, 0, 15, 14, 13, 9, 8, 23, 0, 0, 23, 4, 2, 5, 19, 22, 0, 10, 2, 6,
8, 1, 6, 5, 22, 8, 6, 8, 22, 15, 14, 17, 9, 9, 15, 0, 4, 8, 5, 15, 9, 21, 13, 9, 9, 16,
 20, 6, 23, 2, 20, 9, 1, 6, 10, 19, 20, 3, 12, 15, 8, 20, 6, 0, 0, 16, 13, 22, 22, 21, 1
11, 20, 23, 8, 11, 16, 10, 7, 17, 21, 22, 15, 9, 23, 15, 1, 2, 17, 1, 0, 9, 12, 10, 11,
 11, 11, 16, 4, 14, 5, 8, 23, 21, 19, 0, 3, 6, 18, 20, 13, 17, 19, 1, 2, 20, 0, 5, 0, 0,
 22, 7, 9, 19, 1, 15, 3, 6, 19, 1, 0, 10, 11, 20, 17, 16, 3, 23, 9, 14, 9, 1, 11, 3, 0,
 1, 22, 11, 21, 22, 15, 13, 5, 2, 19, 21, 18, 16, 10, 7, 2, 10, 13, 22, 22, 9, 2, 18, 17
, 22, 22, 15, 4, 23, 2, 13, 3, 21, 22, 19, 7, 5, 18, 6, 10, 1, 3, 0, 1, 1, 10, 13, 17, 3
, 14, 0, 7, 18, 15, 13, 3, 16, 11, 16, 18, 5, 18, 6, 23, 2, 21, 0, 6, 12, 13, 9, 2, 6, 6
1, 16, 7, 15, 1, 7, 17, 1, 14, 0, 11, 21, 0, 12, 1, 20, 16, 17, 16, 0, 0, 14, 23, 20, 16
, 16, 12, 6, 21, 8, 10, 15, 12, 22, 22, 22, 22, 13, 11, 16, 5, 6, 1, 3, 0, 15, 3, 18, 16
 1, 2, 18, 9, 22, 10, 1, 10, 18, 19, 0, 4, 11, 3, 20, 22, 7, 10, 13, 6, 12, 4, 7, 10, 14
14, 0, 3, 9, 23, 12, 1, 18, 18, 18, 13, 16, 5, 17, 19, 18, 8, 14, 15, 21, 2, 20, 12, 8,
, 5, 15, 1, 1, 23, 10, 21, 5, 10, 8, 4, 7, 23, 17, 9, 9, 6, 12, 17, 16, 11, 1, 22, 9, 9,
9, 6, 11, 9, 7, 21, 13, 23, 18, 13, 6, 16, 10, 17, 9, 11, 0, 17, 0, 16, 13, 18, 6, 16, 4
1]

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   9  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  10  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   0  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   1  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   2  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
```



```
Block:  22  Seek:  0  Rotate:330  Transfer: 30  Total: 360
Block:  22  Seek:  0  Rotate:330  Transfer: 30  Total: 360
Block:   0  Seek: 40  Rotate:350  Transfer: 30  Total: 420
Block:   0  Seek:  0  Rotate:330  Transfer: 30  Total: 360

TOTALS      Seek:1520  Rotate:3955  Transfer:30000  Total:35475
```

(9) Here, we have a scheduling window set so the disk will first address all the queries in that window before moving on to the next. But BSATF performs worse than SATF in terms of total time (SATF 525 BSATF 555). But taking this window size would help us avoid starvation. More specifically if the window size is small then we can AVOID starvation. The trade off between performance and avoiding starvation generally depends on many factors and no particular relation can be established. If we have a sequence in which most access are in the same track except one then in such a case we can let starvation occur because if we use window and BSATF, then we have to seek to that request and address which adds a penalty of seeking and coming back to the previous track. If there is another request load that is compensating in terms of this seek penalty or if we have a seek speed then we can have better performance and avoid starvation. If there are requests from all three tracks and the requests from middle track are more, then it might not affect performance even if we use BSATF with window size. This decision to make the trade off also depends on the application that is making the request like if it is important then we address it first using BSATF. Sometimes both might give the same performance too.

`python disk.py -a 12,7,8,9,10,11 -p SATF -G`

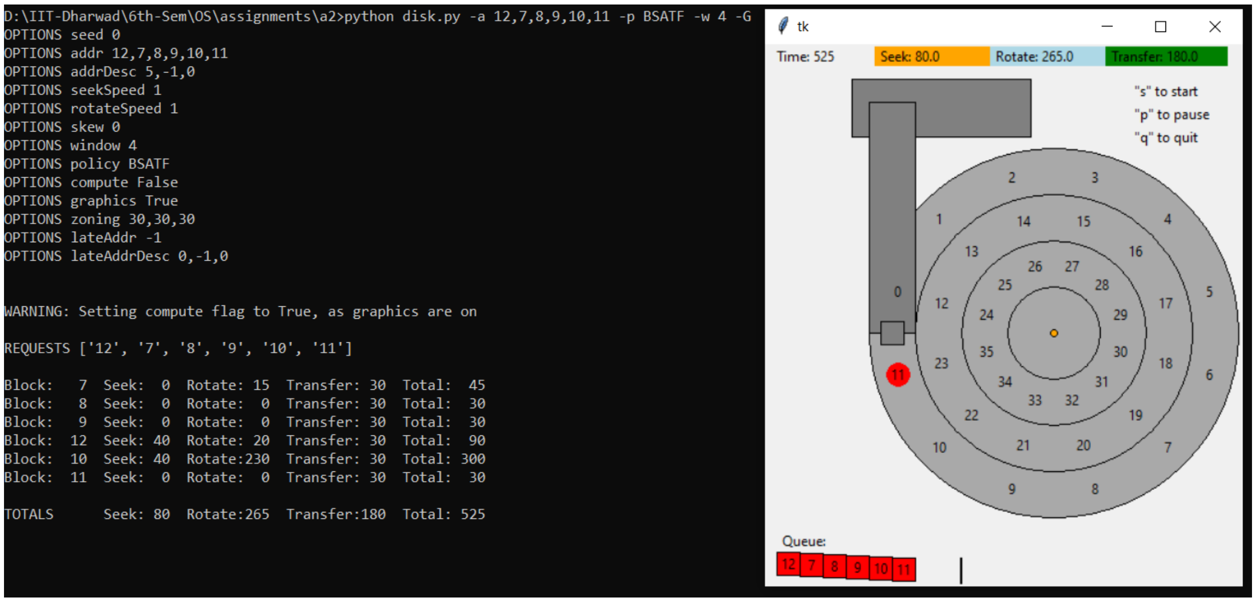


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 12,7,8,9,10,11 -p SATF -G
OPTIONS seed 0
OPTIONS addr 12,7,8,9,10,11
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SATF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['12', '7', '8', '9', '10', '11']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   9  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  10  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate:320  Transfer: 30  Total: 390

TOTALS      Seek: 40  Rotate:335  Transfer:180  Total: 555
```

Time: 555 Seek: 40.0 Rotate: 335.0 Transfer: 180.0

Order: 7,8,9,10,11,12

python disk.py -a 12,7,8,9,10,11 -p BSATF -w 4 -G

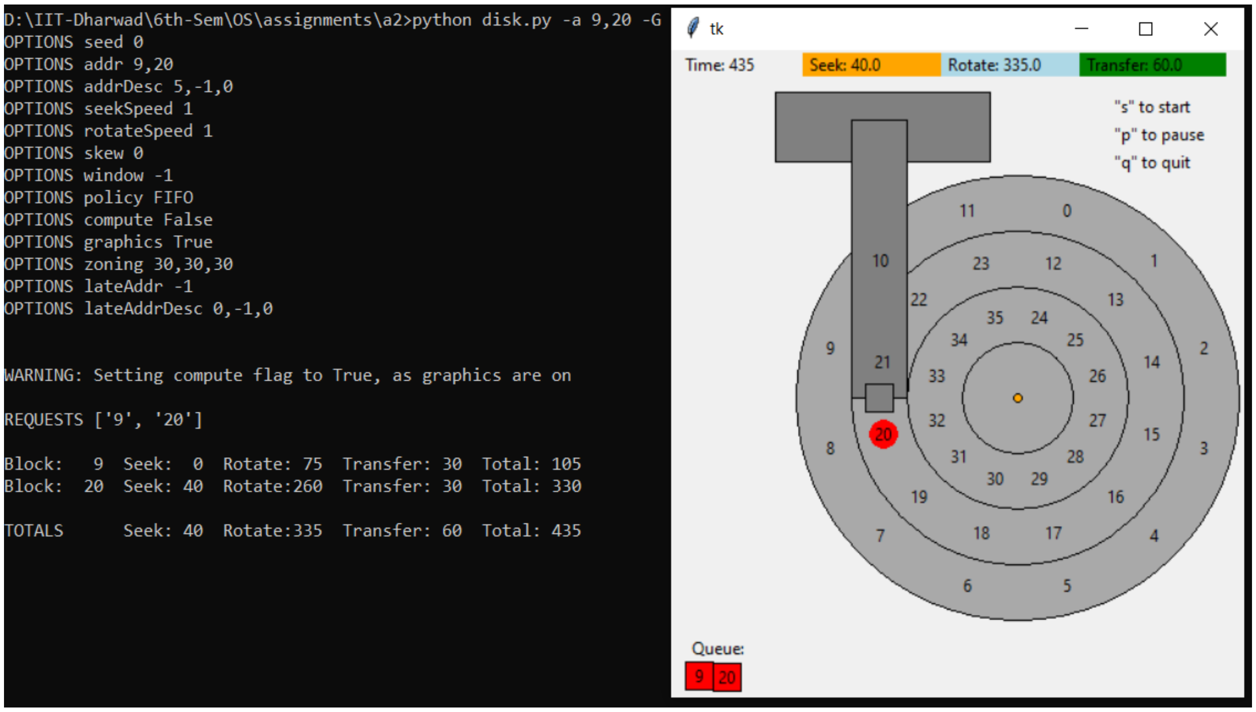


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 12,7,8,9,10,11 -p BSATF -w 4 -G
OPTIONS seed 0
OPTIONS addr 12,7,8,9,10,11
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window 4
OPTIONS policy BSATF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['12', '7', '8', '9', '10', '11']

Block:   7  Seek:  0  Rotate: 15  Transfer: 30  Total:  45
Block:   8  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:   9  Seek:  0  Rotate:  0  Transfer: 30  Total:  30
Block:  12  Seek: 40  Rotate: 20  Transfer: 30  Total:  90
Block:  10  Seek: 40  Rotate:230  Transfer: 30  Total: 300
Block:  11  Seek:  0  Rotate:  0  Transfer: 30  Total:  30

TOTALS      Seek: 80  Rotate:265  Transfer:180  Total: 525
```

Order: 7,8,9,12, 10,11

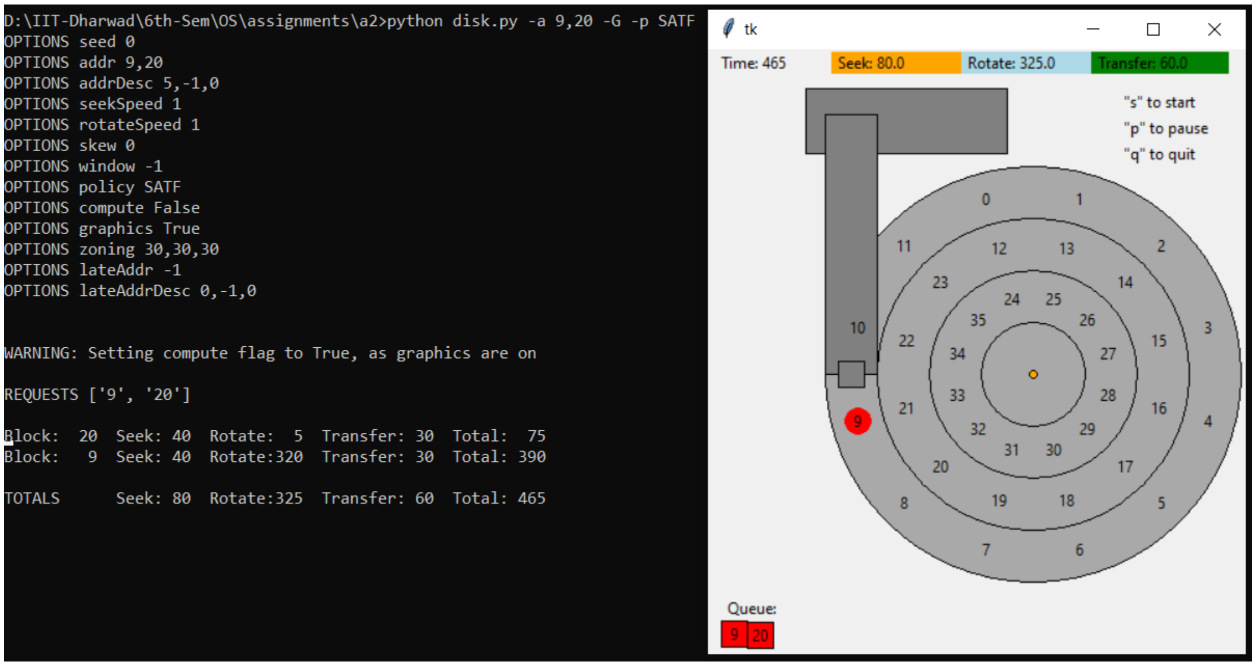
(10) The workload that I came up with that performs better if we go optimal rather than greedy is 9,20. Here we can see that the requests are addressed in order 9, 20 and the total time taken is 435. Now we run the same workload request with SATF scheduling which is greedy as it picks the workload with the next best access time. Now we can see that the total access time is 465 which is greater than 435. Thus for this workload 9,20 is the ideal order to address the requests but the greedy approach fails here by taking an order that leads to more access time.

`python disk.py -a 9,20 -G`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 9,20 -G
OPTIONS seed 0
OPTIONS addr 9,20
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy FIFO
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['9', '20']

Block:   9  Seek:  0  Rotate: 75  Transfer: 30  Total: 105
Block:  20  Seek: 40  Rotate:260  Transfer: 30  Total: 330

TOTALS      Seek: 40  Rotate:335  Transfer: 60  Total: 435
```

`python disk.py -a 9,20 -G -p SATF`

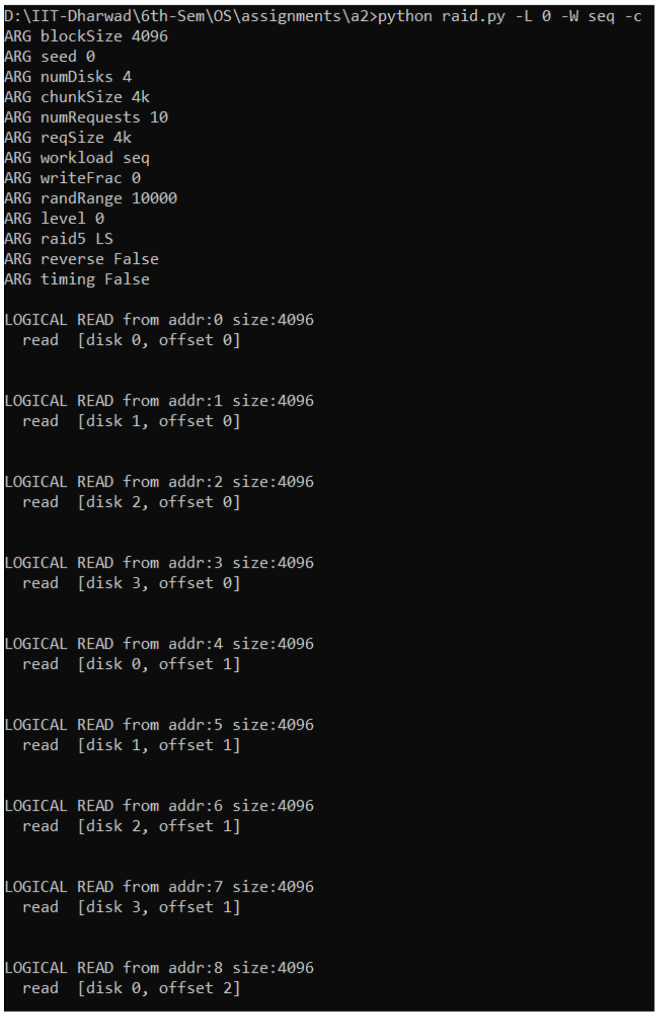


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python disk.py -a 9,20 -G -p SATF
OPTIONS seed 0
OPTIONS addr 9,20
OPTIONS addrDesc 5,-1,0
OPTIONS seekSpeed 1
OPTIONS rotateSpeed 1
OPTIONS skew 0
OPTIONS window -1
OPTIONS policy SATF
OPTIONS compute False
OPTIONS graphics True
OPTIONS zoning 30,30,30
OPTIONS lateAddr -1
OPTIONS lateAddrDesc 0,-1,0


WARNING: Setting compute flag to True, as graphics are on

REQUESTS ['9', '20']

Block:  20  Seek: 40  Rotate:  5  Transfer: 30  Total:  75
Block:   9  Seek: 40  Rotate:320  Transfer: 30  Total: 390

TOTALS      Seek: 80  Rotate:325  Transfer: 60  Total: 465
```

raid.py

(1)

`python raid.py -L 0 -c -W seq`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]


LOGICAL READ from addr:1 size:4096
  read  [disk 1, offset 0]


LOGICAL READ from addr:2 size:4096
  read  [disk 2, offset 0]


LOGICAL READ from addr:3 size:4096
  read  [disk 3, offset 0]


LOGICAL READ from addr:4 size:4096
  read  [disk 0, offset 1]


LOGICAL READ from addr:5 size:4096
  read  [disk 1, offset 1]


LOGICAL READ from addr:6 size:4096
  read  [disk 2, offset 1]


LOGICAL READ from addr:7 size:4096
  read  [disk 3, offset 1]


LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]
```

From the figure above we can see the requests made for them to calculate the Disk and Offset we use the below formulae:

Disk = Addr % disk_count
Offset = Addr / disk_count

In RAID 0 disk count = 4

So, for the first Read request the Address is 0:

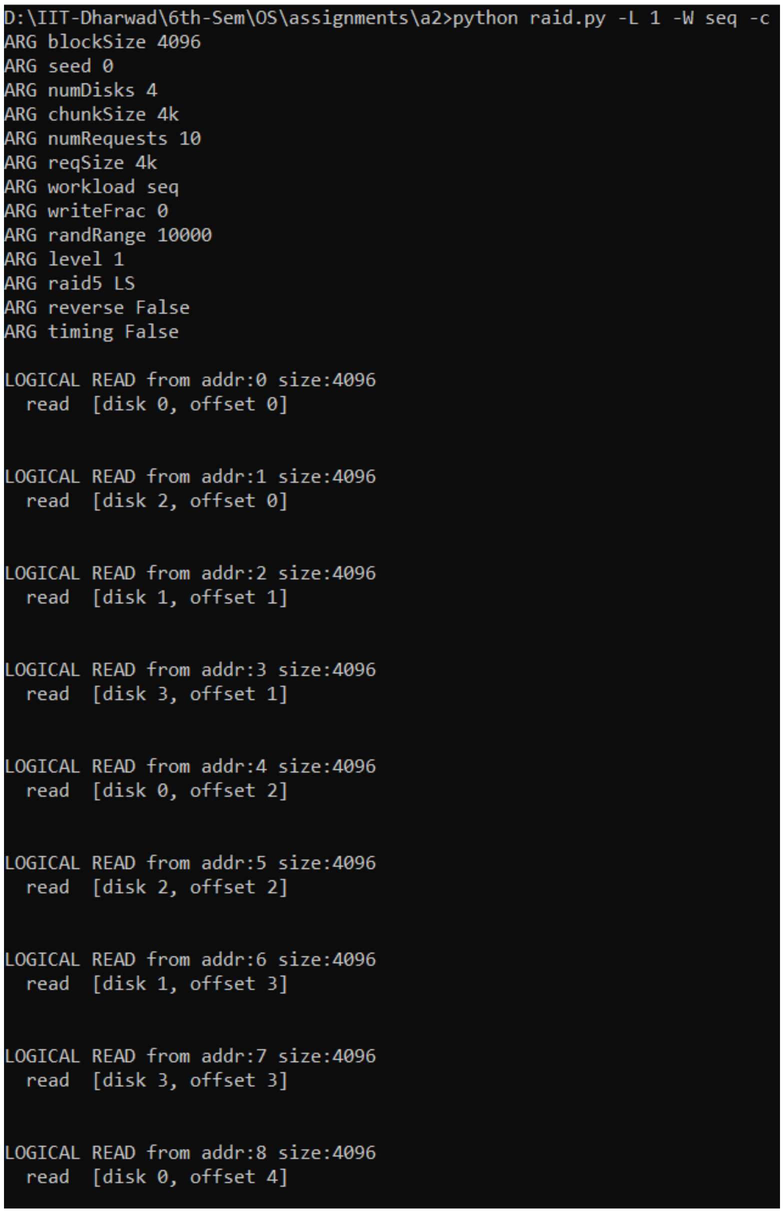
Disk = 0%4 = 0
Offset = 0/4 = 0

For the next read request, the address is 1:

Disk = 1%4 = 1
Offset = 1/4 = 1051

Similarly we can calculate the rest and the results can be verified with the output from the above screenshot.

`python raid.py -L 1 -c -W seq`

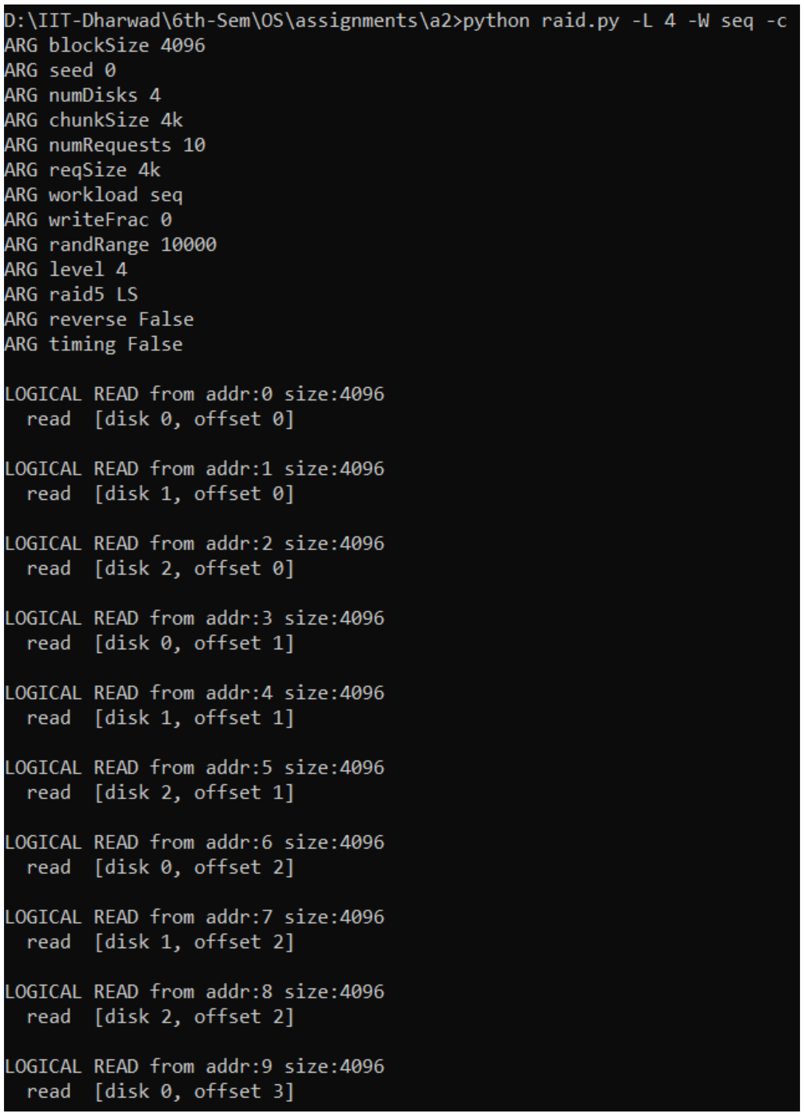


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]


LOGICAL READ from addr:1 size:4096
  read  [disk 2, offset 0]


LOGICAL READ from addr:2 size:4096
  read  [disk 1, offset 1]


LOGICAL READ from addr:3 size:4096
  read  [disk 3, offset 1]


LOGICAL READ from addr:4 size:4096
  read  [disk 0, offset 2]


LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 2]


LOGICAL READ from addr:6 size:4096
  read  [disk 1, offset 3]


LOGICAL READ from addr:7 size:4096
  read  [disk 3, offset 3]


LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 4]
```

In RAID 1 we have mirroring so the data in disks 0,1 is the same and in disks 2,3 is the same. So number of disks = 2
For request 7 the address is 6:
Here the disk = 6%2 = 0;
And offset = 6/2 = 3.

Here we have the read balancing in RAID 1 implementation so the disk actually becomes 1 since offset%2 != 0. So, disk = 1 ; offset = 3.
If offset%2 == 0 then the disk will be 0 or 2 respectively for respective disks. Else if offset%2 != 0 then it will be 1 or 3.
Similarly we can calculate the rest and the results can be verified with the output from the above screenshot.

`python raid.py -L 4 -5 LS -c -W seq`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -W seq -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:2 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:8 size:4096
  read  [disk 2, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 0, offset 3]
```

In RAID 4 we have 1 disk for parity so the number of disks = 3

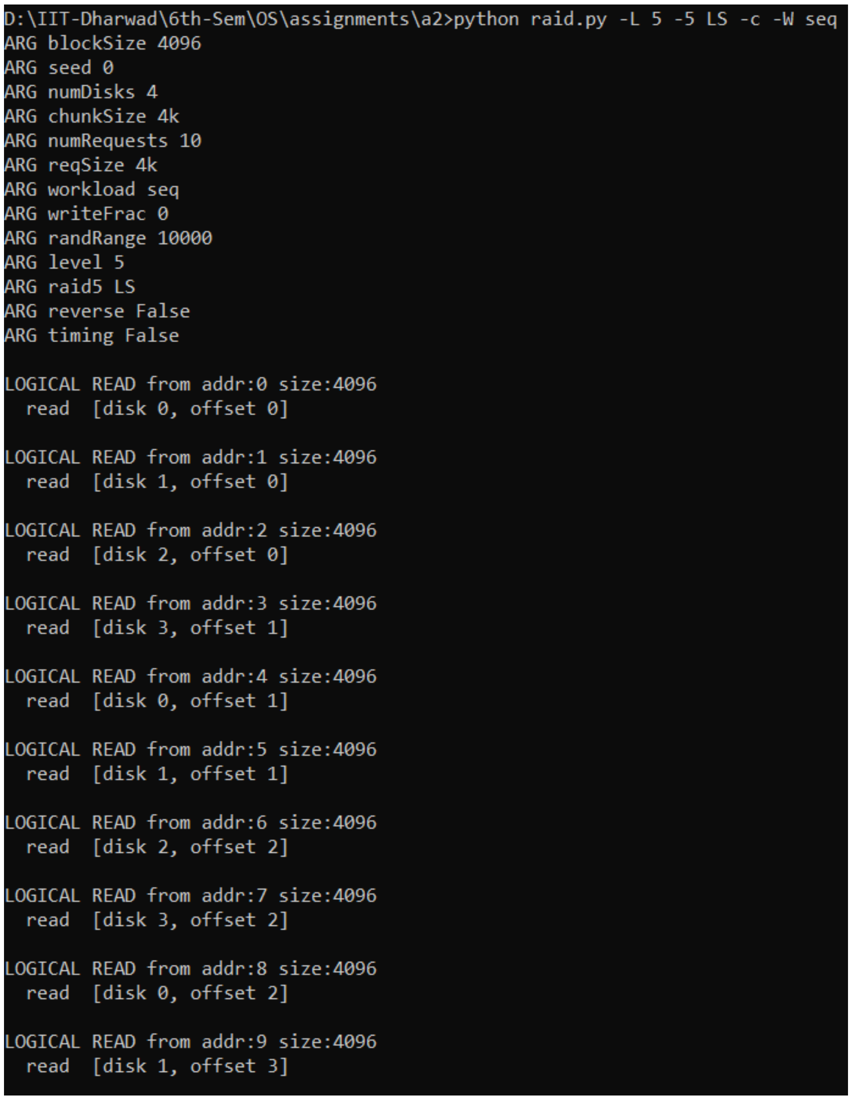
For request 5 the address is 4:
Here the disk = 4%3 = 1;
And offset = 4/3 = 1.

Similarly we can calculate the rest and the results can be verified with the output from the above screenshot.

RAID level	mappings
0	0 1 2 3 4 5 6 7 8 9 P P
1	0 P 1 P P 2 P 3 4 P 5 P P 6 P 7 8 P 9 P
4	0 1 2 P 3 4 5 P 6 7 8 P 9 P P P

RAID level	left-symmetric layout	left-asymmetric layout
5	0 1 2 P 4 5 P 3 8 P 6 7 9 P P P	0 1 2 P 3 4 P 5 6 P 7 8 9 P P P

python raid.py -L 5 -5 LS -c -W seq



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LS -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:2 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 3, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:5 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 2, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]
```

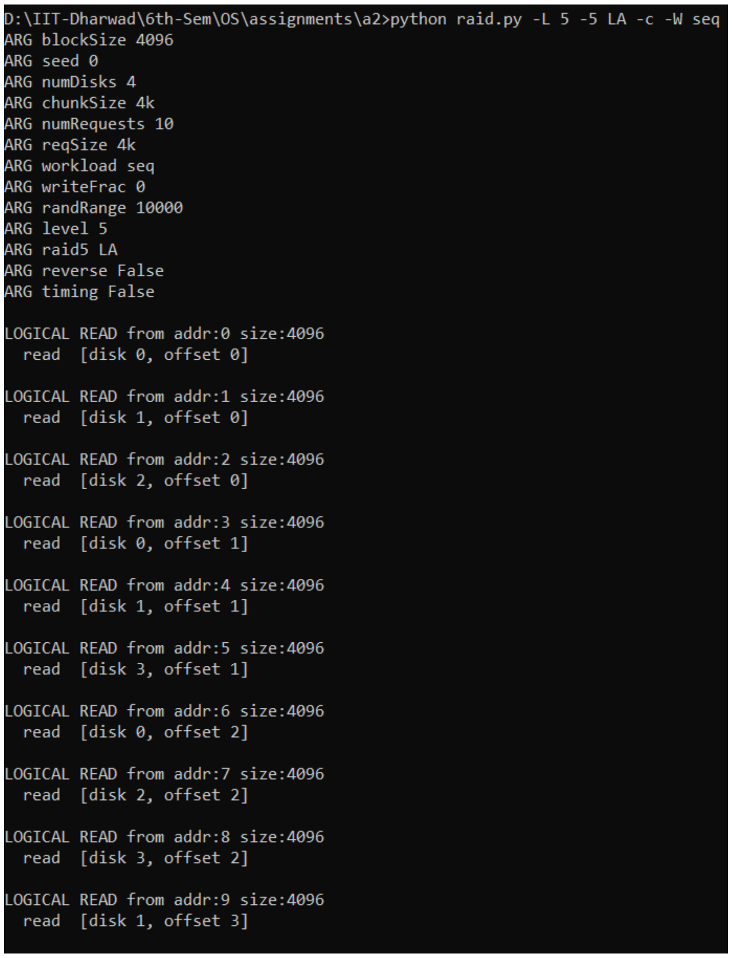
In RAID 5 we have a parity entry on all hard disks so the number of disks = 4 and here we are following Left symmetric by default.

For request 4 the address is 3:
Here the disk = 3%4 = 3;

The offset denominator becomes 3 as in the simulation implementation because each disk has 1 parity and here we are using "Left Symmetric" by default.
In Symmetric whenever we see a parity we move sideways and down to the next strip set as we can see here that leads to 3 according to the calculation.
And offset = 3/3 = 1.

Similarly we can calculate the rest and the results can be verified with the output from the above screenshot.

python raid.py -L 5 -5 LA -c -W seq



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LA -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LA
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:2 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:5 size:4096
  read  [disk 3, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 2, offset 2]

LOGICAL READ from addr:8 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]
```

In RAID 5 we have a parity entry on all hard disks so the number of disks = 4 and here we are following Left Asymmetric by default.

For request 8 the address is 0. The disk becomes 3 as in the simulation implementation because each disk has 1 parity and here we are using “Left Asymmetric”. In Asymmetric whenever we encounter a parity we skip and move to the next available space in this case it is 3 as we obtain from the calculations.

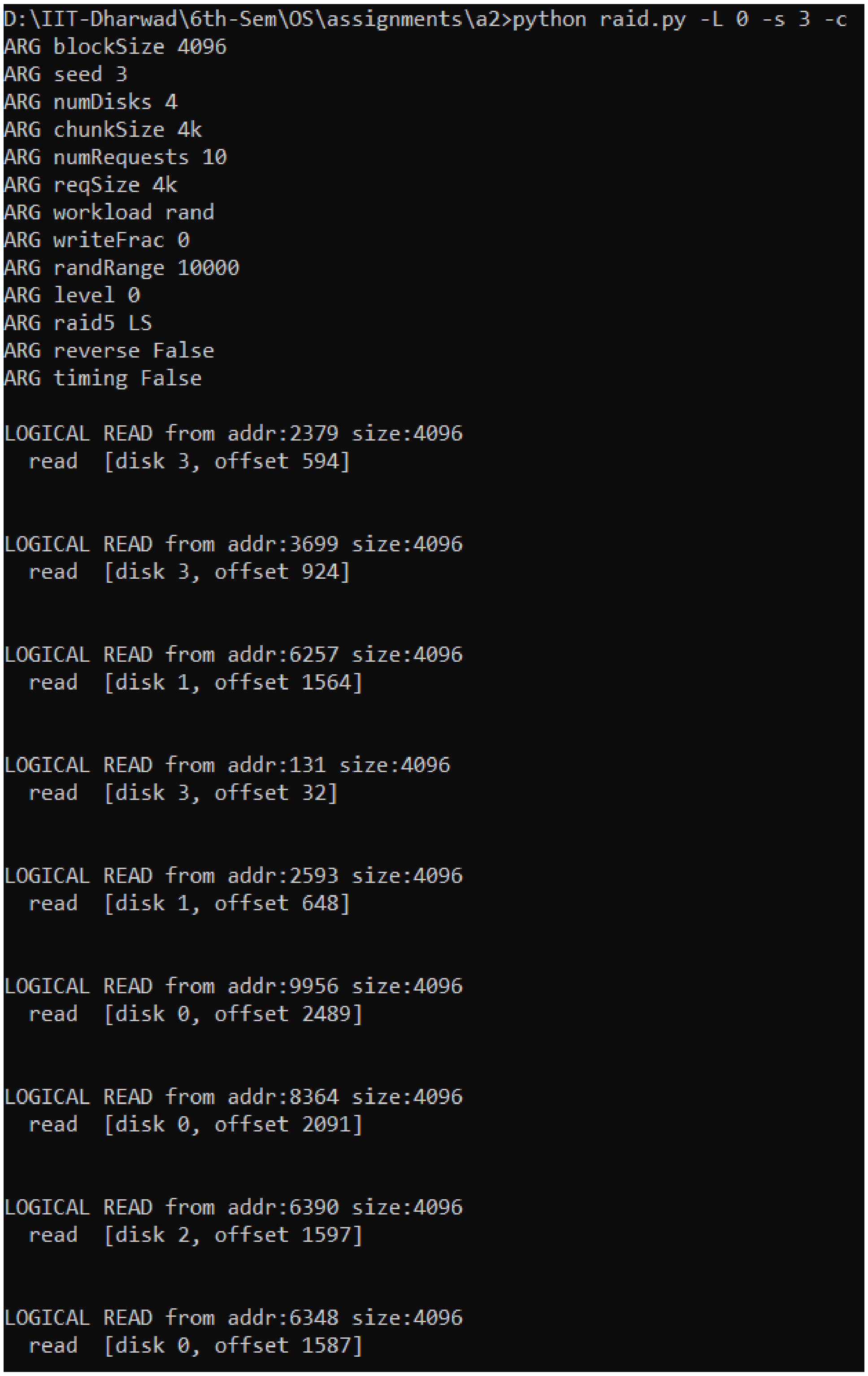
Here the disk = 3;
And offset = 8/3 = 2.

Similarly we can calculate the rest and the results can be verified with the output from the above screenshot.

From the above we can say that the difference between Symmetric and Asymmetric is: In asymmetric RAIDs the data strips ignore parity, they skip it until they reach the next available space. Symmetric RAIDs handle data strips in a slightly different(or complex) way; once the data encounters a parity block, they move sideways and down to the next stripe set.

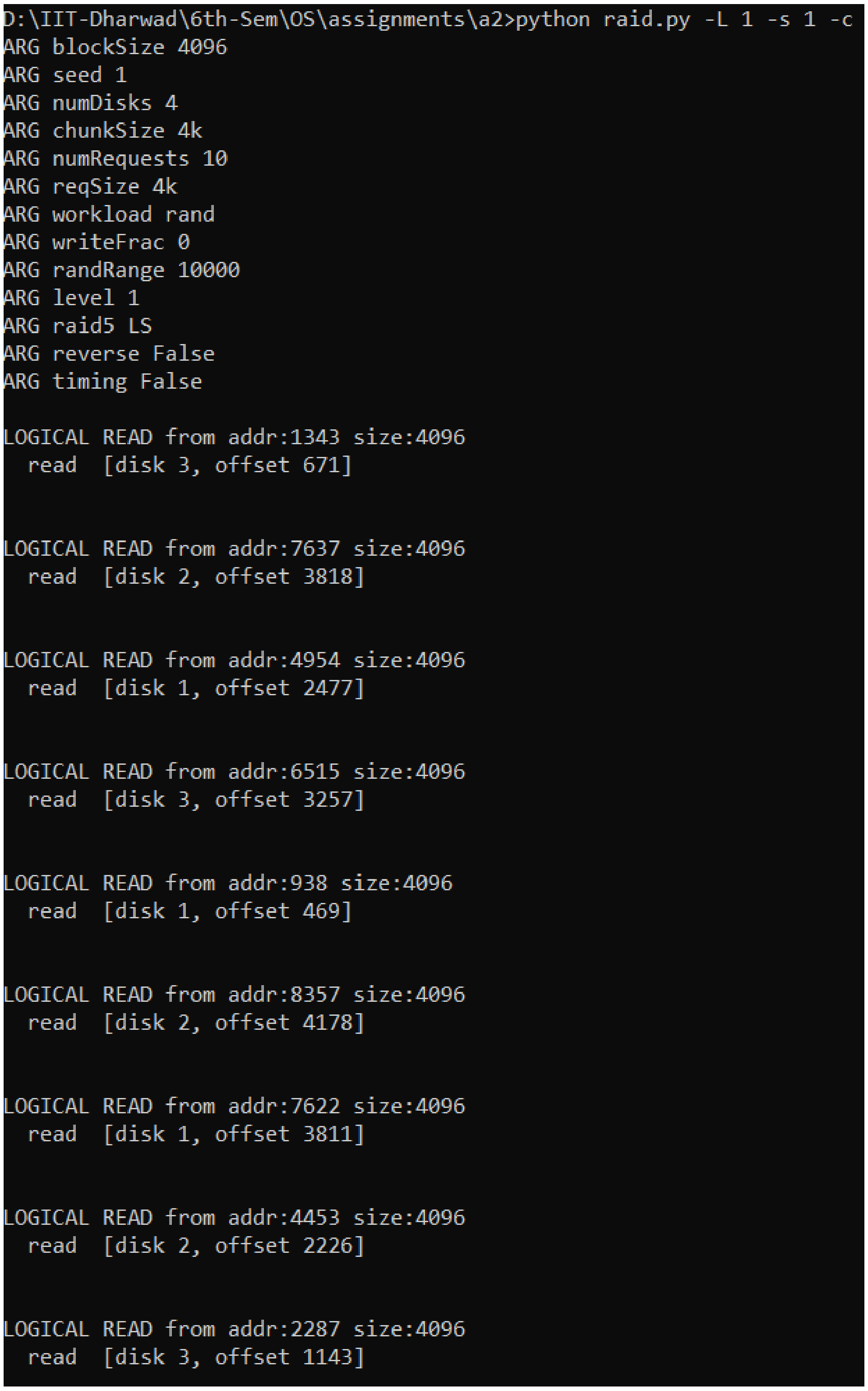
Some problems generated using different random seeds:

`python raid.py -L 0 -s 3 -c`



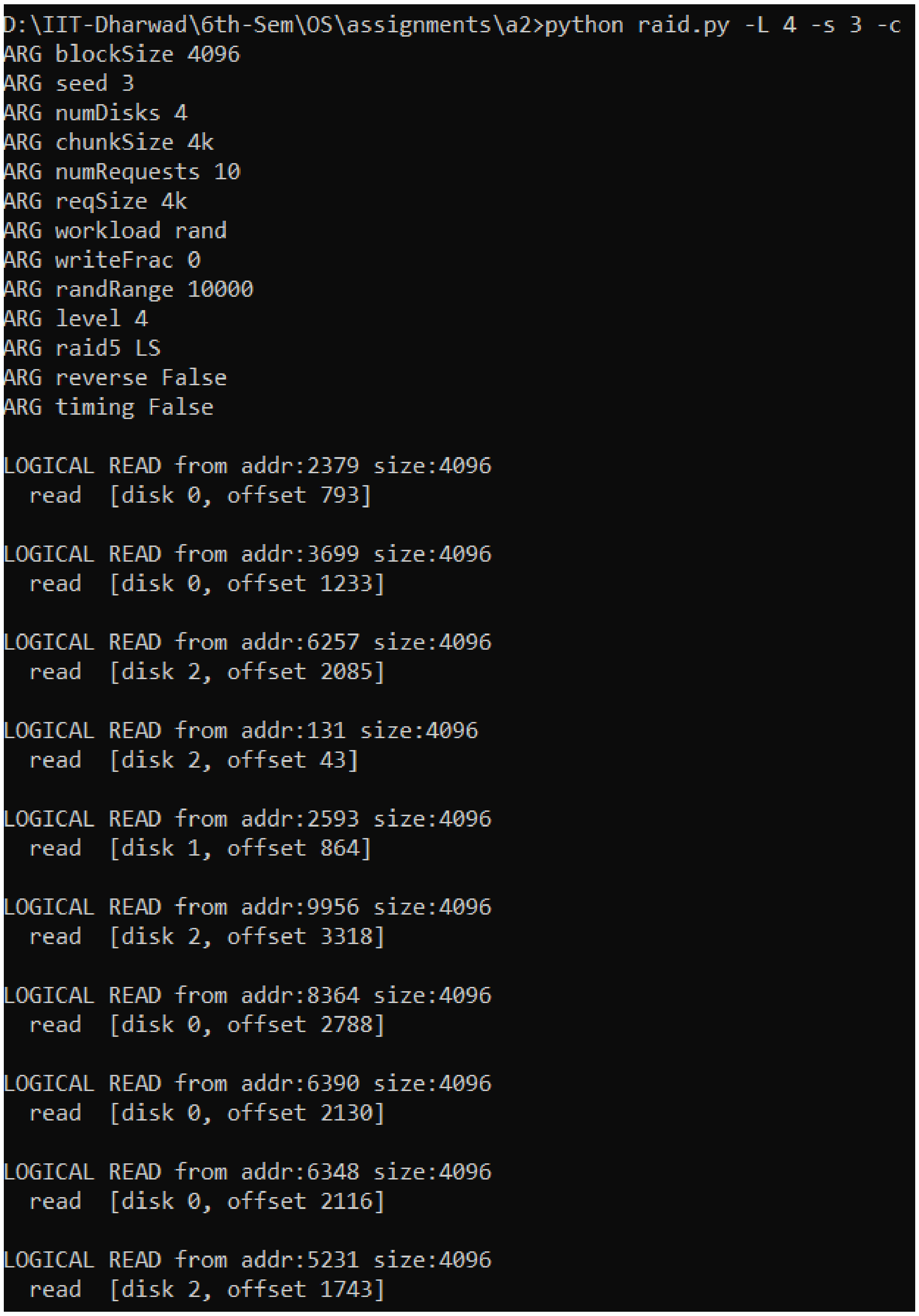
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -s 3 -c
ARG blockSize 4096
ARG seed 3
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:2379 size:4096
  read  [disk 3, offset 594]


LOGICAL READ from addr:3699 size:4096
  read  [disk 3, offset 924]


LOGICAL READ from addr:6257 size:4096
  read  [disk 1, offset 1564]


LOGICAL READ from addr:131 size:4096
  read  [disk 3, offset 32]


LOGICAL READ from addr:2593 size:4096
  read  [disk 1, offset 648]


LOGICAL READ from addr:9956 size:4096
  read  [disk 0, offset 2489]


LOGICAL READ from addr:8364 size:4096
  read  [disk 0, offset 2091]


LOGICAL READ from addr:6390 size:4096
  read  [disk 2, offset 1597]


LOGICAL READ from addr:6348 size:4096
  read  [disk 0, offset 1587]
```

python raid.py -L 1 -s 1 -c



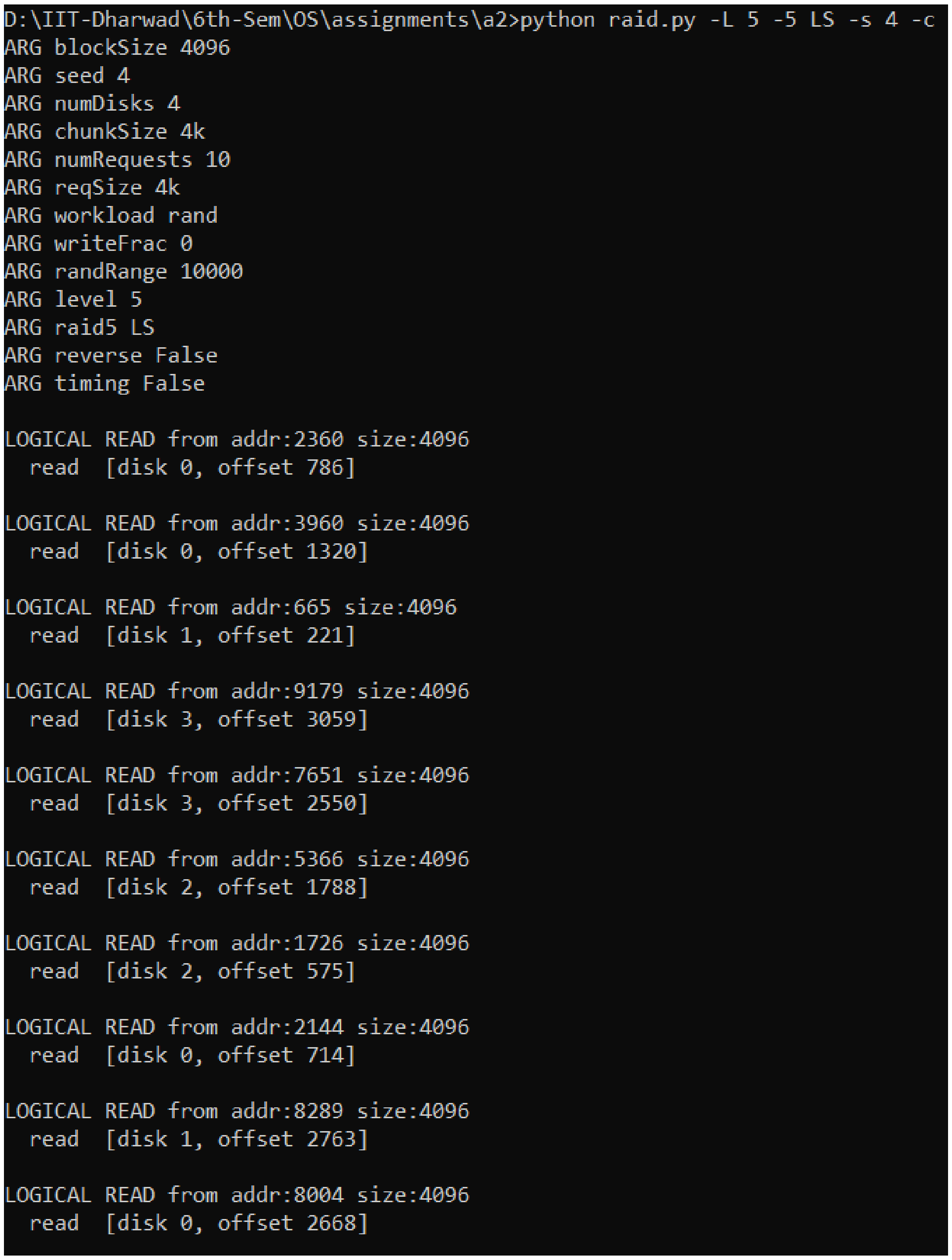
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -s 1 -c
ARG blockSize 4096
ARG seed 1
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:1343 size:4096
  read  [disk 3, offset 671]


LOGICAL READ from addr:7637 size:4096
  read  [disk 2, offset 3818]


LOGICAL READ from addr:4954 size:4096
  read  [disk 1, offset 2477]


LOGICAL READ from addr:6515 size:4096
  read  [disk 3, offset 3257]


LOGICAL READ from addr:938 size:4096
  read  [disk 1, offset 469]


LOGICAL READ from addr:8357 size:4096
  read  [disk 2, offset 4178]


LOGICAL READ from addr:7622 size:4096
  read  [disk 1, offset 3811]


LOGICAL READ from addr:4453 size:4096
  read  [disk 2, offset 2226]


LOGICAL READ from addr:2287 size:4096
  read  [disk 3, offset 1143]
```

python raid.py -L 4 -s 3 -c



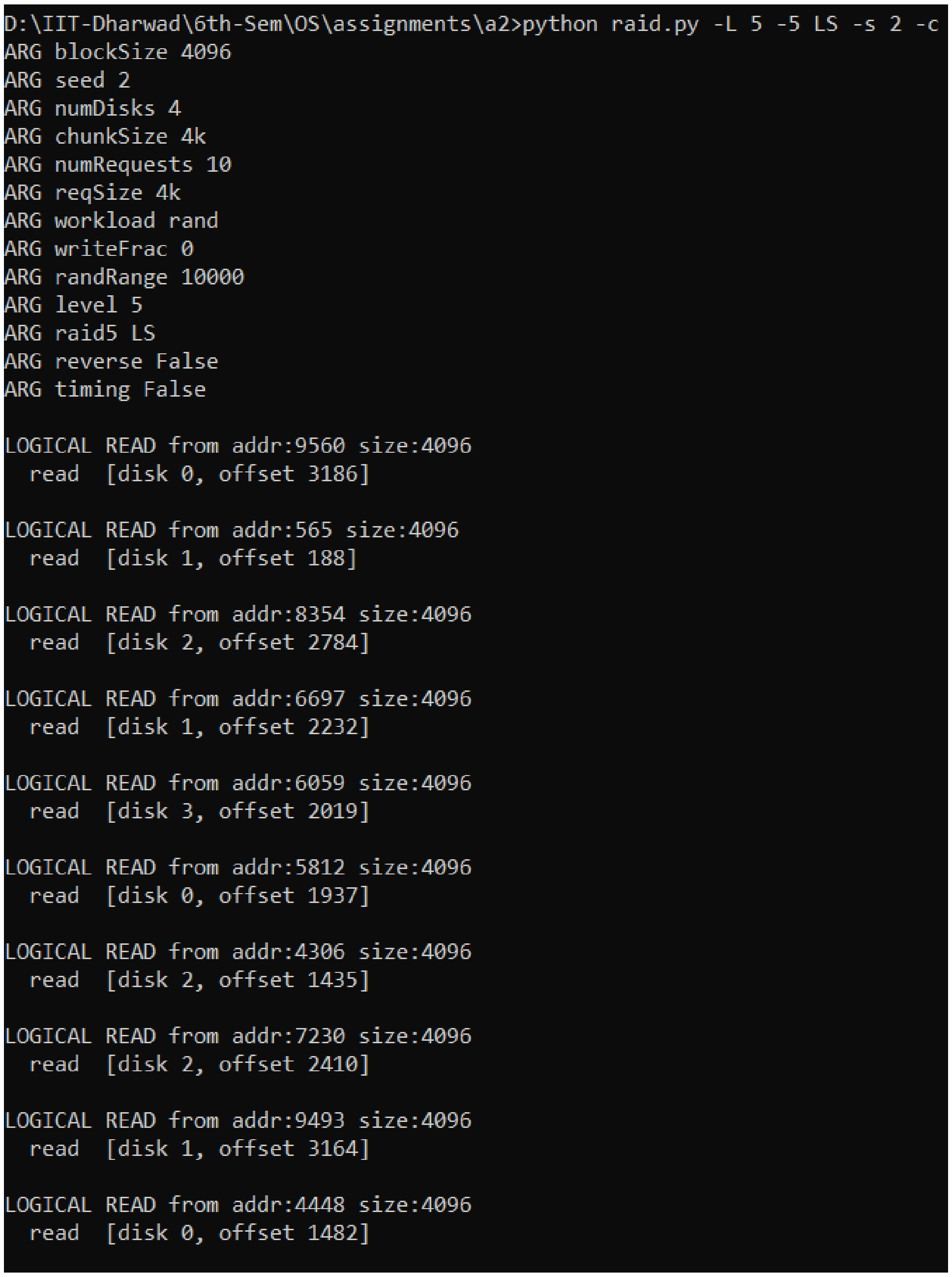
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -s 3 -c
ARG blockSize 4096
ARG seed 3
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:2379 size:4096
  read  [disk 0, offset 793]

LOGICAL READ from addr:3699 size:4096
  read  [disk 0, offset 1233]

LOGICAL READ from addr:6257 size:4096
  read  [disk 2, offset 2085]

LOGICAL READ from addr:131 size:4096
  read  [disk 2, offset 43]

LOGICAL READ from addr:2593 size:4096
  read  [disk 1, offset 864]

LOGICAL READ from addr:9956 size:4096
  read  [disk 2, offset 3318]

LOGICAL READ from addr:8364 size:4096
  read  [disk 0, offset 2788]

LOGICAL READ from addr:6390 size:4096
  read  [disk 0, offset 2130]

LOGICAL READ from addr:6348 size:4096
  read  [disk 0, offset 2116]

LOGICAL READ from addr:5231 size:4096
  read  [disk 2, offset 1743]
```

python raid.py -L 5 -5 LS -s 4 -c



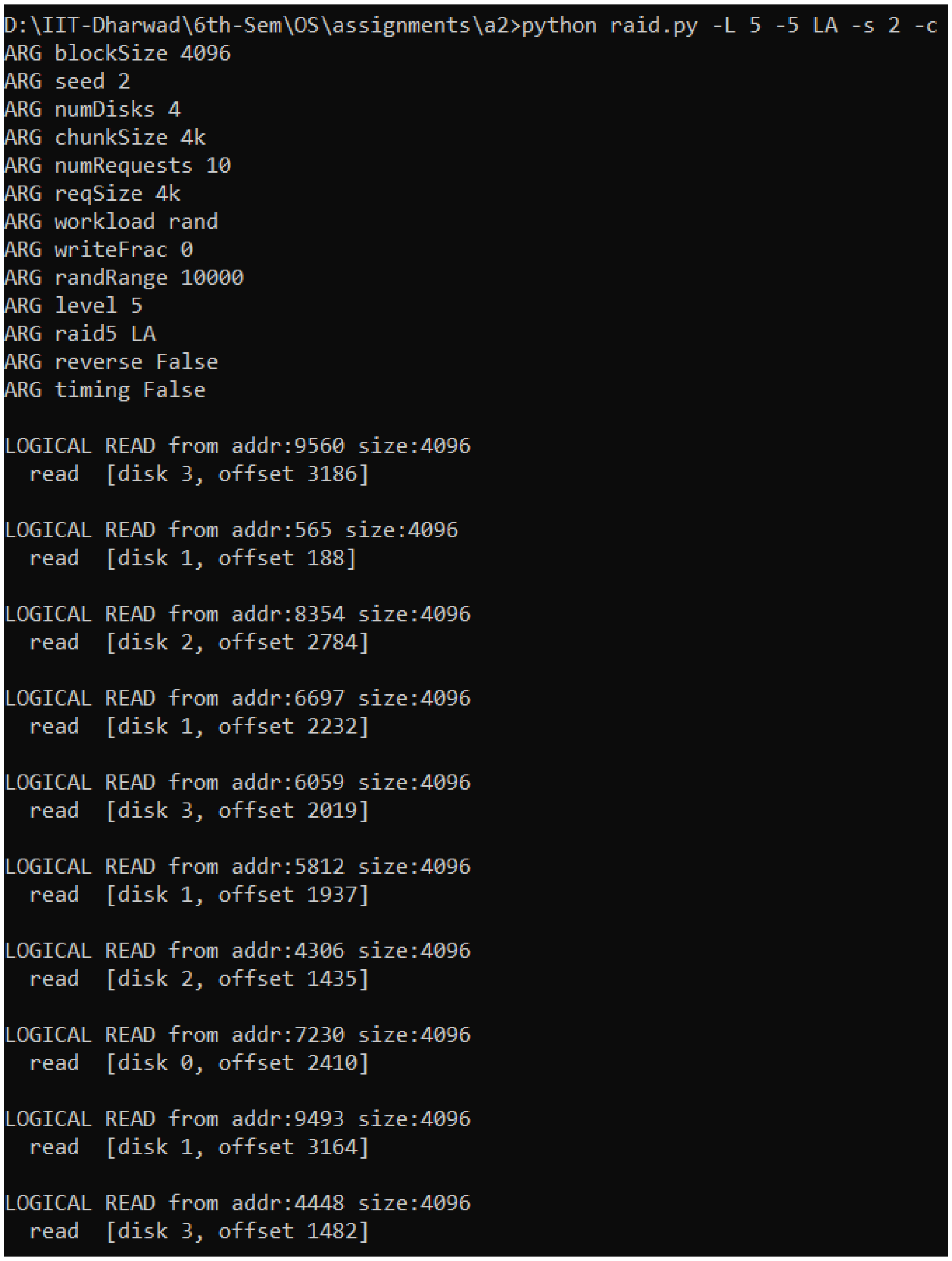
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LS -s 4 -c
ARG blockSize 4096
ARG seed 4
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:2360 size:4096
  read  [disk 0, offset 786]

LOGICAL READ from addr:3960 size:4096
  read  [disk 0, offset 1320]

LOGICAL READ from addr:665 size:4096
  read  [disk 1, offset 221]

LOGICAL READ from addr:9179 size:4096
  read  [disk 3, offset 3059]

LOGICAL READ from addr:7651 size:4096
  read  [disk 3, offset 2550]

LOGICAL READ from addr:5366 size:4096
  read  [disk 2, offset 1788]

LOGICAL READ from addr:1726 size:4096
  read  [disk 2, offset 575]

LOGICAL READ from addr:2144 size:4096
  read  [disk 0, offset 714]

LOGICAL READ from addr:8289 size:4096
  read  [disk 1, offset 2763]

LOGICAL READ from addr:8004 size:4096
  read  [disk 0, offset 2668]
```

python raid.py -L 5 -5 LS -s 2 -c



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LS -s 2 -c
ARG blockSize 4096
ARG seed 2
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:9560 size:4096
  read  [disk 0, offset 3186]

LOGICAL READ from addr:565 size:4096
  read  [disk 1, offset 188]

LOGICAL READ from addr:8354 size:4096
  read  [disk 2, offset 2784]

LOGICAL READ from addr:6697 size:4096
  read  [disk 1, offset 2232]

LOGICAL READ from addr:6059 size:4096
  read  [disk 3, offset 2019]

LOGICAL READ from addr:5812 size:4096
  read  [disk 0, offset 1937]

LOGICAL READ from addr:4306 size:4096
  read  [disk 2, offset 1435]

LOGICAL READ from addr:7230 size:4096
  read  [disk 2, offset 2410]

LOGICAL READ from addr:9493 size:4096
  read  [disk 1, offset 3164]

LOGICAL READ from addr:4448 size:4096
  read  [disk 0, offset 1482]
```

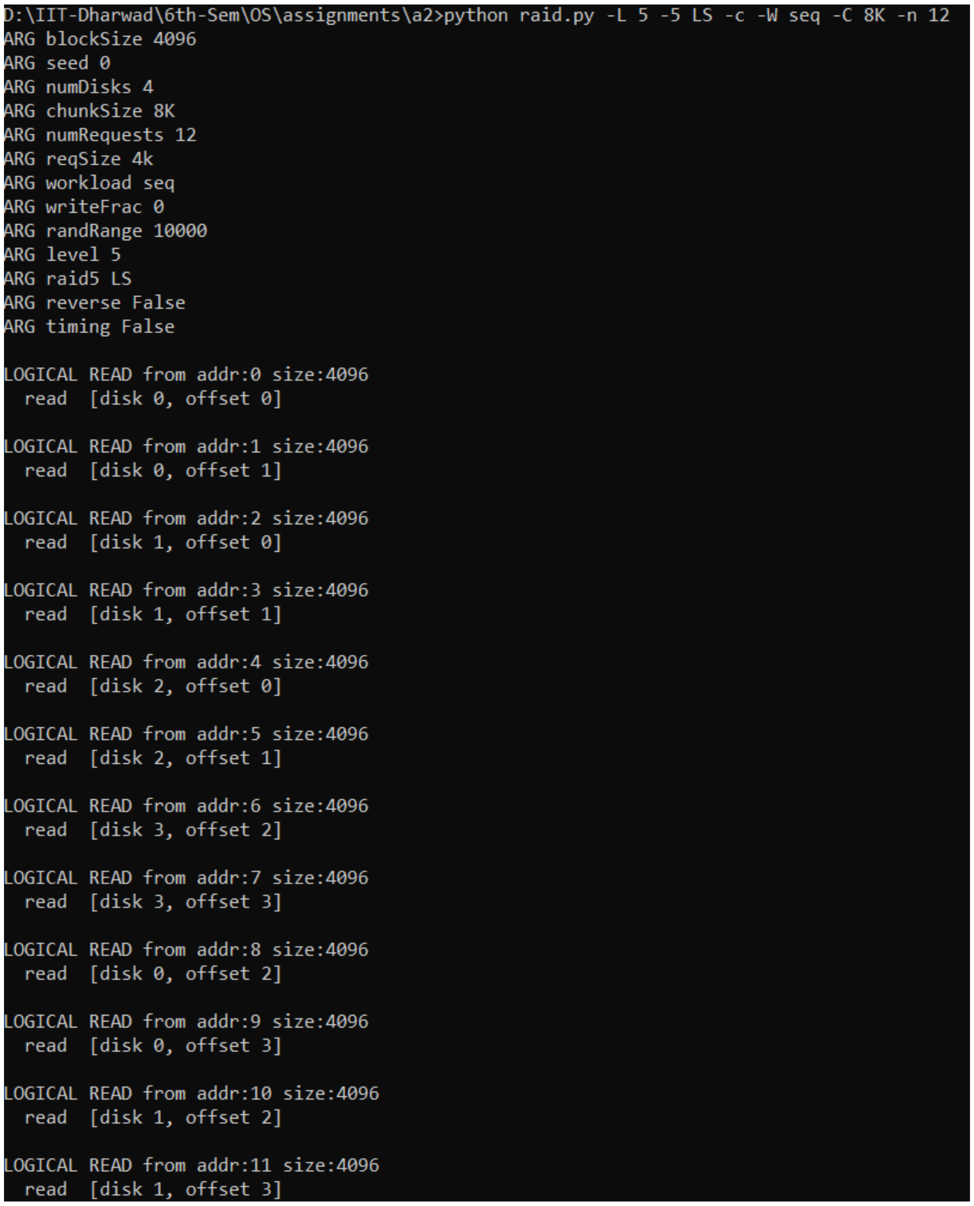
python raid.py -L 5 -5 LA -s 2 -c



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LA -s 2 -c
ARG blockSize 4096
ARG seed 2
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LA
ARG reverse False
ARG timing False

LOGICAL READ from addr:9560 size:4096
  read  [disk 3, offset 3186]

LOGICAL READ from addr:565 size:4096
  read  [disk 1, offset 188]

LOGICAL READ from addr:8354 size:4096
  read  [disk 2, offset 2784]

LOGICAL READ from addr:6697 size:4096
  read  [disk 1, offset 2232]

LOGICAL READ from addr:6059 size:4096
  read  [disk 3, offset 2019]

LOGICAL READ from addr:5812 size:4096
  read  [disk 1, offset 1937]

LOGICAL READ from addr:4306 size:4096
  read  [disk 2, offset 1435]

LOGICAL READ from addr:7230 size:4096
  read  [disk 0, offset 2410]

LOGICAL READ from addr:9493 size:4096
  read  [disk 1, offset 3164]

LOGICAL READ from addr:4448 size:4096
  read  [disk 3, offset 1482]
```

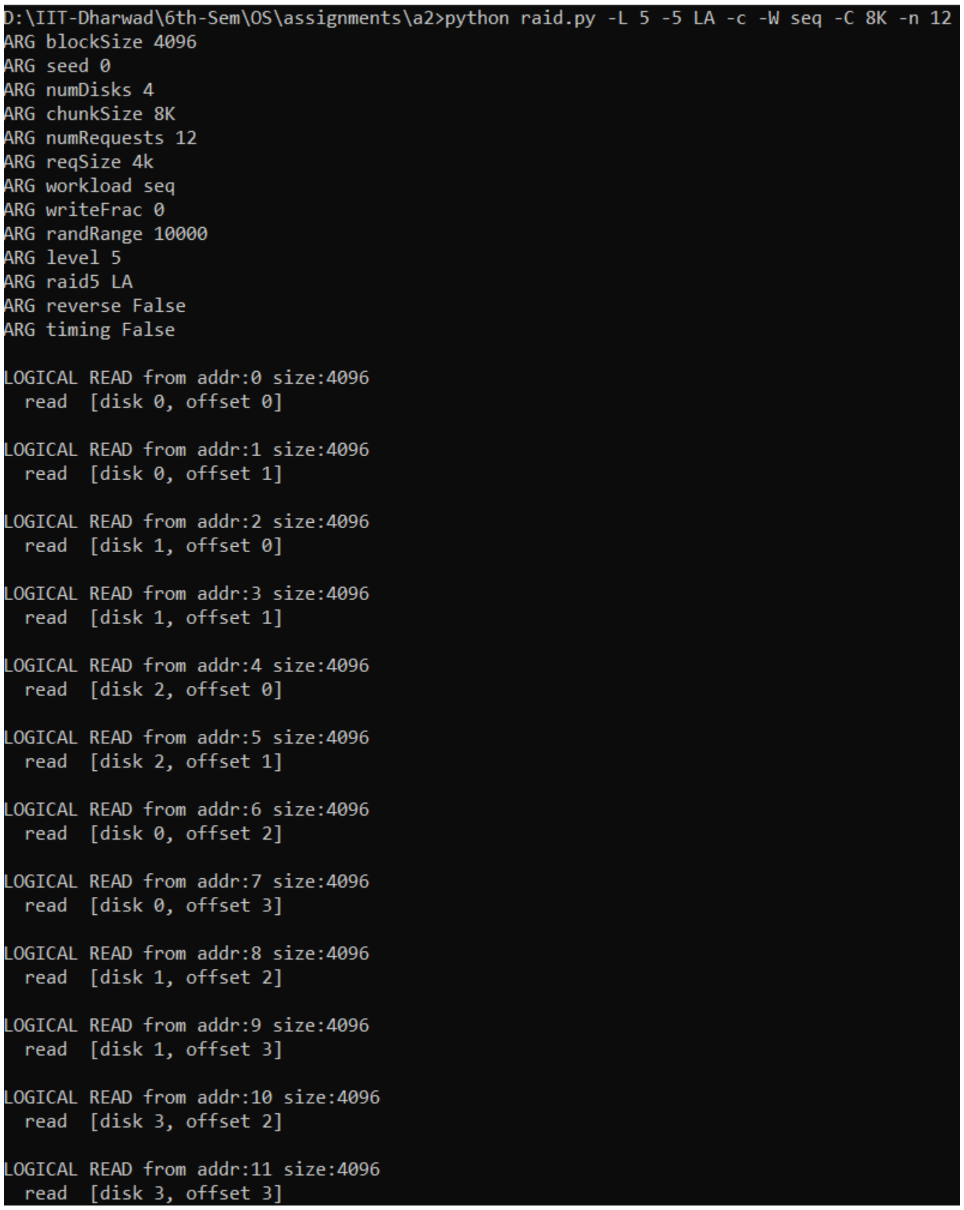
(2) Using a bigger chunk again puts multiple parity blocks in the same disks for level 5.

`python raid.py -L 5 -5 LS -c -W seq -C 8K -n 12`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LS -c -W seq -C 8K -n 12
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 8K
ARG numRequests 12
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:2 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 3, offset 3]

LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 0, offset 3]

LOGICAL READ from addr:10 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:11 size:4096
  read  [disk 1, offset 3]
```

python raid.py -L 5 -5 LA -c -W seq -C 8K -n 12

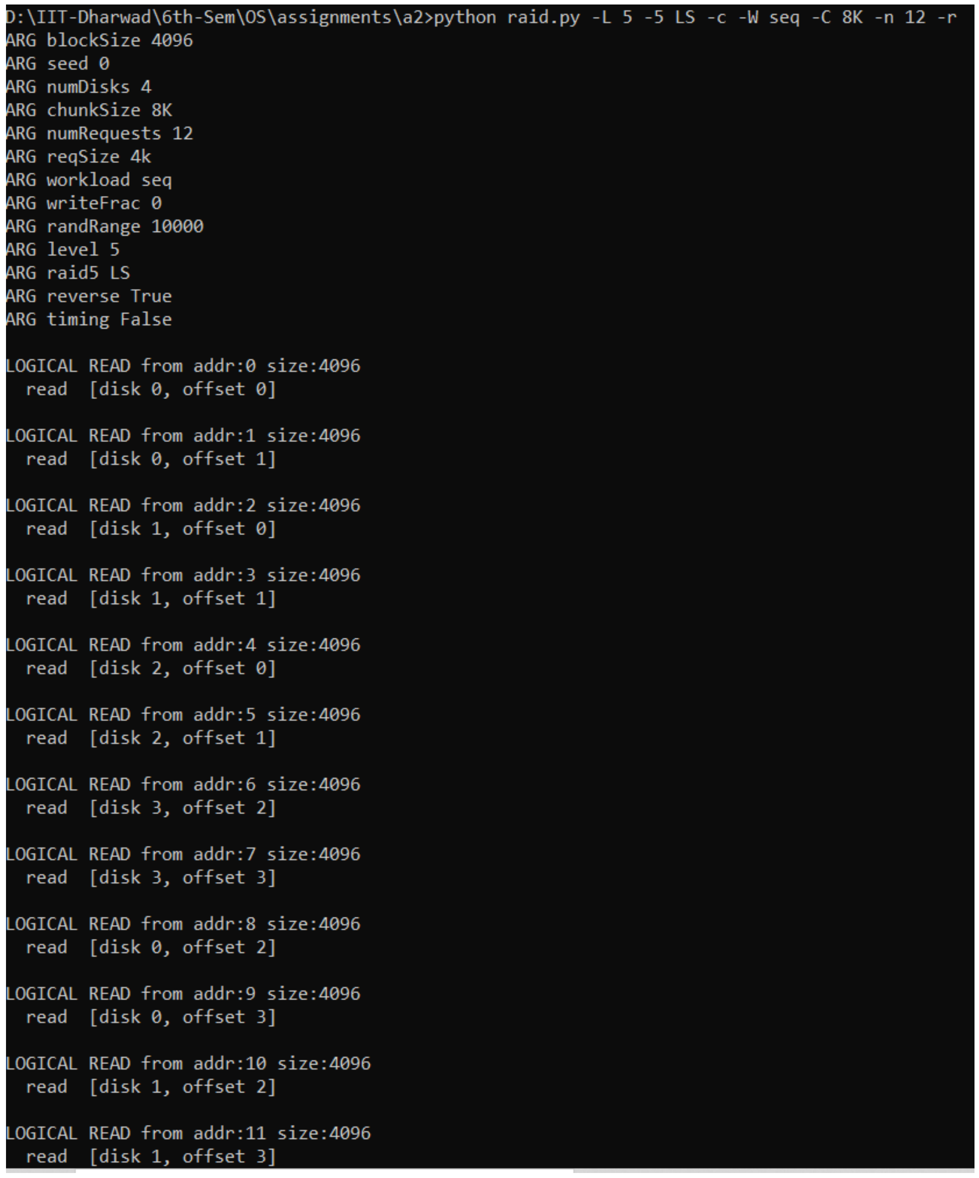


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LA -c -W seq -C 8K -n 12
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 8K
ARG numRequests 12
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LA
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:2 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 0, offset 3]

LOGICAL READ from addr:8 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]

LOGICAL READ from addr:10 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:11 size:4096
  read  [disk 3, offset 3]
```

From the above outputs and the simulation code we can deduce that using a bigger chunk puts multiple parity blocks in the same disks for level 5, here it is disk 3.

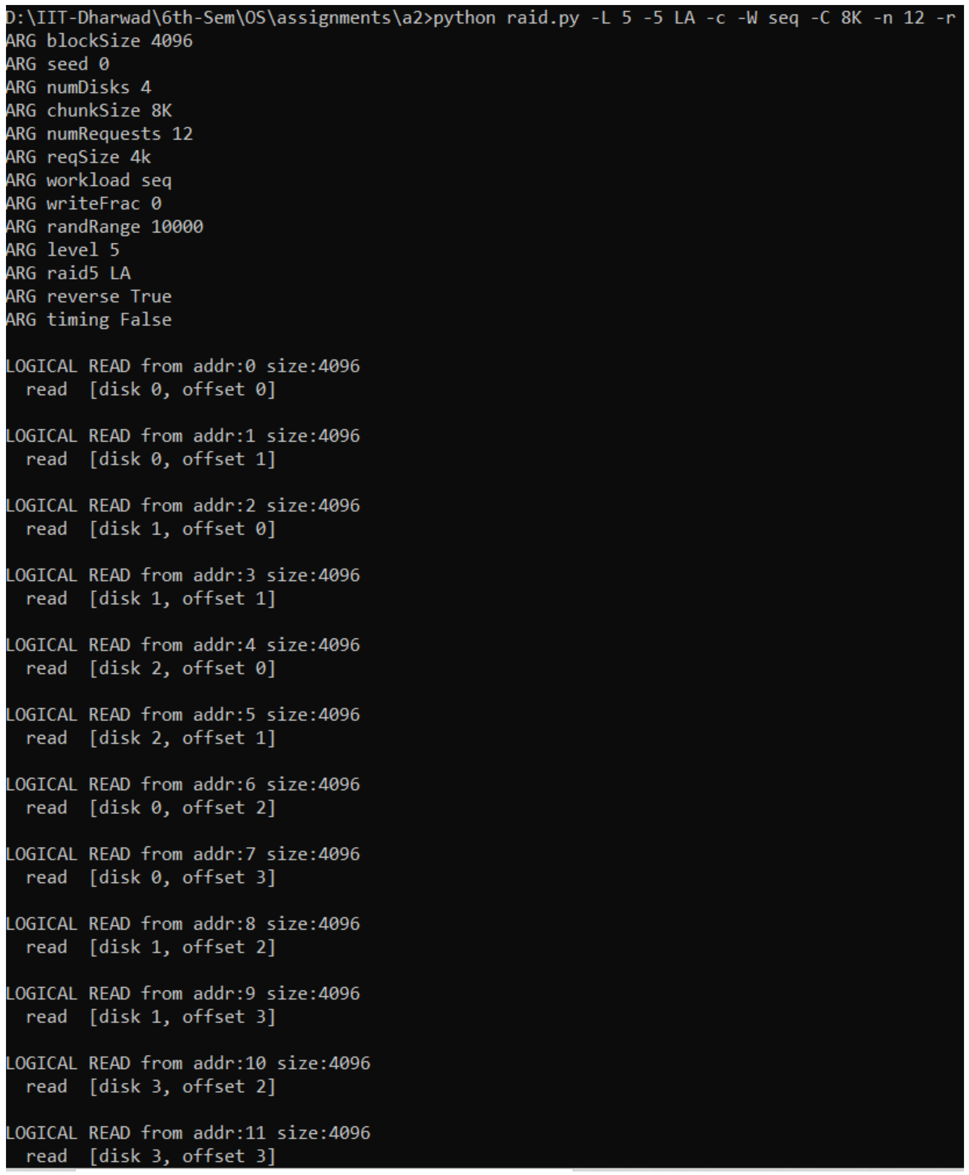
(3)

python raid.py -L 5 -5 LS -c -W seq -C 8K -n 12 -r



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LS -c -W seq -C 8K -n 12 -r
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 8K
ARG numRequests 12
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse True
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:2 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 3, offset 3]

LOGICAL READ from addr:8 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 0, offset 3]

LOGICAL READ from addr:10 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:11 size:4096
  read  [disk 1, offset 3]
```

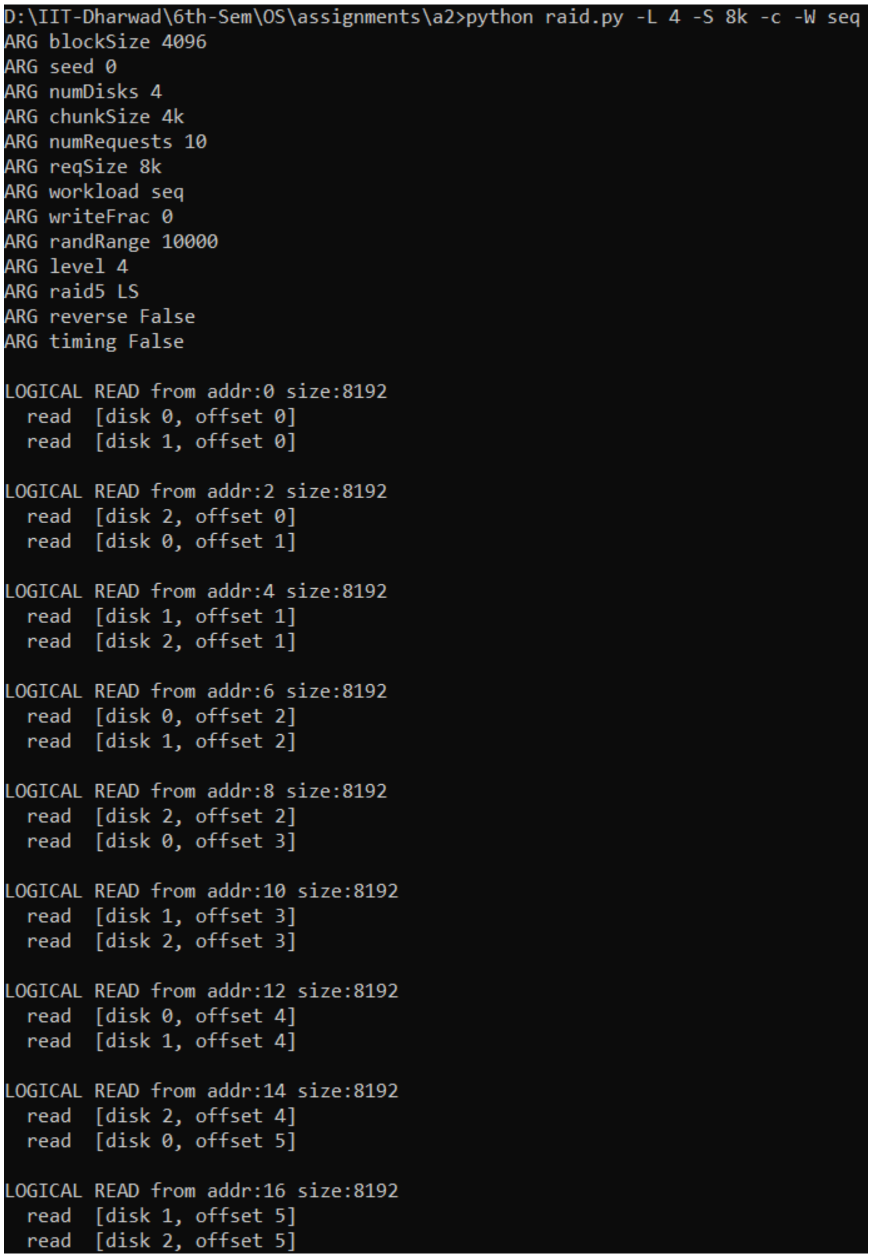
python raid.py -L 5 -5 LA -c -W seq -C 8K -n 12 -r



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -5 LA -c -W seq -C 8K -n 12 -r
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 8K
ARG numRequests 12
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LA
ARG reverse True
ARG timing False

LOGICAL READ from addr:0 size:4096
  read  [disk 0, offset 0]

LOGICAL READ from addr:1 size:4096
  read  [disk 0, offset 1]

LOGICAL READ from addr:2 size:4096
  read  [disk 1, offset 0]

LOGICAL READ from addr:3 size:4096
  read  [disk 1, offset 1]

LOGICAL READ from addr:4 size:4096
  read  [disk 2, offset 0]

LOGICAL READ from addr:5 size:4096
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:4096
  read  [disk 0, offset 2]

LOGICAL READ from addr:7 size:4096
  read  [disk 0, offset 3]

LOGICAL READ from addr:8 size:4096
  read  [disk 1, offset 2]

LOGICAL READ from addr:9 size:4096
  read  [disk 1, offset 3]

LOGICAL READ from addr:10 size:4096
  read  [disk 3, offset 2]

LOGICAL READ from addr:11 size:4096
  read  [disk 3, offset 3]
```

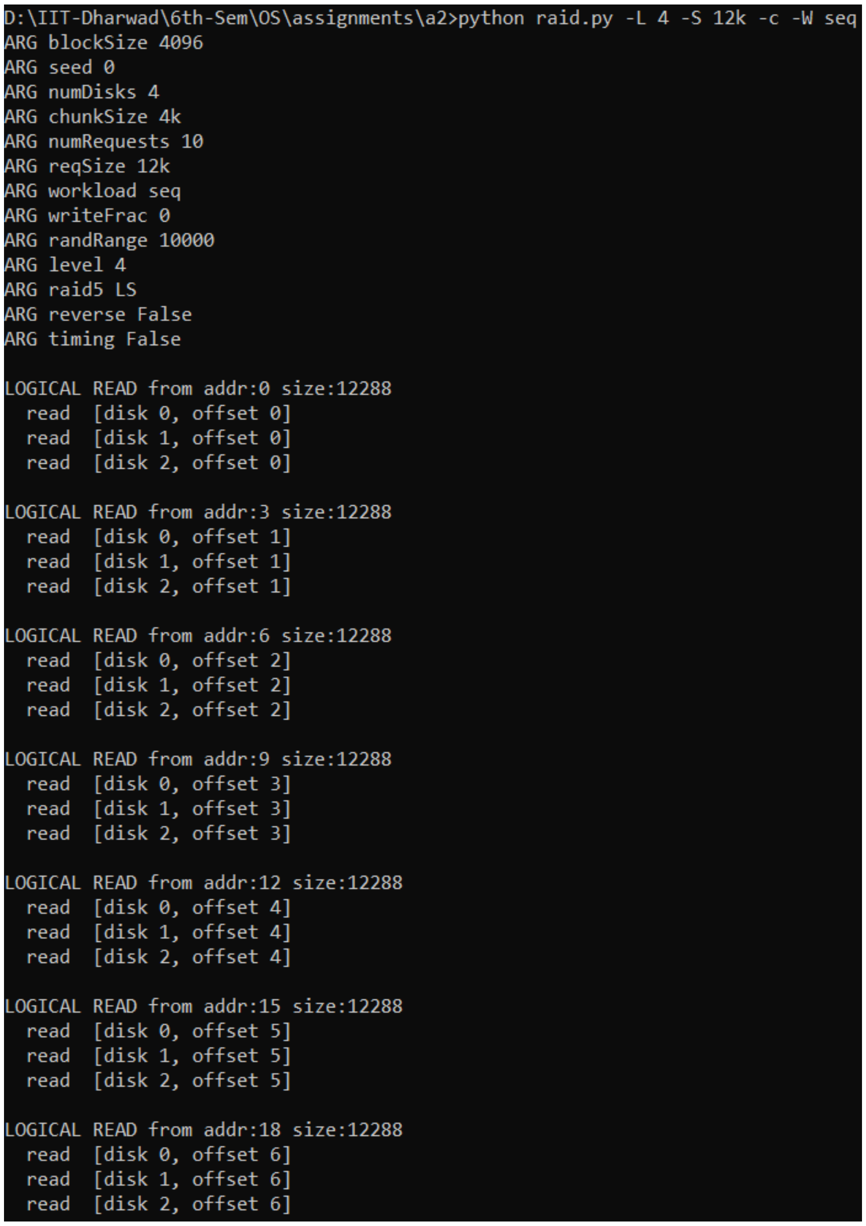
(4) For RAID 4, as we increase the request size the read occurs from the adjacent disks too. So, we have to make more disk accesses which thus increases the I/O time. The same effect is seen in the case of RAID 5 with an increase in request size. The number of disk accesses for each request increases as the data is stored in the next disk. RAID 4 and RAID 5 are much more efficient for large request sizes (like having all 4 disks) because the data reads are faster due to parity and striping. The data is generally divided among all disks so that it is more profitable. Here, for the 16k request size RAID 4 and RAID 5 are much more I/O efficient.

`python raid.py -L 4 -S 8k -c -W seq`



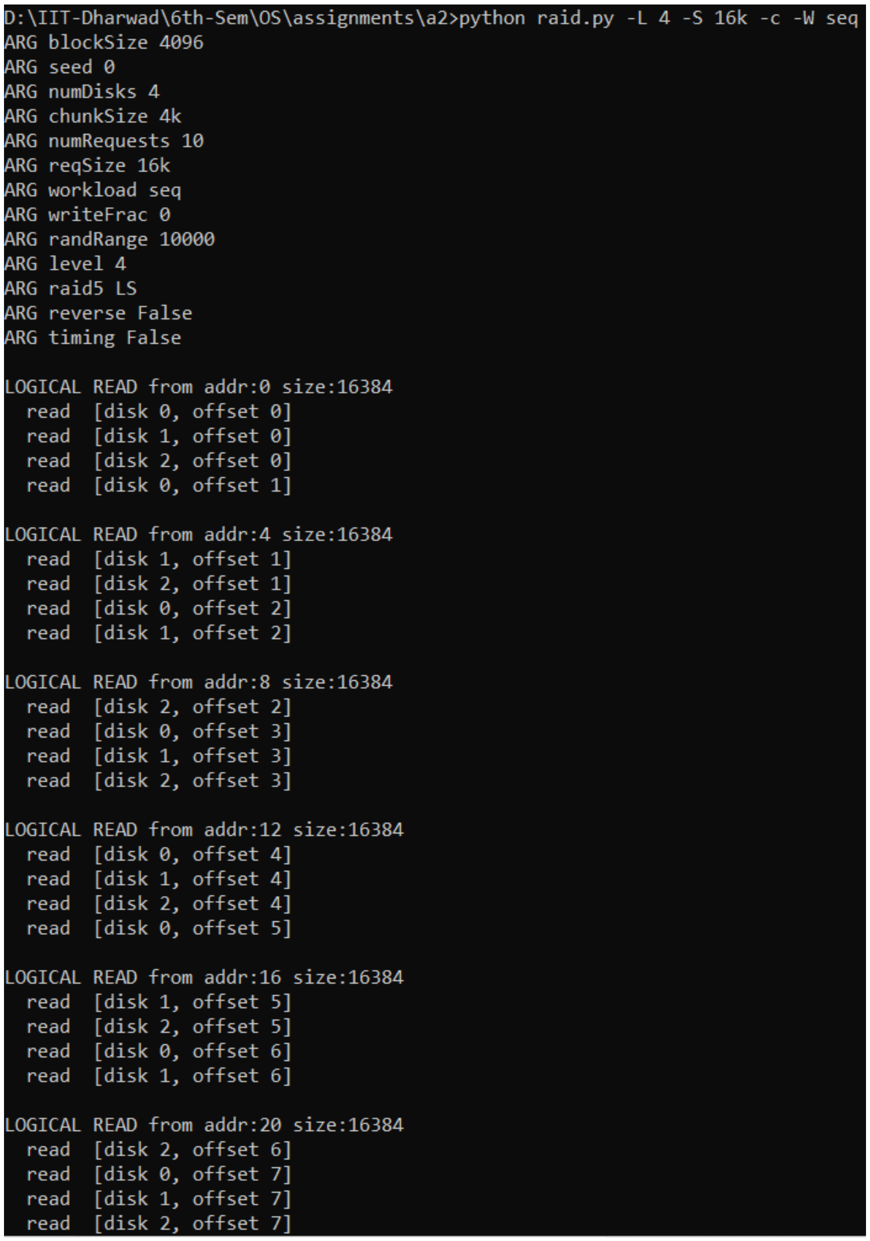
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -S 8k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 8k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:8192
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]

LOGICAL READ from addr:2 size:8192
  read  [disk 2, offset 0]
  read  [disk 0, offset 1]

LOGICAL READ from addr:4 size:8192
  read  [disk 1, offset 1]
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:8192
  read  [disk 0, offset 2]
  read  [disk 1, offset 2]

LOGICAL READ from addr:8 size:8192
  read  [disk 2, offset 2]
  read  [disk 0, offset 3]

LOGICAL READ from addr:10 size:8192
  read  [disk 1, offset 3]
  read  [disk 2, offset 3]

LOGICAL READ from addr:12 size:8192
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]

LOGICAL READ from addr:14 size:8192
  read  [disk 2, offset 4]
  read  [disk 0, offset 5]

LOGICAL READ from addr:16 size:8192
  read  [disk 1, offset 5]
  read  [disk 2, offset 5]
```

python raid.py -L 4 -S 12k -c -W seq



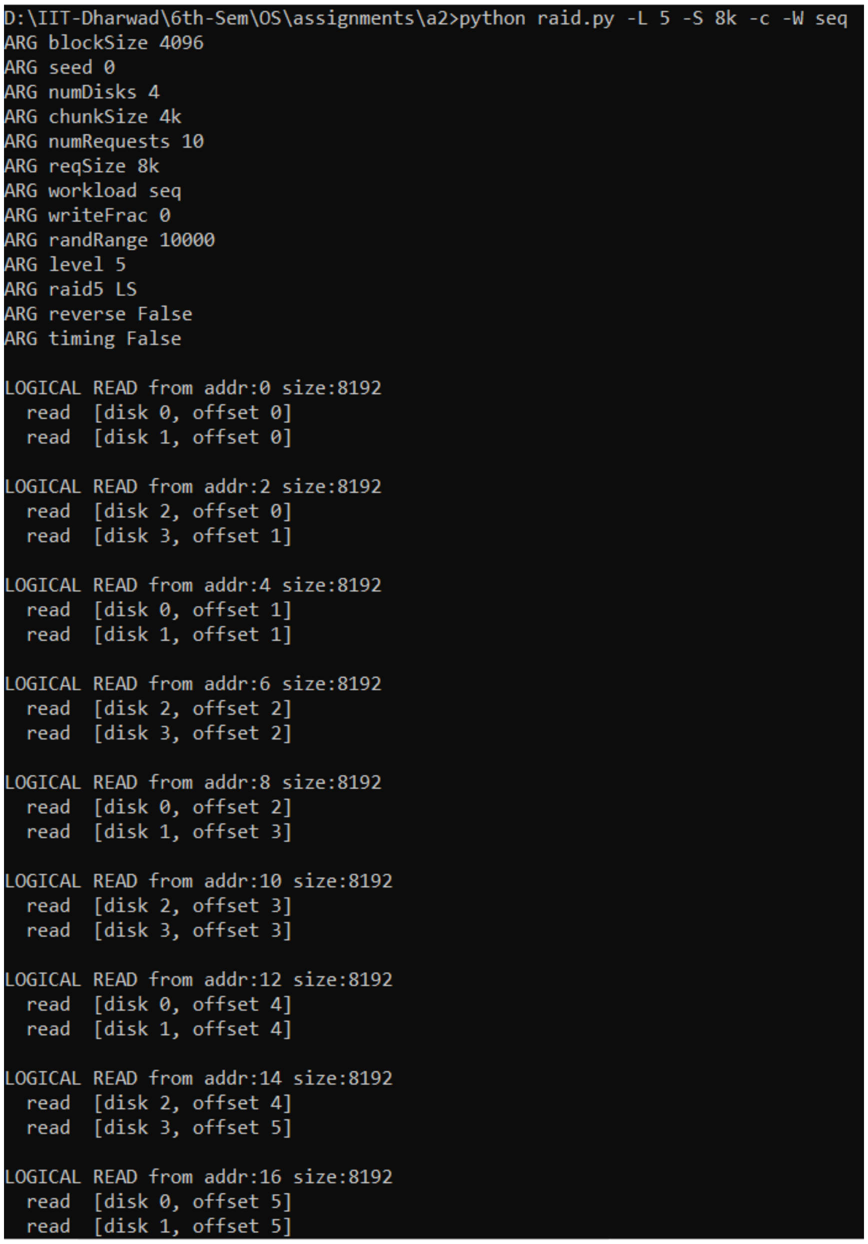
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -S 12k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 12k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:12288
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]
  read  [disk 2, offset 0]

LOGICAL READ from addr:3 size:12288
  read  [disk 0, offset 1]
  read  [disk 1, offset 1]
  read  [disk 2, offset 1]

LOGICAL READ from addr:6 size:12288
  read  [disk 0, offset 2]
  read  [disk 1, offset 2]
  read  [disk 2, offset 2]

LOGICAL READ from addr:9 size:12288
  read  [disk 0, offset 3]
  read  [disk 1, offset 3]
  read  [disk 2, offset 3]

LOGICAL READ from addr:12 size:12288
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]
  read  [disk 2, offset 4]

LOGICAL READ from addr:15 size:12288
  read  [disk 0, offset 5]
  read  [disk 1, offset 5]
  read  [disk 2, offset 5]

LOGICAL READ from addr:18 size:12288
  read  [disk 0, offset 6]
  read  [disk 1, offset 6]
  read  [disk 2, offset 6]
```

python raid.py -L 4 -S 16k -c -W seq



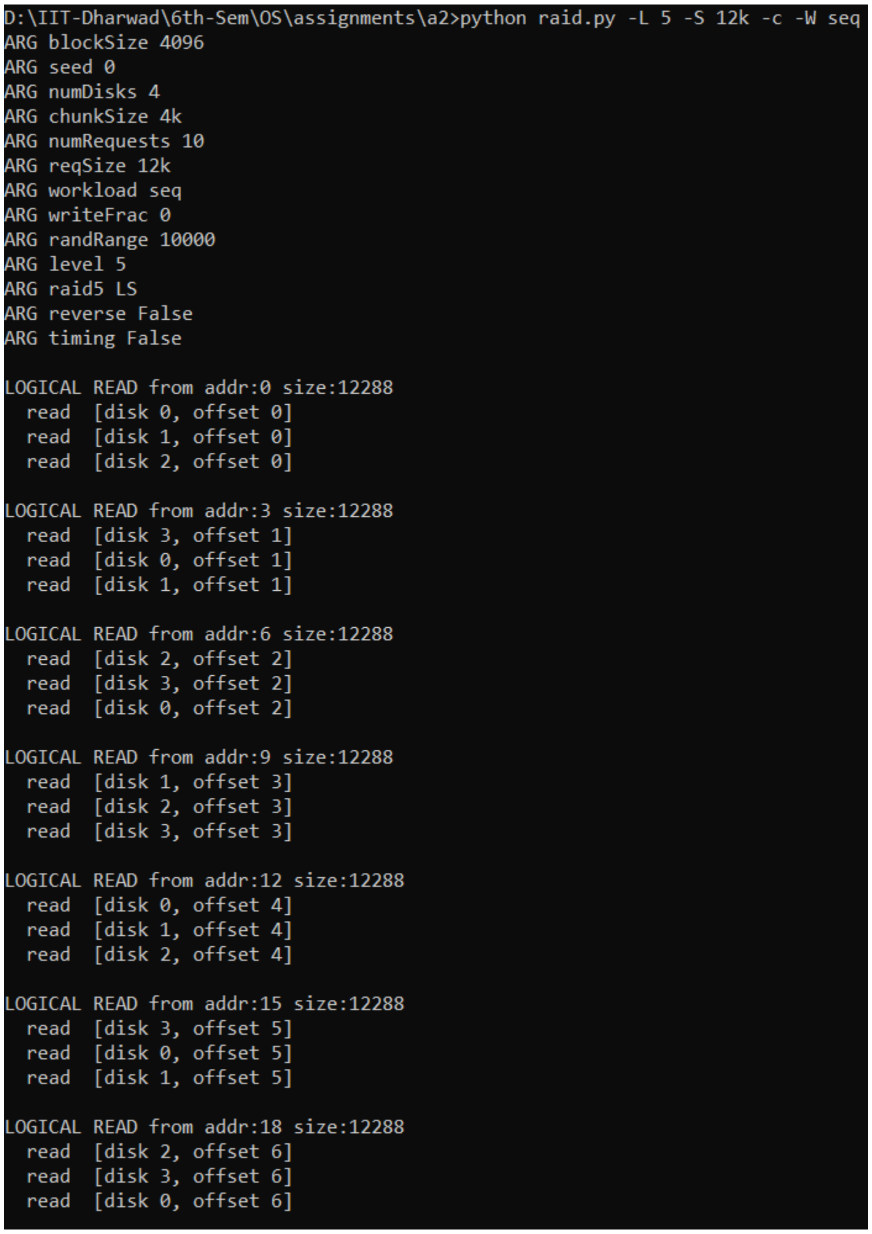
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -S 16k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 16k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:16384
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]
  read  [disk 2, offset 0]
  read  [disk 0, offset 1]

LOGICAL READ from addr:4 size:16384
  read  [disk 1, offset 1]
  read  [disk 2, offset 1]
  read  [disk 0, offset 2]
  read  [disk 1, offset 2]

LOGICAL READ from addr:8 size:16384
  read  [disk 2, offset 2]
  read  [disk 0, offset 3]
  read  [disk 1, offset 3]
  read  [disk 2, offset 3]

LOGICAL READ from addr:12 size:16384
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]
  read  [disk 2, offset 4]
  read  [disk 0, offset 5]

LOGICAL READ from addr:16 size:16384
  read  [disk 1, offset 5]
  read  [disk 2, offset 5]
  read  [disk 0, offset 6]
  read  [disk 1, offset 6]

LOGICAL READ from addr:20 size:16384
  read  [disk 2, offset 6]
  read  [disk 0, offset 7]
  read  [disk 1, offset 7]
  read  [disk 2, offset 7]
```

python raid.py -L 5 -S 8k -c -W seq



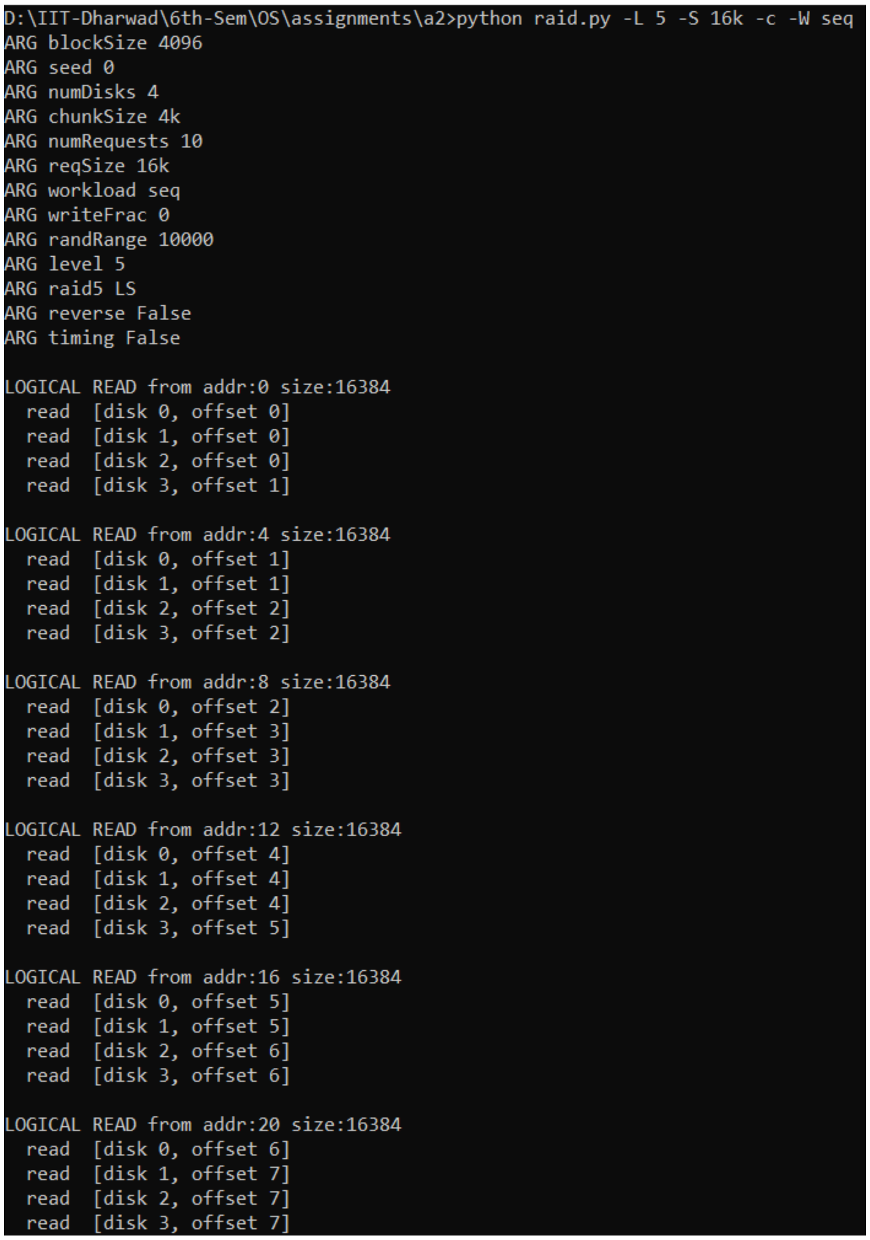
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -S 8k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 8k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:8192
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]

LOGICAL READ from addr:2 size:8192
  read  [disk 2, offset 0]
  read  [disk 3, offset 1]

LOGICAL READ from addr:4 size:8192
  read  [disk 0, offset 1]
  read  [disk 1, offset 1]

LOGICAL READ from addr:6 size:8192
  read  [disk 2, offset 2]
  read  [disk 3, offset 2]

LOGICAL READ from addr:8 size:8192
  read  [disk 0, offset 2]
  read  [disk 1, offset 3]

LOGICAL READ from addr:10 size:8192
  read  [disk 2, offset 3]
  read  [disk 3, offset 3]

LOGICAL READ from addr:12 size:8192
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]

LOGICAL READ from addr:14 size:8192
  read  [disk 2, offset 4]
  read  [disk 3, offset 5]

LOGICAL READ from addr:16 size:8192
  read  [disk 0, offset 5]
  read  [disk 1, offset 5]
```

python raid.py -L 5 -S 12k -c -W seq



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -S 12k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 12k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:12288
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]
  read  [disk 2, offset 0]

LOGICAL READ from addr:3 size:12288
  read  [disk 3, offset 1]
  read  [disk 0, offset 1]
  read  [disk 1, offset 1]

LOGICAL READ from addr:6 size:12288
  read  [disk 2, offset 2]
  read  [disk 3, offset 2]
  read  [disk 0, offset 2]

LOGICAL READ from addr:9 size:12288
  read  [disk 1, offset 3]
  read  [disk 2, offset 3]
  read  [disk 3, offset 3]

LOGICAL READ from addr:12 size:12288
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]
  read  [disk 2, offset 4]

LOGICAL READ from addr:15 size:12288
  read  [disk 3, offset 5]
  read  [disk 0, offset 5]
  read  [disk 1, offset 5]

LOGICAL READ from addr:18 size:12288
  read  [disk 2, offset 6]
  read  [disk 3, offset 6]
  read  [disk 0, offset 6]
```

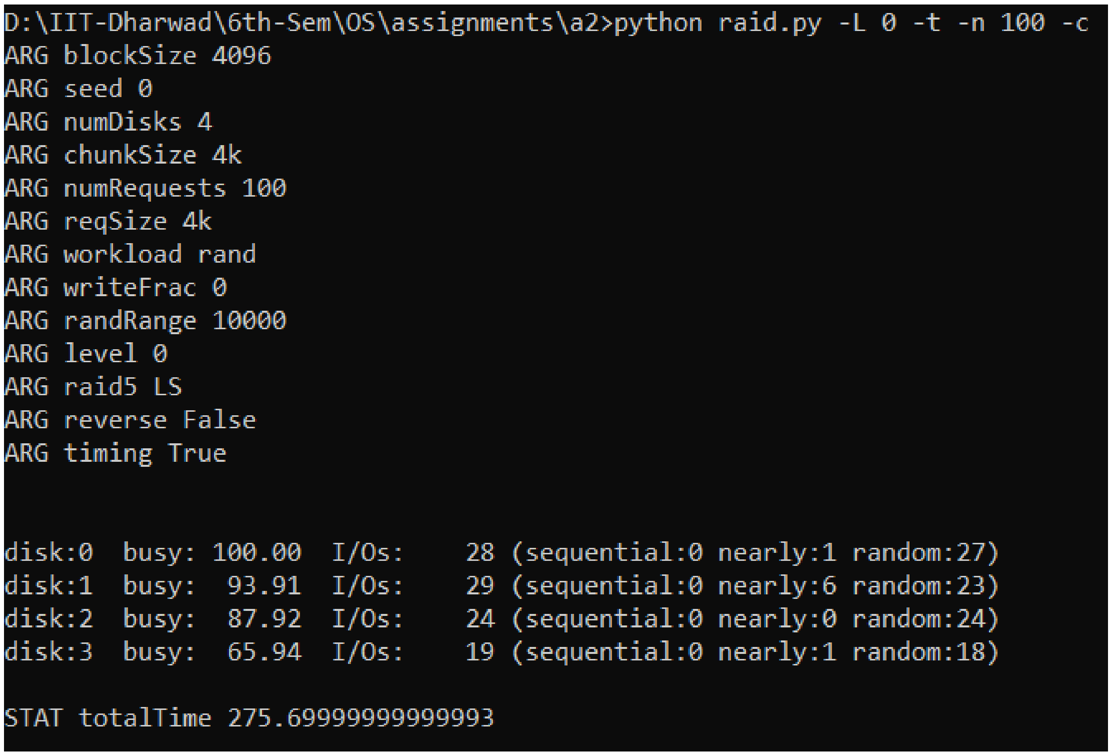
python raid.py -L 5 -S 16k -c -W seq



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -S 16k -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 10
ARG reqSize 16k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing False

LOGICAL READ from addr:0 size:16384
  read  [disk 0, offset 0]
  read  [disk 1, offset 0]
  read  [disk 2, offset 0]
  read  [disk 3, offset 1]

LOGICAL READ from addr:4 size:16384
  read  [disk 0, offset 1]
  read  [disk 1, offset 1]
  read  [disk 2, offset 2]
  read  [disk 3, offset 2]

LOGICAL READ from addr:8 size:16384
  read  [disk 0, offset 2]
  read  [disk 1, offset 3]
  read  [disk 2, offset 3]
  read  [disk 3, offset 3]

LOGICAL READ from addr:12 size:16384
  read  [disk 0, offset 4]
  read  [disk 1, offset 4]
  read  [disk 2, offset 4]
  read  [disk 3, offset 5]

LOGICAL READ from addr:16 size:16384
  read  [disk 0, offset 5]
  read  [disk 1, offset 5]
  read  [disk 2, offset 6]
  read  [disk 3, offset 6]

LOGICAL READ from addr:20 size:16384
  read  [disk 0, offset 6]
  read  [disk 1, offset 7]
  read  [disk 2, offset 7]
  read  [disk 3, offset 7]
```

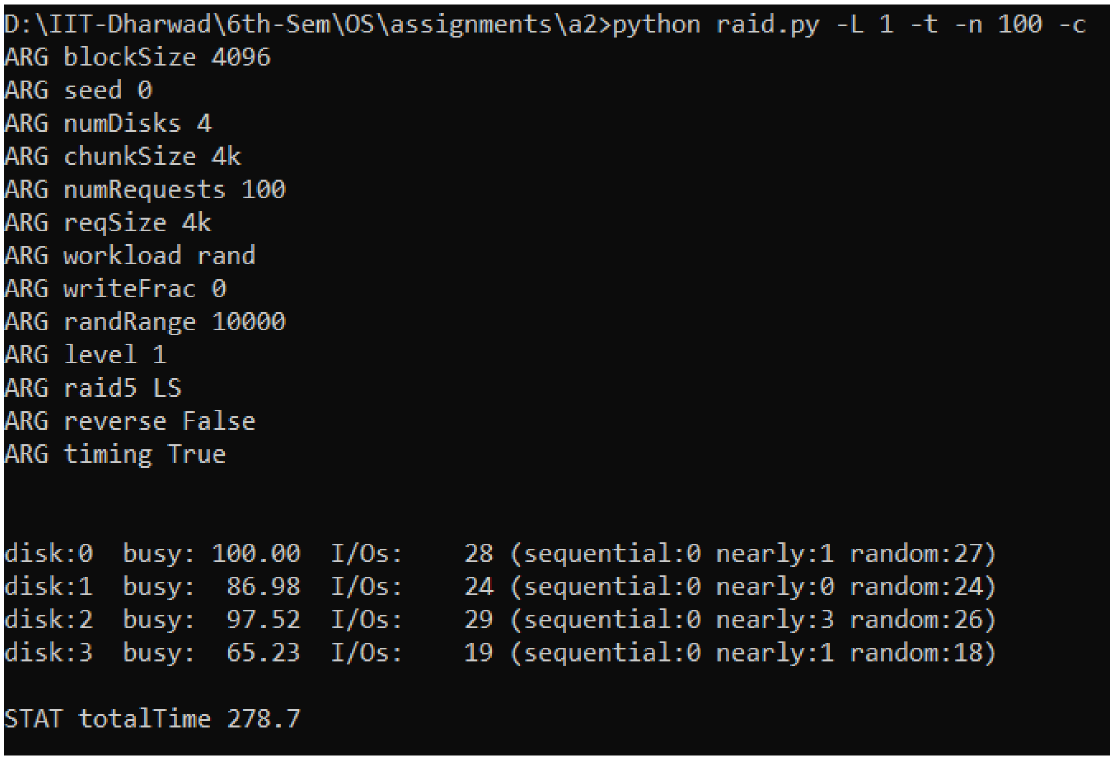
(5)

python raid.py -L 0 -t -n 100 -c



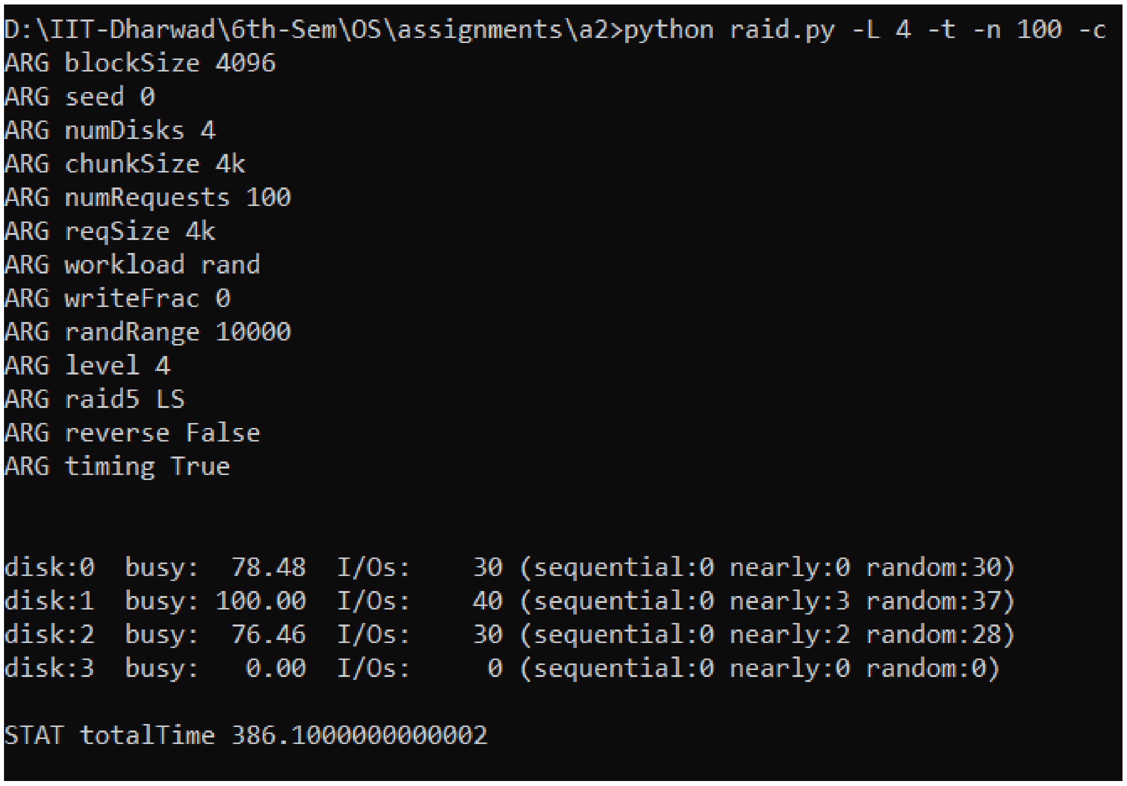
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:1  busy:  93.91  I/Os:    29 (sequential:0 nearly:6 random:23)
disk:2  busy:  87.92  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:3  busy:  65.94  I/Os:    19 (sequential:0 nearly:1 random:18)

STAT totalTime 275.69999999999993
```

python raid.py -L 1 -t -n 100 -c



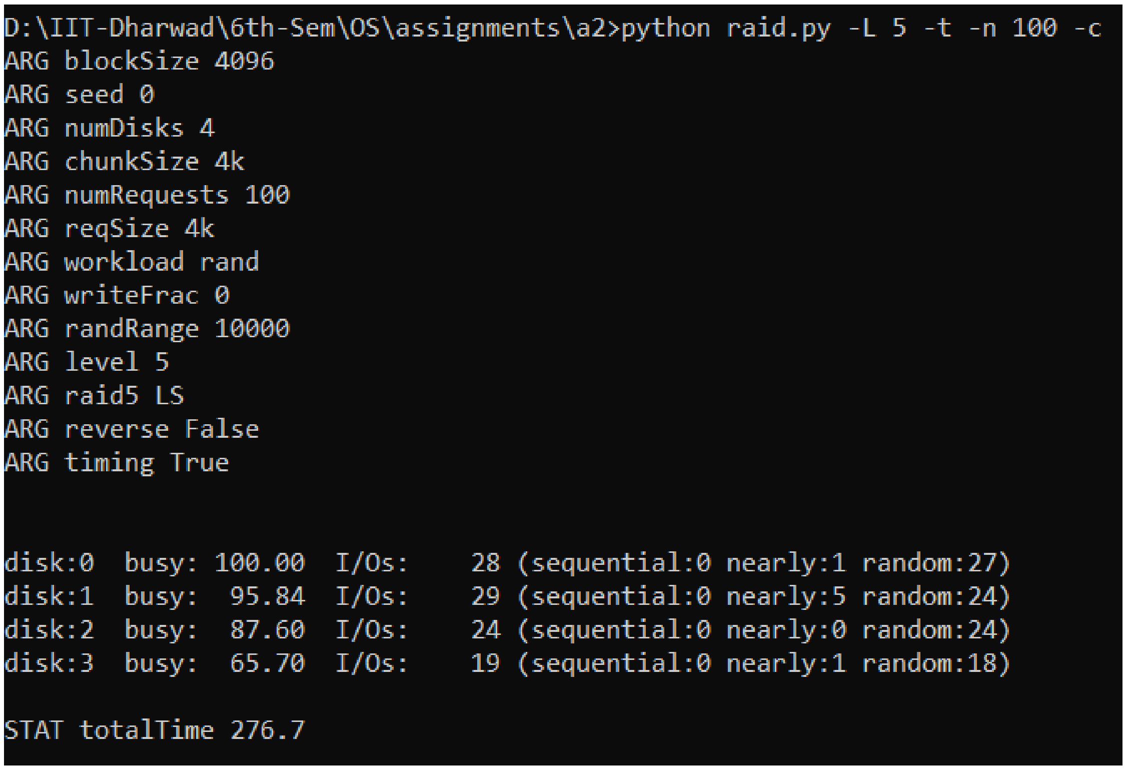
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:1  busy:  86.98  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:2  busy:  97.52  I/Os:    29 (sequential:0 nearly:3 random:26)
disk:3  busy:  65.23  I/Os:    19 (sequential:0 nearly:1 random:18)

STAT totalTime 278.7
```

python raid.py -L 4 -t -n 100 -c



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  78.48  I/Os:    30 (sequential:0 nearly:0 random:30)
disk:1  busy: 100.00  I/Os:    40 (sequential:0 nearly:3 random:37)
disk:2  busy:  76.46  I/Os:    30 (sequential:0 nearly:2 random:28)
disk:3  busy:   0.00  I/Os:     0 (sequential:0 nearly:0 random:0)

STAT totalTime 386.1000000000002
```

python raid.py -L 5 -t -n 100 -c



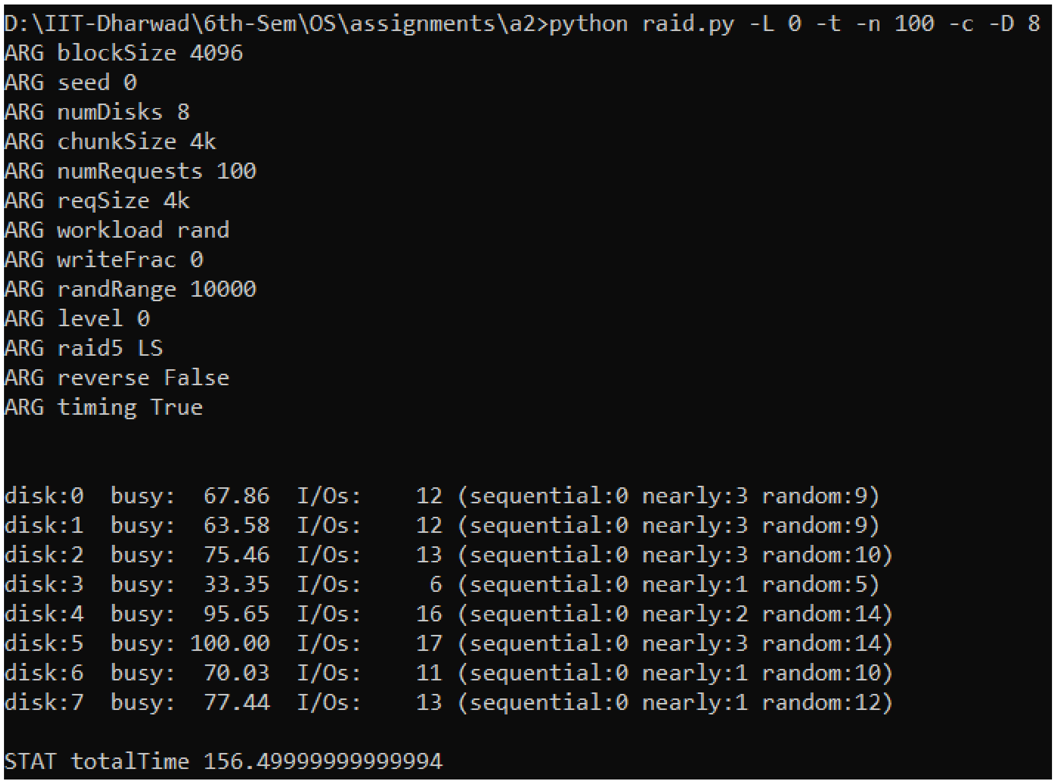
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:1  busy:  95.84  I/Os:    29 (sequential:0 nearly:5 random:24)
disk:2  busy:  87.60  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:3  busy:  65.70  I/Os:    19 (sequential:0 nearly:1 random:18)

STAT totalTime 276.7
```

From the above outputs, we can see that the total times are:
RAID 0 : 275.7
RAID 1 : 278.7
RAID 4 : 386.1
RAID 5 : 276.7

Thus, RAID 0 is the fastest among others as expected.
RAID 5 is faster than RAID 4.
RAID 5 has faster writes than RAID 1 so RAID 5 is a little better than RAID 1.

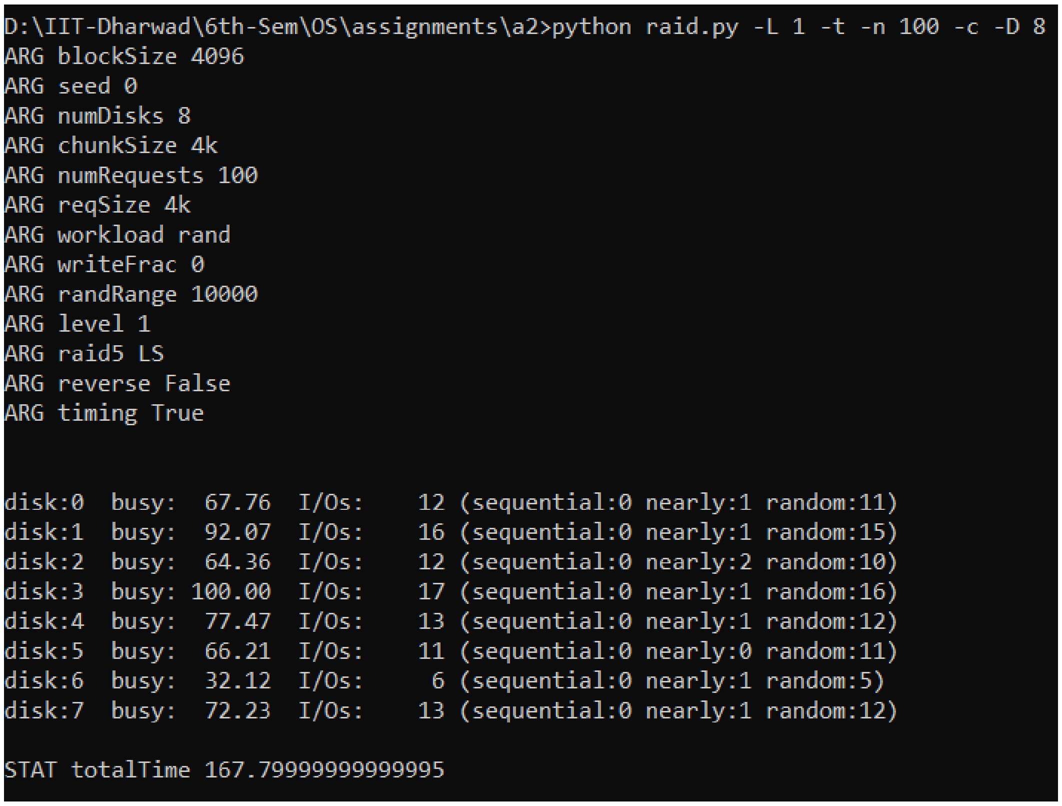
(6)

`python raid.py -L 0 -t -n 100 -c -D 8`



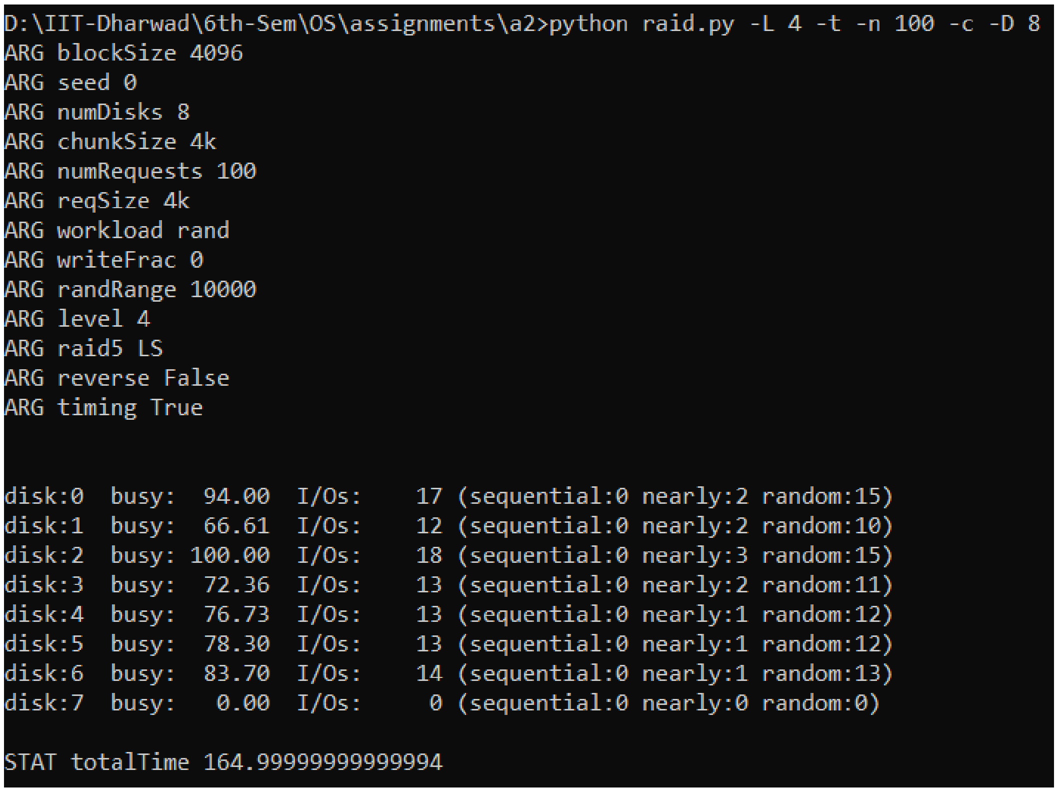
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c -D 8
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  67.86  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:1  busy:  63.58  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:2  busy:  75.46  I/Os:    13 (sequential:0 nearly:3 random:10)
disk:3  busy:  33.35  I/Os:     6 (sequential:0 nearly:1 random:5)
disk:4  busy:  95.65  I/Os:    16 (sequential:0 nearly:2 random:14)
disk:5  busy: 100.00  I/Os:    17 (sequential:0 nearly:3 random:14)
disk:6  busy:  70.03  I/Os:    11 (sequential:0 nearly:1 random:10)
disk:7  busy:  77.44  I/Os:    13 (sequential:0 nearly:1 random:12)

STAT totalTime 156.49999999999994
```

`python raid.py -L 1 -t -n 100 -c -D 8`

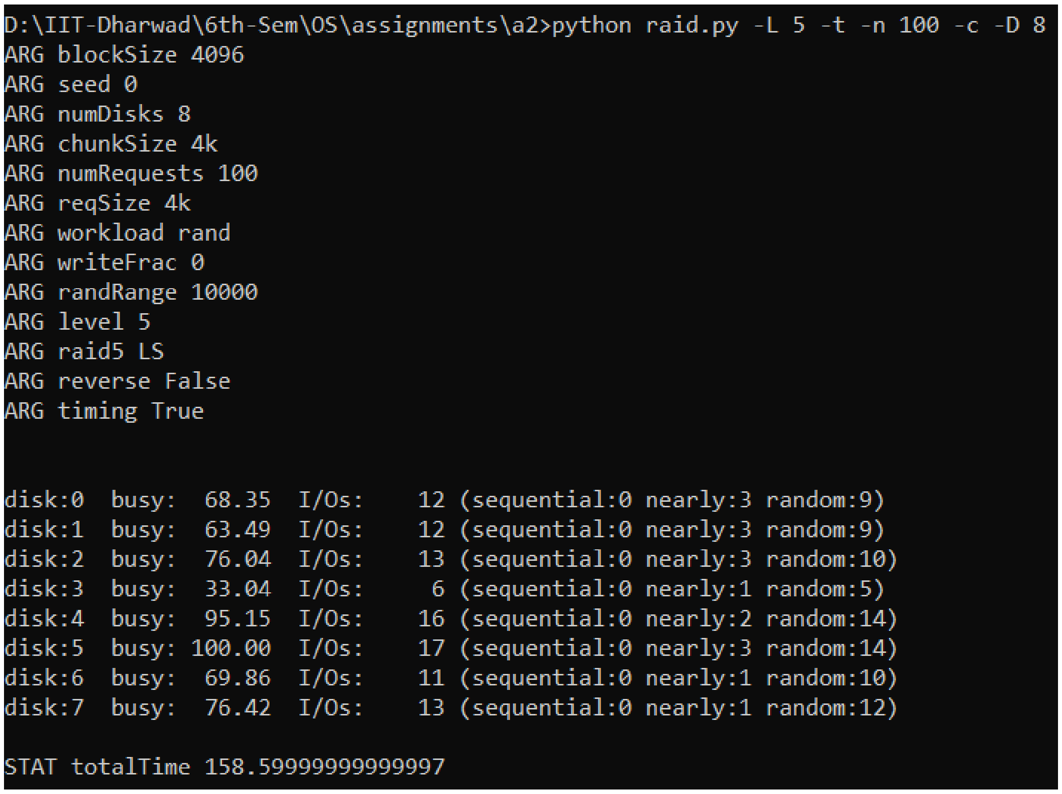


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c -D 8
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  67.76  I/Os:    12 (sequential:0 nearly:1 random:11)
disk:1  busy:  92.07  I/Os:    16 (sequential:0 nearly:1 random:15)
disk:2  busy:  64.36  I/Os:    12 (sequential:0 nearly:2 random:10)
disk:3  busy: 100.00  I/Os:    17 (sequential:0 nearly:1 random:16)
disk:4  busy:  77.47  I/Os:    13 (sequential:0 nearly:1 random:12)
disk:5  busy:  66.21  I/Os:    11 (sequential:0 nearly:0 random:11)
disk:6  busy:  32.12  I/Os:     6 (sequential:0 nearly:1 random:5)
disk:7  busy:  72.23  I/Os:    13 (sequential:0 nearly:1 random:12)

STAT totalTime 167.79999999999995
```

python raid.py -L 4 -t -n 100 -c -D 8


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c -D 8
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  94.00  I/Os:    17 (sequential:0 nearly:2 random:15)
disk:1  busy:  66.61  I/Os:    12 (sequential:0 nearly:2 random:10)
disk:2  busy: 100.00  I/Os:    18 (sequential:0 nearly:3 random:15)
disk:3  busy:  72.36  I/Os:    13 (sequential:0 nearly:2 random:11)
disk:4  busy:  76.73  I/Os:    13 (sequential:0 nearly:1 random:12)
disk:5  busy:  78.30  I/Os:    13 (sequential:0 nearly:1 random:12)
disk:6  busy:  83.70  I/Os:    14 (sequential:0 nearly:1 random:13)
disk:7  busy:   0.00  I/Os:     0 (sequential:0 nearly:0 random:0)

STAT totalTime 164.99999999999994
```

python raid.py -L 5 -t -n 100 -c -D 8

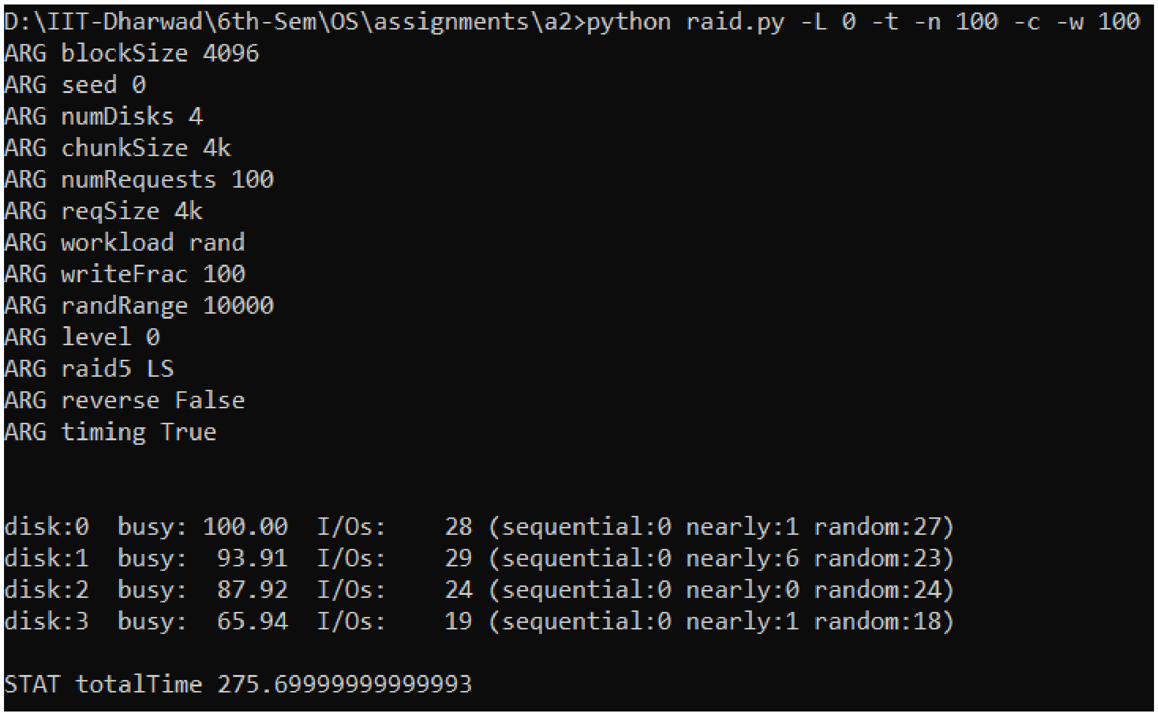

```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c -D 8
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  68.35  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:1  busy:  63.49  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:2  busy:  76.04  I/Os:    13 (sequential:0 nearly:3 random:10)
disk:3  busy:  33.04  I/Os:     6 (sequential:0 nearly:1 random:5)
disk:4  busy:  95.15  I/Os:    16 (sequential:0 nearly:2 random:14)
disk:5  busy: 100.00  I/Os:    17 (sequential:0 nearly:3 random:14)
disk:6  busy:  69.86  I/Os:    11 (sequential:0 nearly:1 random:10)
disk:7  busy:  76.42  I/Os:    13 (sequential:0 nearly:1 random:12)

STAT totalTime 158.59999999999997
```

RAID Level	**Performance Scale**
0	275.7 / 156.5 = 1.76
1	278.7 / 167.8 = 1.66
4	386.1 / 165.0 = 2.34
5	276.5 / 158.6 = 1.74

Here, only level 4 scales differently whereas all the others levels scale similarly.

(7)

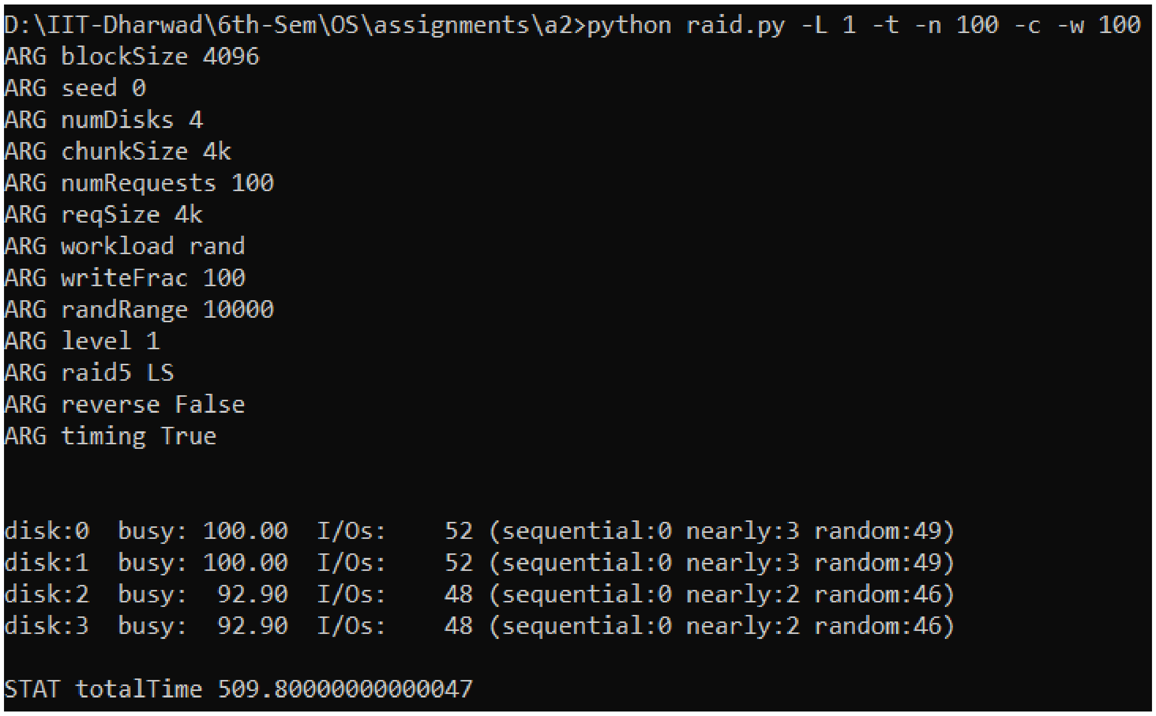
`python raid.py -L 0 -t -n 100 -c -w 100`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:1  busy:  93.91  I/Os:    29 (sequential:0 nearly:6 random:23)
disk:2  busy:  87.92  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:3  busy:  65.94  I/Os:    19 (sequential:0 nearly:1 random:18)

STAT totalTime 275.69999999999993
```

To do a rough estimate of the time taken we have 100 accesses and 4 disks. So, let us take an average of 10 units of time for each write then the estimate becomes 100*10/4 equals 250 that is a nearest value to 275.7.

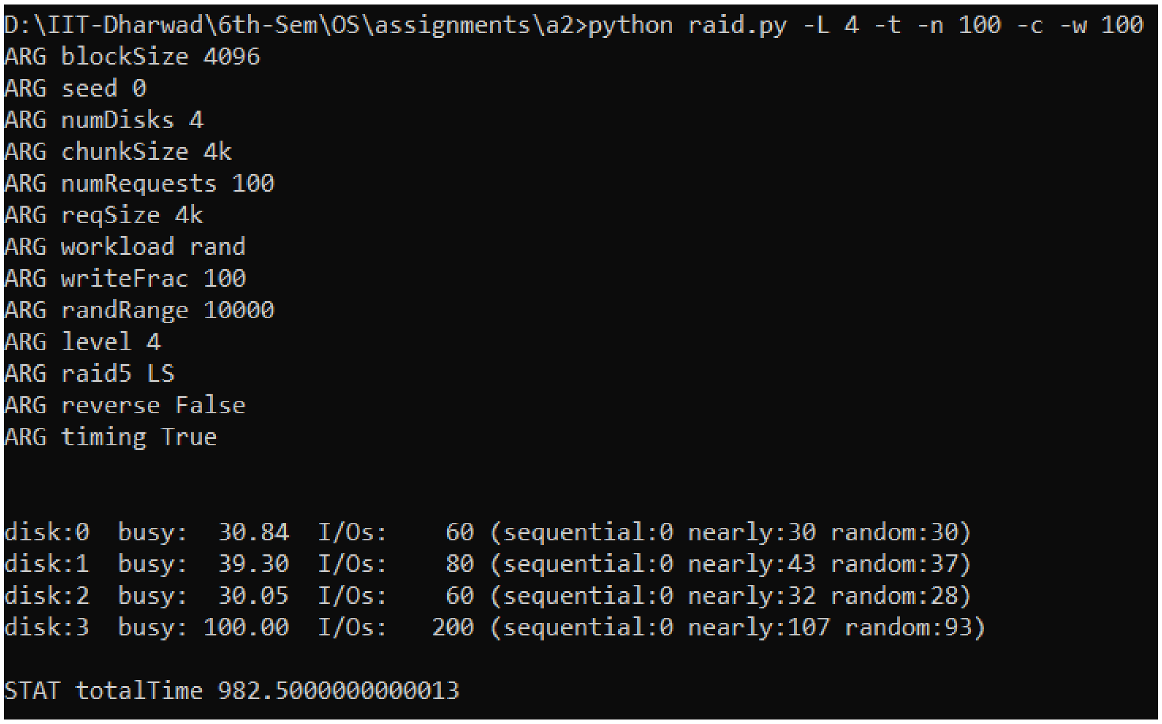
python raid.py -L 1 -t -n 100 -c -w 100



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    52 (sequential:0 nearly:3 random:49)
disk:1  busy: 100.00  I/Os:    52 (sequential:0 nearly:3 random:49)
disk:2  busy:  92.90  I/Os:    48 (sequential:0 nearly:2 random:46)
disk:3  busy:  92.90  I/Os:    48 (sequential:0 nearly:2 random:46)

STAT totalTime 509.80000000000047
```

To do a rough estimate of the time taken we have 100 accesses and 4 disks (since this is RAID 1). So, let us take an average of 10 units of time for each write then the estimate becomes 100*10 / (4/2) equals 500 that is a nearest estimate to 509.8.

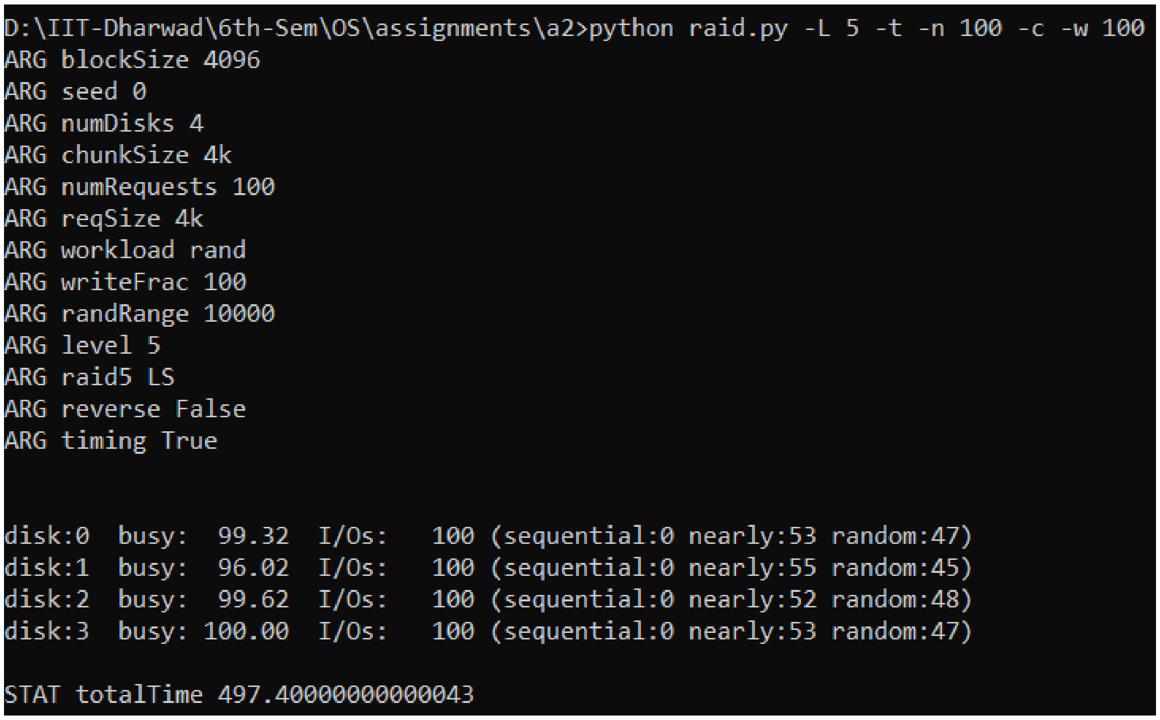
python raid.py -L 4 -t -n 100 -c -w 100



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  30.84  I/Os:    60 (sequential:0 nearly:30 random:30)
disk:1  busy:  39.30  I/Os:    80 (sequential:0 nearly:43 random:37)
disk:2  busy:  30.05  I/Os:    60 (sequential:0 nearly:32 random:28)
disk:3  busy: 100.00  I/Os:   200 (sequential:0 nearly:107 random:93)

STAT totalTime 982.5000000000013
```

Here we have parity involved and the writes are expensive so the estimation cannot be done linearly as above.

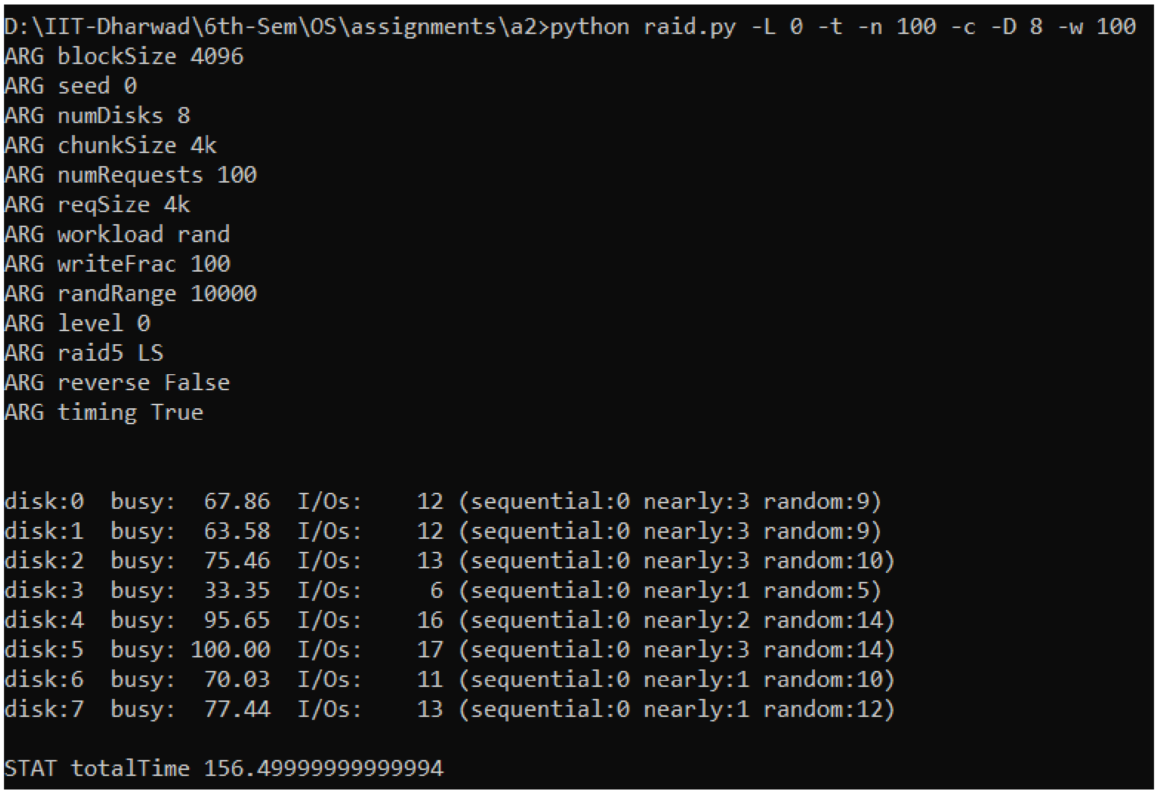
python raid.py -L 5 -t -n 100 -c -w 100



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  99.32  I/Os:   100 (sequential:0 nearly:53 random:47)
disk:1  busy:  96.02  I/Os:   100 (sequential:0 nearly:55 random:45)
disk:2  busy:  99.62  I/Os:   100 (sequential:0 nearly:52 random:48)
disk:3  busy: 100.00  I/Os:   100 (sequential:0 nearly:53 random:47)

STAT totalTime 497.40000000000043
```

Here we have parity involved and the writes are expensive so the estimation cannot be done linearly as above.

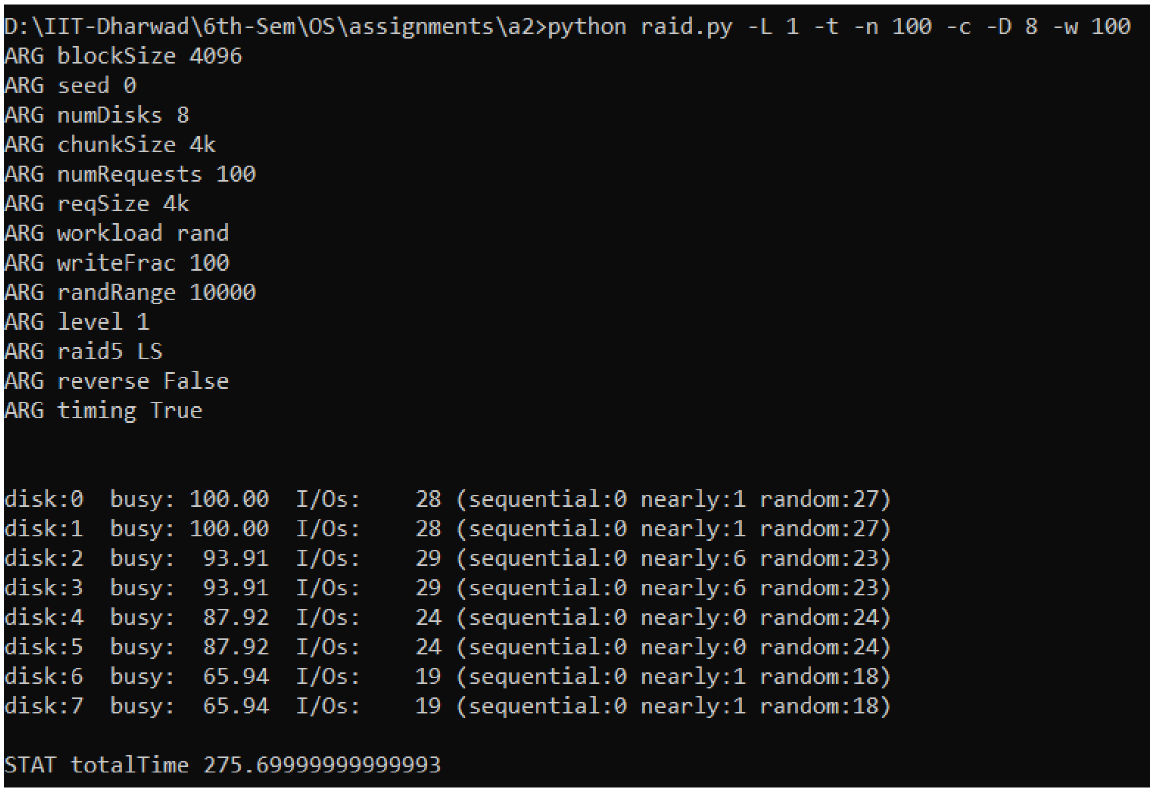
`python raid.py -L 0 -t -n 100 -c -D 8 -w 100`



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c -D 8 -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  67.86  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:1  busy:  63.58  I/Os:    12 (sequential:0 nearly:3 random:9)
disk:2  busy:  75.46  I/Os:    13 (sequential:0 nearly:3 random:10)
disk:3  busy:  33.35  I/Os:     6 (sequential:0 nearly:1 random:5)
disk:4  busy:  95.65  I/Os:    16 (sequential:0 nearly:2 random:14)
disk:5  busy: 100.00  I/Os:    17 (sequential:0 nearly:3 random:14)
disk:6  busy:  70.03  I/Os:    11 (sequential:0 nearly:1 random:10)
disk:7  busy:  77.44  I/Os:    13 (sequential:0 nearly:1 random:12)

STAT totalTime 156.49999999999994
```

To do a rough estimate of the time taken we have 100 accesses and 8 disks. So, let us take an average of 10 units of time for each write then the estimate becomes 100*10/8 equals 125 that is a nearest value to 156.5.

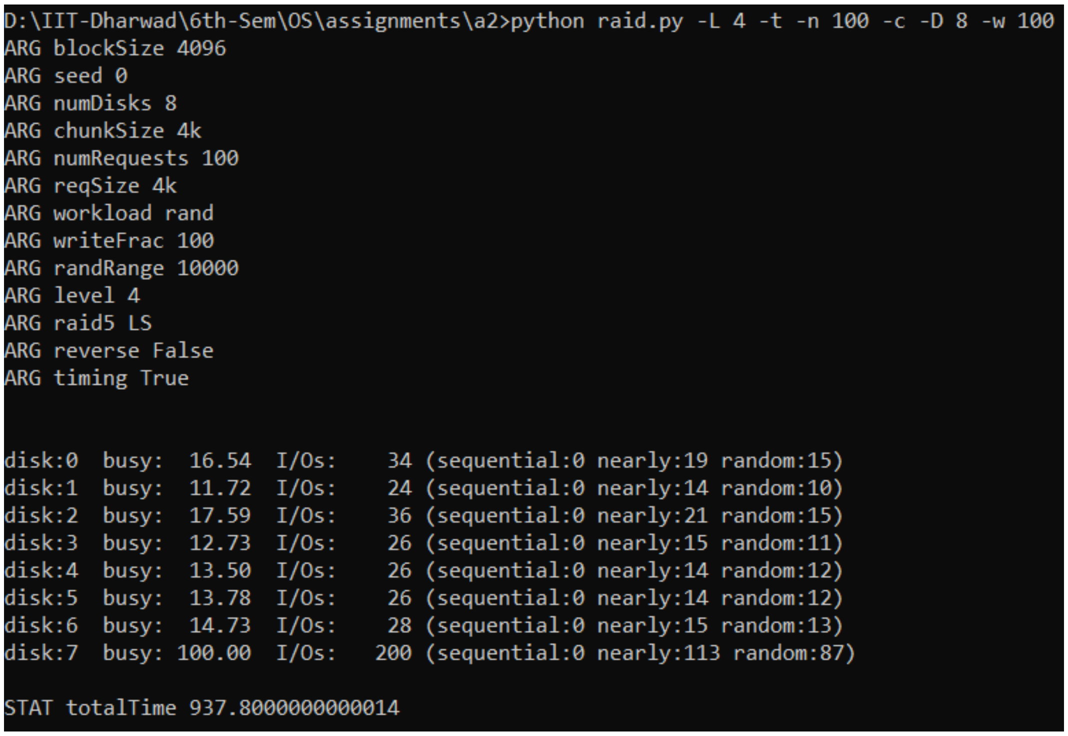
python raid.py -L 1 -t -n 100 -c -D 8 -w 100



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c -D 8 -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:1  busy: 100.00  I/Os:    28 (sequential:0 nearly:1 random:27)
disk:2  busy:  93.91  I/Os:    29 (sequential:0 nearly:6 random:23)
disk:3  busy:  93.91  I/Os:    29 (sequential:0 nearly:6 random:23)
disk:4  busy:  87.92  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:5  busy:  87.92  I/Os:    24 (sequential:0 nearly:0 random:24)
disk:6  busy:  65.94  I/Os:    19 (sequential:0 nearly:1 random:18)
disk:7  busy:  65.94  I/Os:    19 (sequential:0 nearly:1 random:18)

STAT totalTime 275.69999999999993
```

To do a rough estimate of the time taken we have 100 accesses and 8 disks (since this is RAID 1). So, let us take an average of 10 units of time for each write then the estimate becomes 100*10 / (8/2) equals 250 that is a nearest estimate to 275.7.

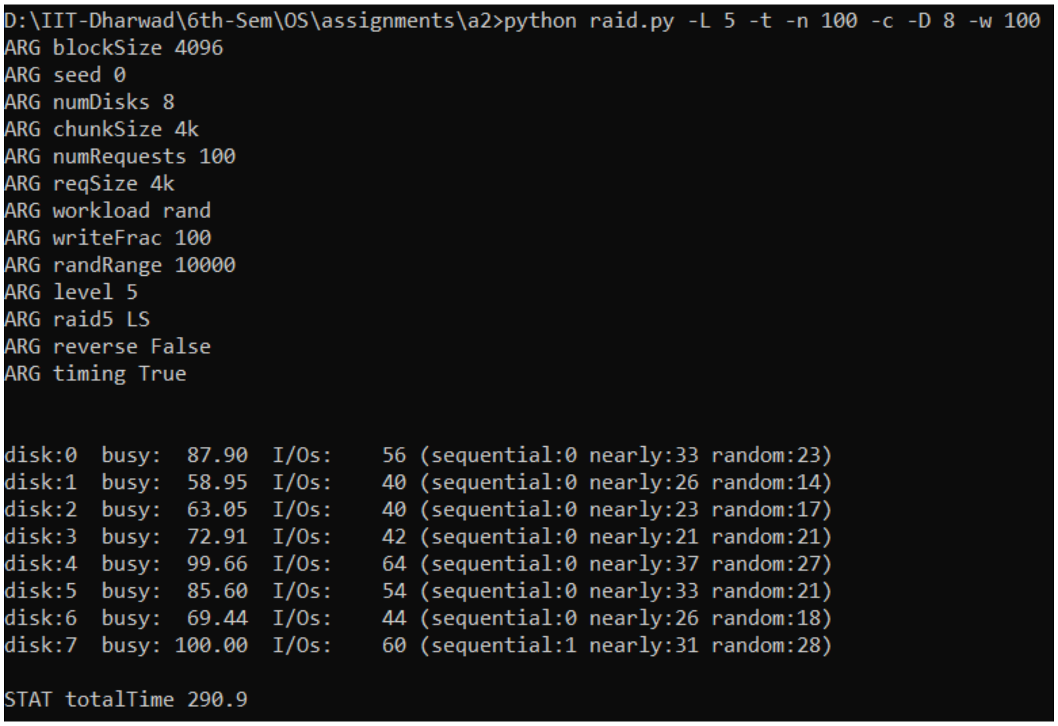
python raid.py -L 4 -t -n 100 -c -D 8 -w 100



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c -D 8 -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  16.54  I/Os:    34 (sequential:0 nearly:19 random:15)
disk:1  busy:  11.72  I/Os:    24 (sequential:0 nearly:14 random:10)
disk:2  busy:  17.59  I/Os:    36 (sequential:0 nearly:21 random:15)
disk:3  busy:  12.73  I/Os:    26 (sequential:0 nearly:15 random:11)
disk:4  busy:  13.50  I/Os:    26 (sequential:0 nearly:14 random:12)
disk:5  busy:  13.78  I/Os:    26 (sequential:0 nearly:14 random:12)
disk:6  busy:  14.73  I/Os:    28 (sequential:0 nearly:15 random:13)
disk:7  busy: 100.00  I/Os:   200 (sequential:0 nearly:113 random:87)

STAT totalTime 937.8000000000014
```

Here we have parity involved and the writes are expensive so the estimation cannot be done linearly as above.

python raid.py -L 5 -t -n 100 -c -D 8 -w 100



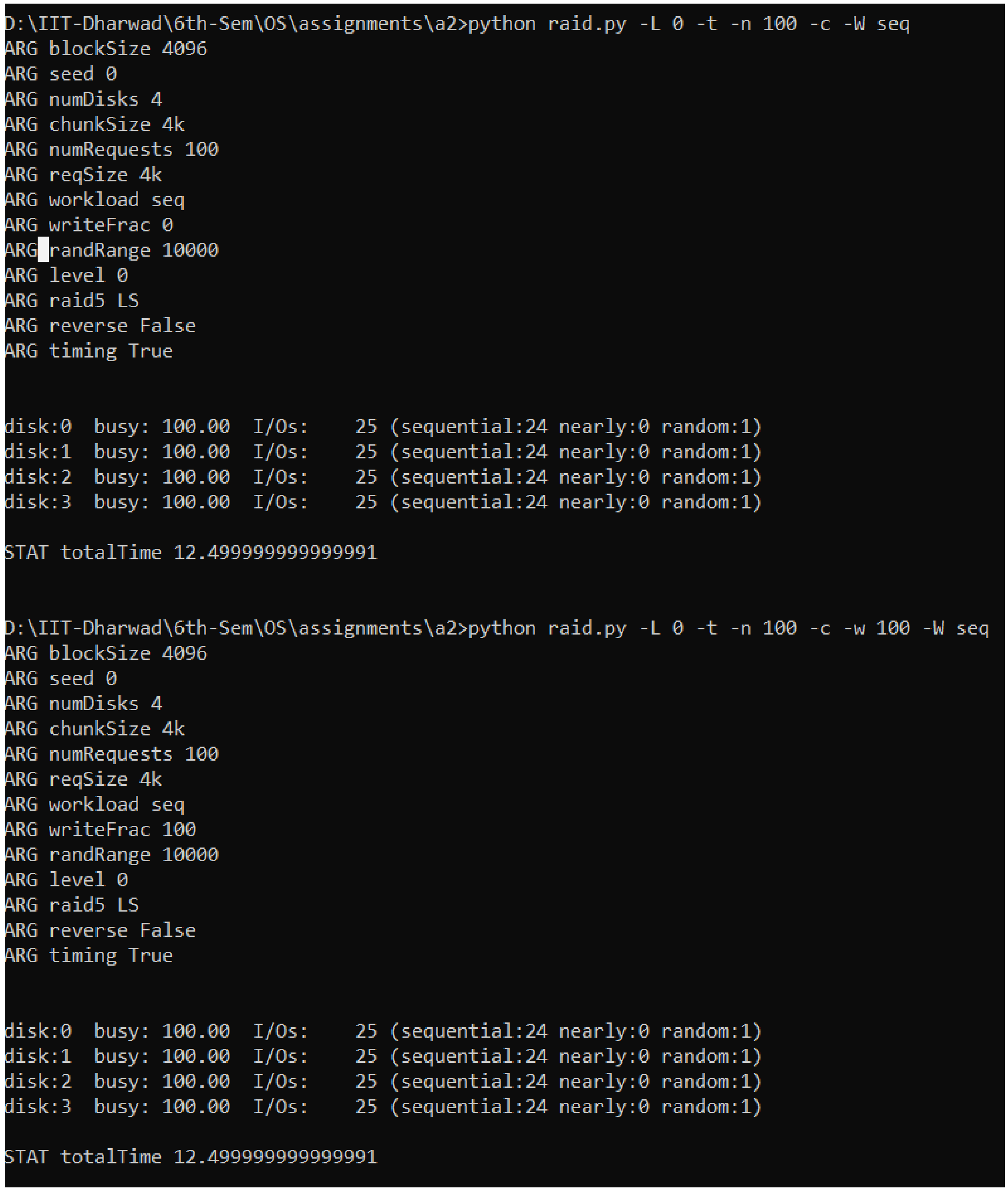
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c -D 8 -w 100
ARG blockSize 4096
ARG seed 0
ARG numDisks 8
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload rand
ARG writeFrac 100
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  87.90  I/Os:    56 (sequential:0 nearly:33 random:23)
disk:1  busy:  58.95  I/Os:    40 (sequential:0 nearly:26 random:14)
disk:2  busy:  63.05  I/Os:    40 (sequential:0 nearly:23 random:17)
disk:3  busy:  72.91  I/Os:    42 (sequential:0 nearly:21 random:21)
disk:4  busy:  99.66  I/Os:    64 (sequential:0 nearly:37 random:27)
disk:5  busy:  85.60  I/Os:    54 (sequential:0 nearly:33 random:21)
disk:6  busy:  69.44  I/Os:    44 (sequential:0 nearly:26 random:18)
disk:7  busy: 100.00  I/Os:    60 (sequential:1 nearly:31 random:28)

STAT totalTime 290.9
```

Here we have parity involved and the writes are expensive so the estimation cannot be done linearly as above.

RAID Level	**Performance Scale**
0	275.7 / 156.5 = 1.76
1	509.8 / 275.7 = 1.85
4	982.5 / 937.8 = 1.05
5	497.4 / 290.9 = 1.71

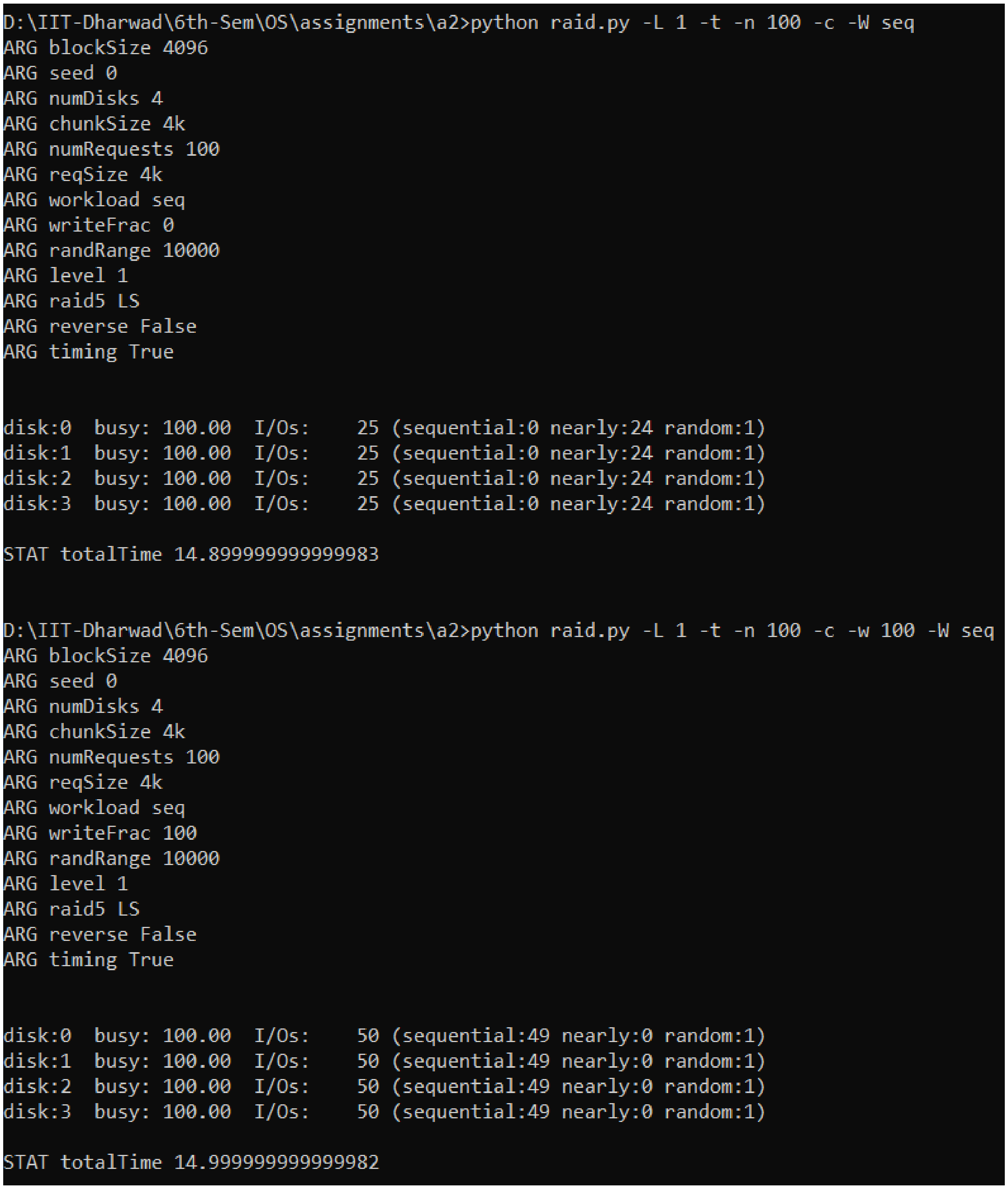
Here, only level 4 scales badly whereas all the others levels scale similarly.

(8)



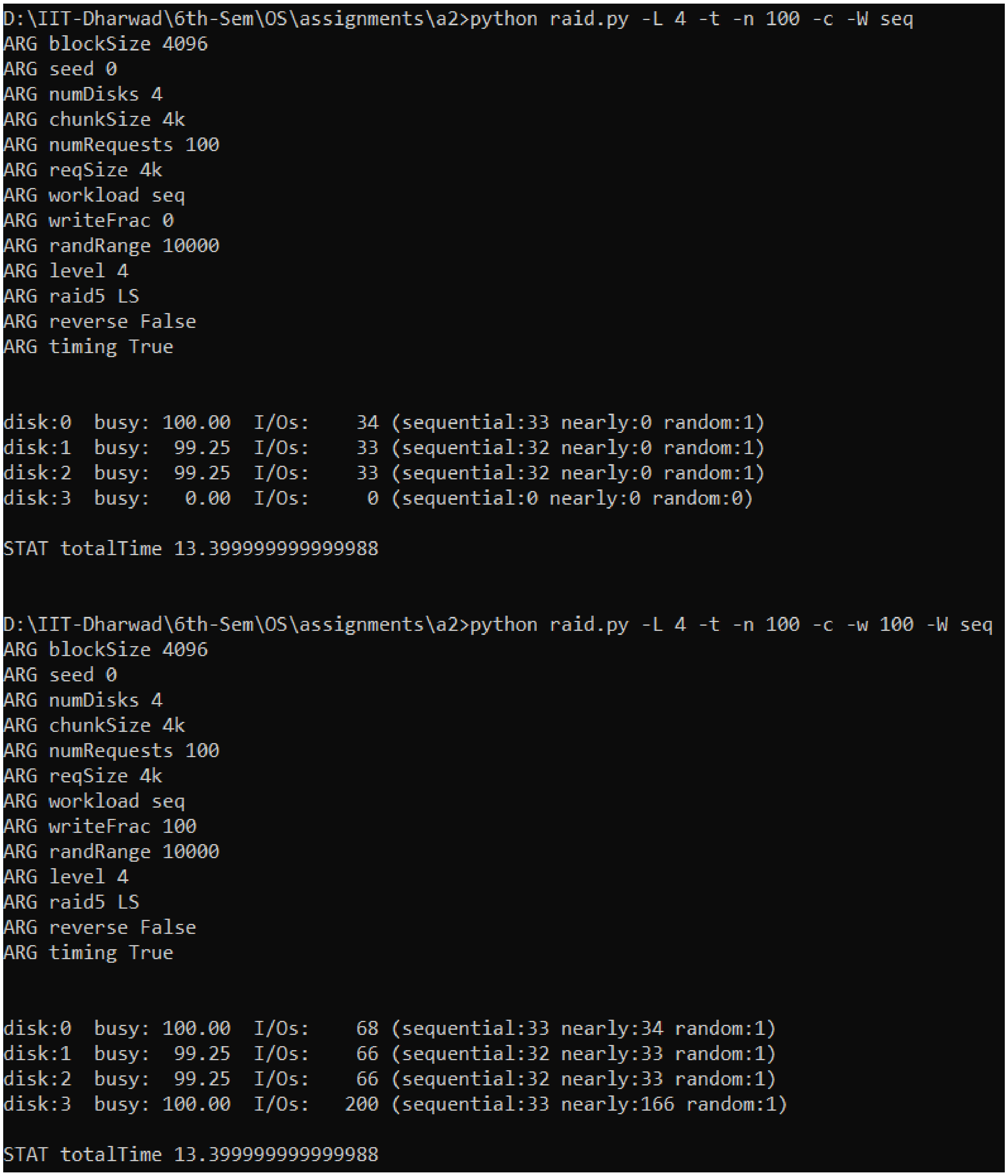
```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:1  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:2  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:3  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)

STAT totalTime 12.499999999999991


D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 0 -t -n 100 -c -w 100 -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 100
ARG randRange 10000
ARG level 0
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:1  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:2  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)
disk:3  busy: 100.00  I/Os:    25 (sequential:24 nearly:0 random:1)

STAT totalTime 12.499999999999991
```

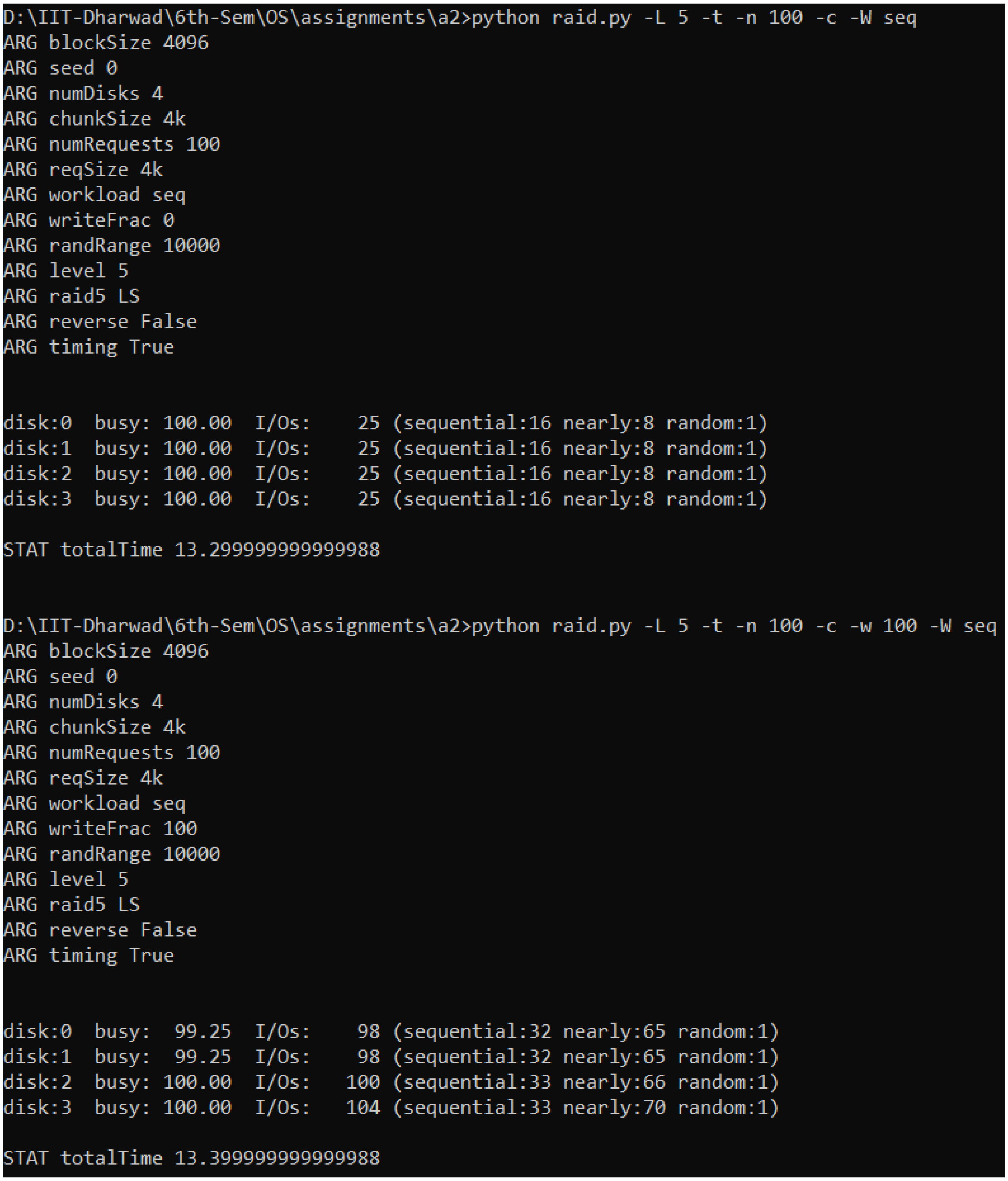
Both read and write take the same total time of 12.5. So we can say that there is no performance variance with sequential workload for read and write. But for a sequential workload the total time is 12.5 whereas for a non-sequential workload, the total time is 257.7. Therefore there is 275.7 / 12.5 = 22 times increase in performance for a sequential workload.



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    25 (sequential:0 nearly:24 random:1)
disk:1  busy: 100.00  I/Os:    25 (sequential:0 nearly:24 random:1)
disk:2  busy: 100.00  I/Os:    25 (sequential:0 nearly:24 random:1)
disk:3  busy: 100.00  I/Os:    25 (sequential:0 nearly:24 random:1)

STAT totalTime 14.899999999999983


D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 1 -t -n 100 -c -w 100 -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 100
ARG randRange 10000
ARG level 1
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    50 (sequential:49 nearly:0 random:1)
disk:1  busy: 100.00  I/Os:    50 (sequential:49 nearly:0 random:1)
disk:2  busy: 100.00  I/Os:    50 (sequential:49 nearly:0 random:1)
disk:3  busy: 100.00  I/Os:    50 (sequential:49 nearly:0 random:1)

STAT totalTime 14.999999999999982
```

Both read and write take the same total time of around 15. So we can say that there is no performance variance with sequential workload for read and write. But for a sequential workload the total time is 15 whereas for a non-sequential workload, the total time is 509.8. Therefore there is 509.8 / 15 = 34 times increase in performance for a sequential workload.


```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    34 (sequential:33 nearly:0 random:1)
disk:1  busy:  99.25  I/Os:    33 (sequential:32 nearly:0 random:1)
disk:2  busy:  99.25  I/Os:    33 (sequential:32 nearly:0 random:1)
disk:3  busy:   0.00  I/Os:     0 (sequential:0 nearly:0 random:0)

STAT totalTime 13.399999999999988


D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 4 -t -n 100 -c -w 100 -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 100
ARG randRange 10000
ARG level 4
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    68 (sequential:33 nearly:34 random:1)
disk:1  busy:  99.25  I/Os:    66 (sequential:32 nearly:33 random:1)
disk:2  busy:  99.25  I/Os:    66 (sequential:32 nearly:33 random:1)
disk:3  busy: 100.00  I/Os:   200 (sequential:33 nearly:166 random:1)

STAT totalTime 13.399999999999988
```
Both read and write take the same total time of around 13.4. So we can say that there is no performance variance with sequential workload for read and write. But for a sequential workload the total time is 13.4 whereas for a non-sequential workload, the total time is 982.5. Therefore there is 982.5 / 13.4 = 73 times increase in performance for a sequential workload.



```
D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 0
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy: 100.00  I/Os:    25 (sequential:16 nearly:8 random:1)
disk:1  busy: 100.00  I/Os:    25 (sequential:16 nearly:8 random:1)
disk:2  busy: 100.00  I/Os:    25 (sequential:16 nearly:8 random:1)
disk:3  busy: 100.00  I/Os:    25 (sequential:16 nearly:8 random:1)

STAT totalTime 13.299999999999988


D:\IIT-Dharwad\6th-Sem\OS\assignments\a2>python raid.py -L 5 -t -n 100 -c -w 100 -W seq
ARG blockSize 4096
ARG seed 0
ARG numDisks 4
ARG chunkSize 4k
ARG numRequests 100
ARG reqSize 4k
ARG workload seq
ARG writeFrac 100
ARG randRange 10000
ARG level 5
ARG raid5 LS
ARG reverse False
ARG timing True


disk:0  busy:  99.25  I/Os:    98 (sequential:32 nearly:65 random:1)
disk:1  busy:  99.25  I/Os:    98 (sequential:32 nearly:65 random:1)
disk:2  busy: 100.00  I/Os:   100 (sequential:33 nearly:66 random:1)
disk:3  busy: 100.00  I/Os:   104 (sequential:33 nearly:70 random:1)

STAT totalTime 13.399999999999988
```

Both read and write take the same total time of around 13.4. So we can say that there is no performance variance with sequential workload for read and write. But for a sequential workload the total time is 13.4 whereas for a non-sequential workload, the total time is 497.4. Therefore there is 497.4 / 13.4 = 37 times increase in performance for a sequential workload.

For different request sizes like 8k,12k,16k also the total time increases as the workload request size increases because this increases the total access time as we may have to access more disks for processing a request. But RAID4 and RAID 5 are generally better for workloads of sizes 12k, 16k. As they scale up well in these cases. This is true for sequential workloads also thus even in sequential workloads RAID 4,5 are better for larger workloads request sizes like 12k, 16k.