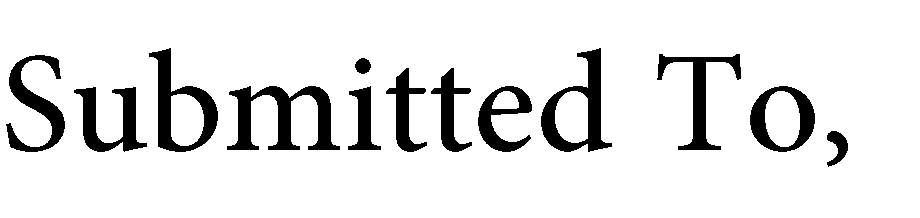
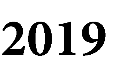
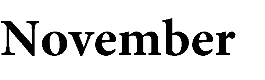
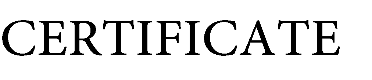


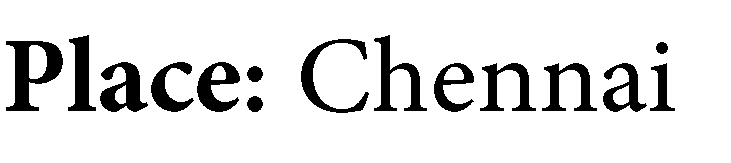
Mahale Shantanu 17BCE1161



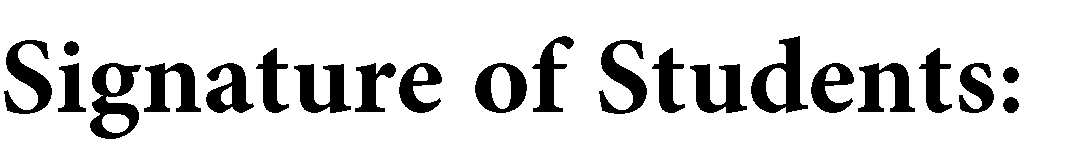


**S. Harini**





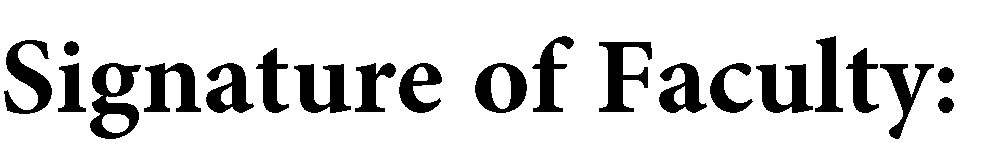


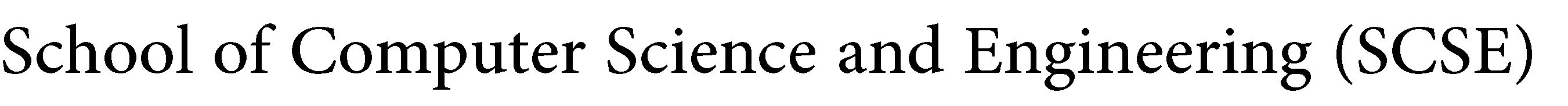


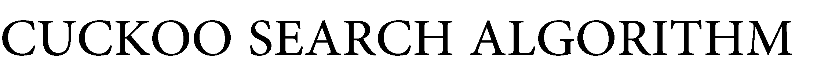
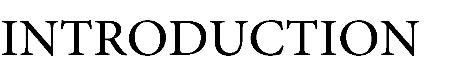
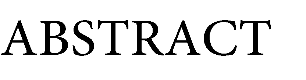
Kumar Shikhar 17BCE1198

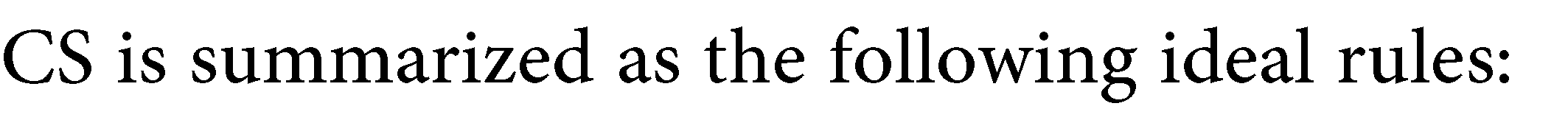
Abhijit Pingle 17BCE1254

Vinay C Shekhar 17BCE1317



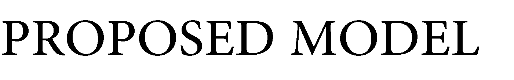
**Ayesha**

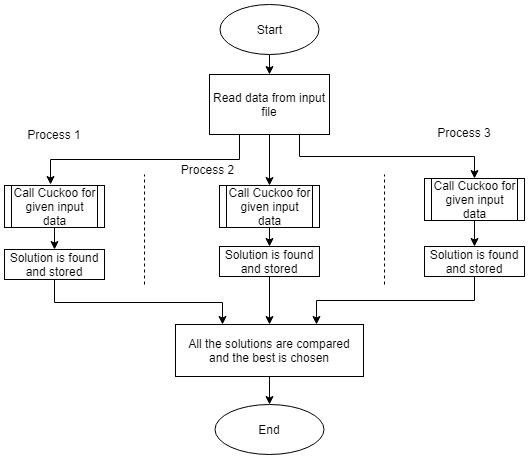


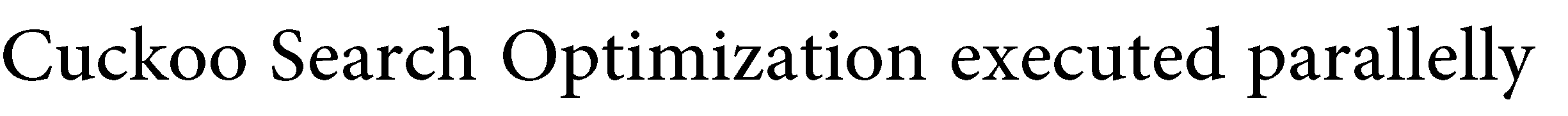


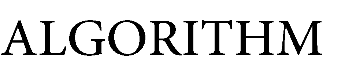
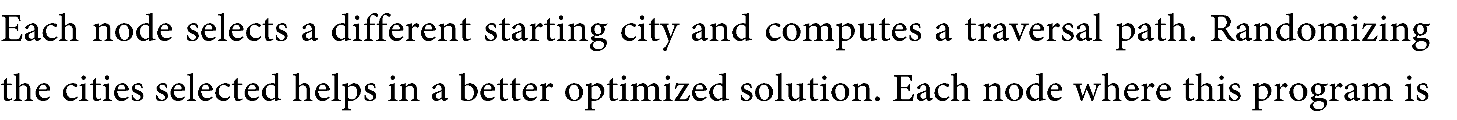
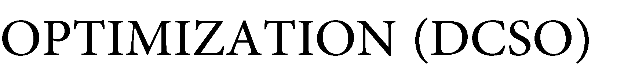
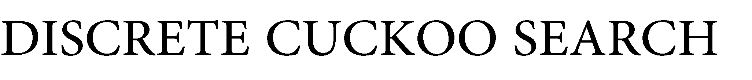


∈









**Algorithm:** Discrete Cuckoo Search Optimization

**Input:** Distance Matrix

**Output:** Optimized path and cost

1: numNest = sizeof(InputMatrix) 2: Pa = 0.2 \* numNest

3: MaxGen = 50

4: Generate initial population with randomly chosen city and store in nests.

5: **while** (MaxGen)

6: Assign random index i from nests to Cnest 7: Calculate Levy Flight

8: **if** Levy Flight>2:

Perform Double Bridge for Cnest with random values 9: **else**:

Perform 2-Opt-Move for Cnest with random values

# end if

10: Get random index j from nests 11: **if** cost(nest\_j) > cost(Cnest)

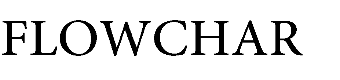
Replace j by the new solution;

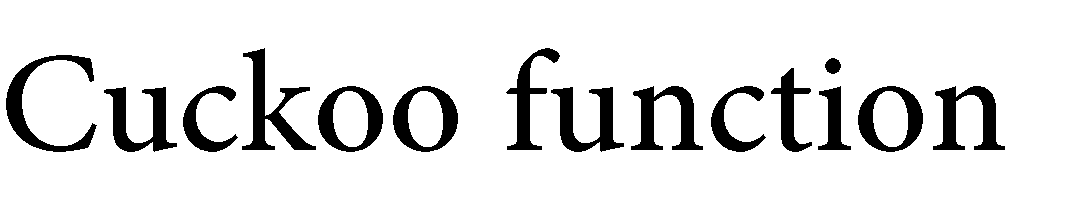
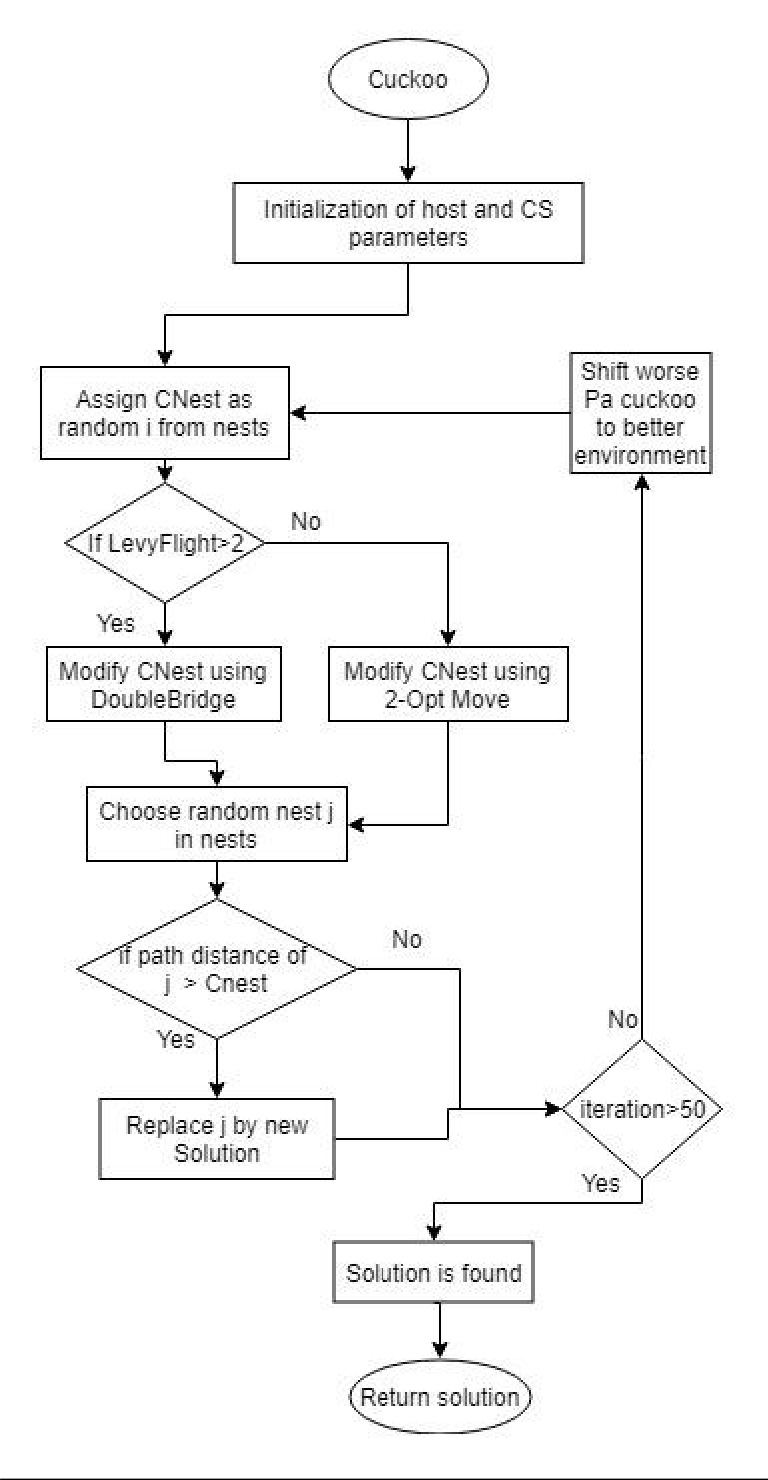
# end if

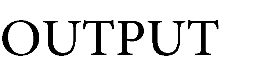
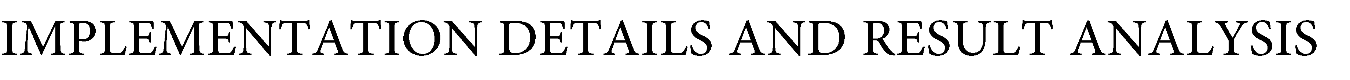
12: A fraction(Pa) of the worse nests are abandoned. 13: Sort the nests based on cost

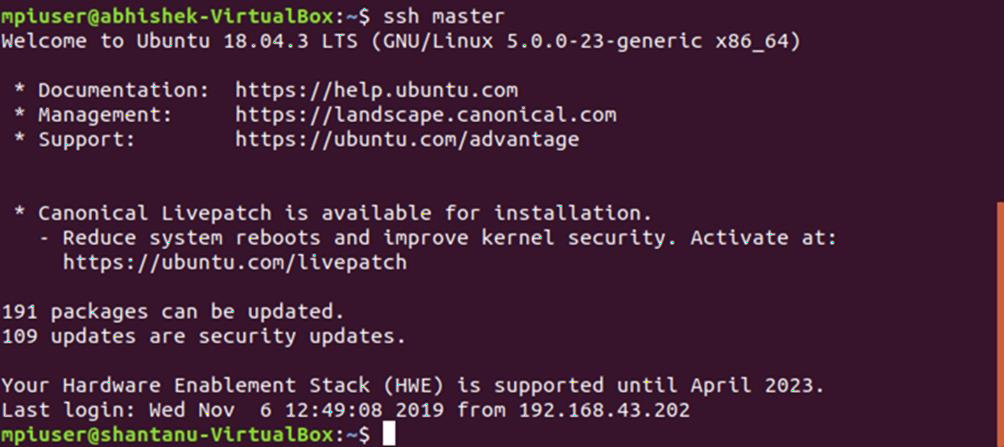
# end while

14: nest[0] is the solution

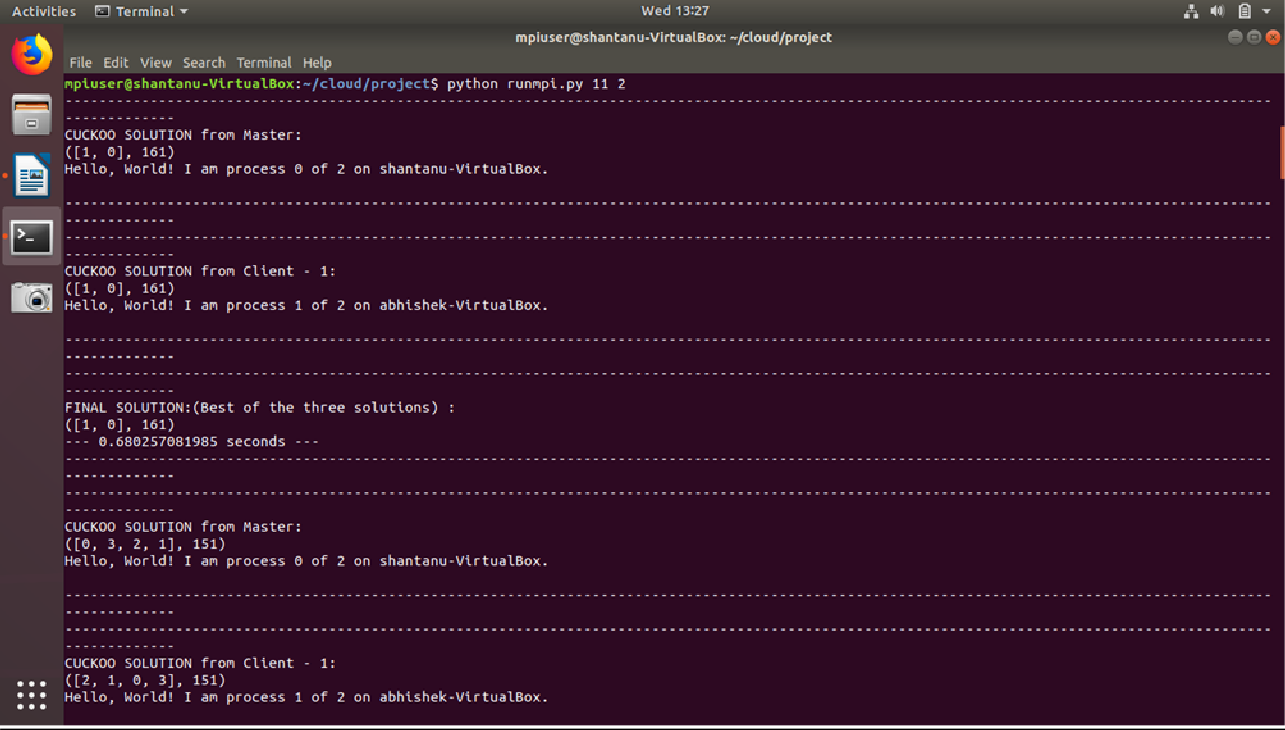


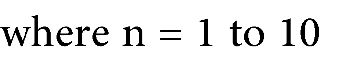
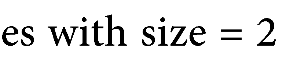
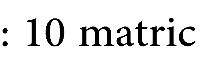
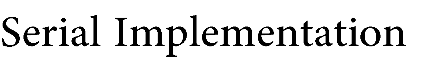


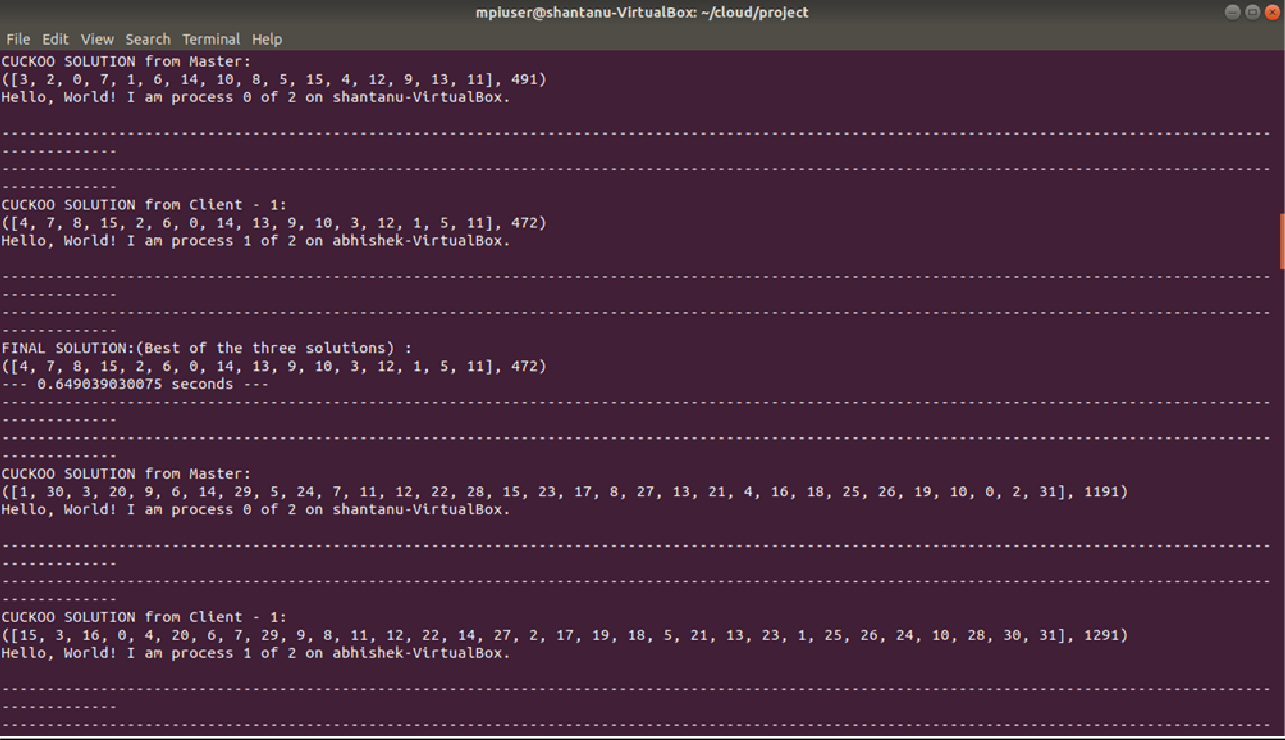


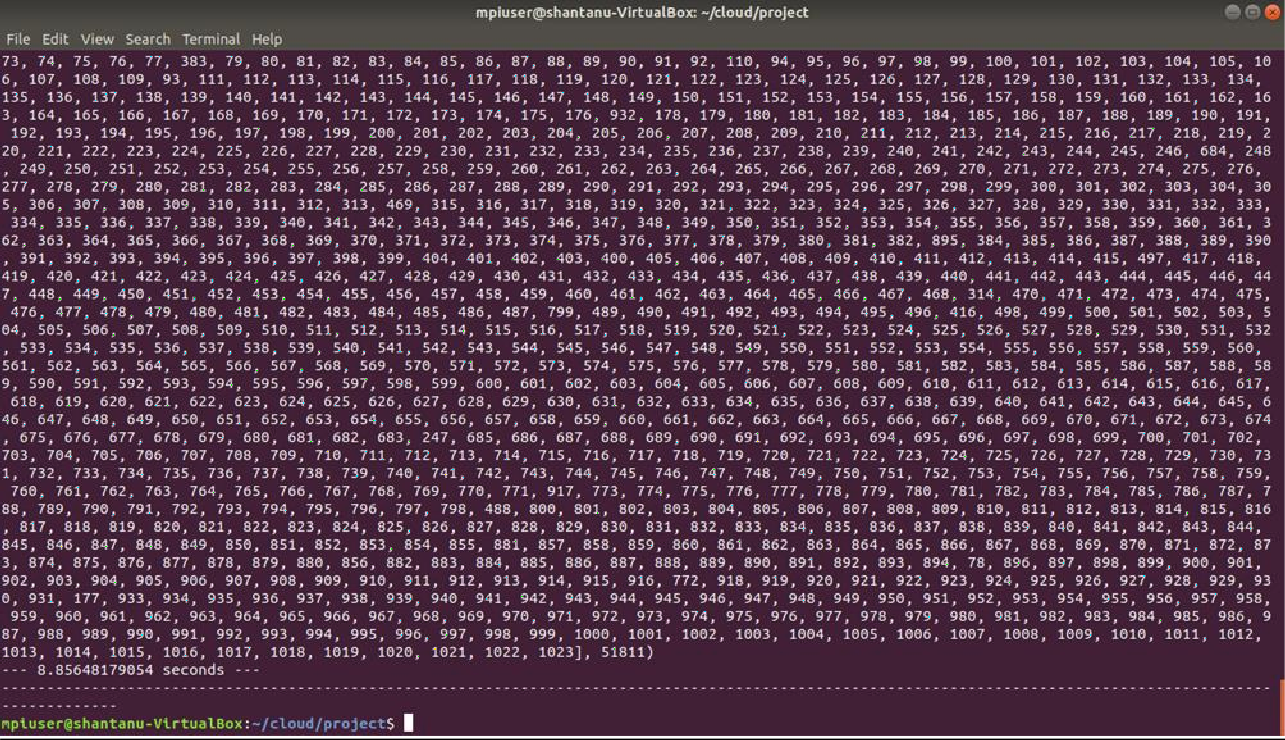


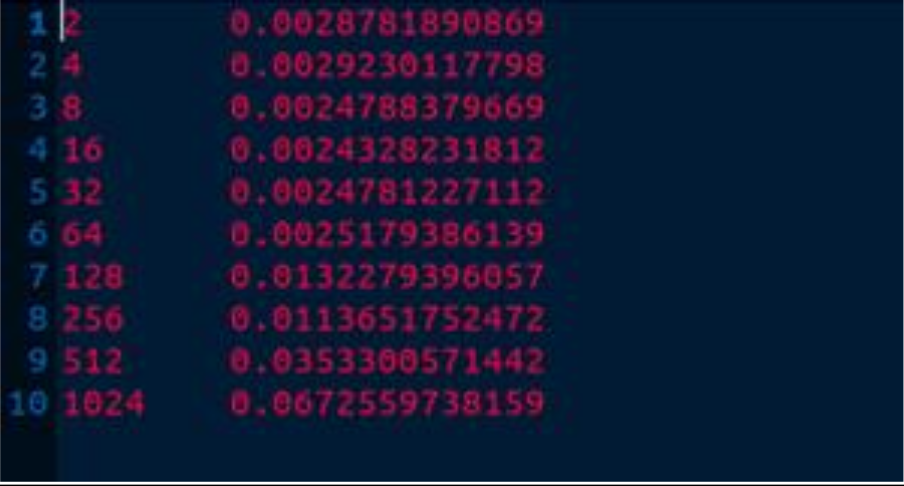


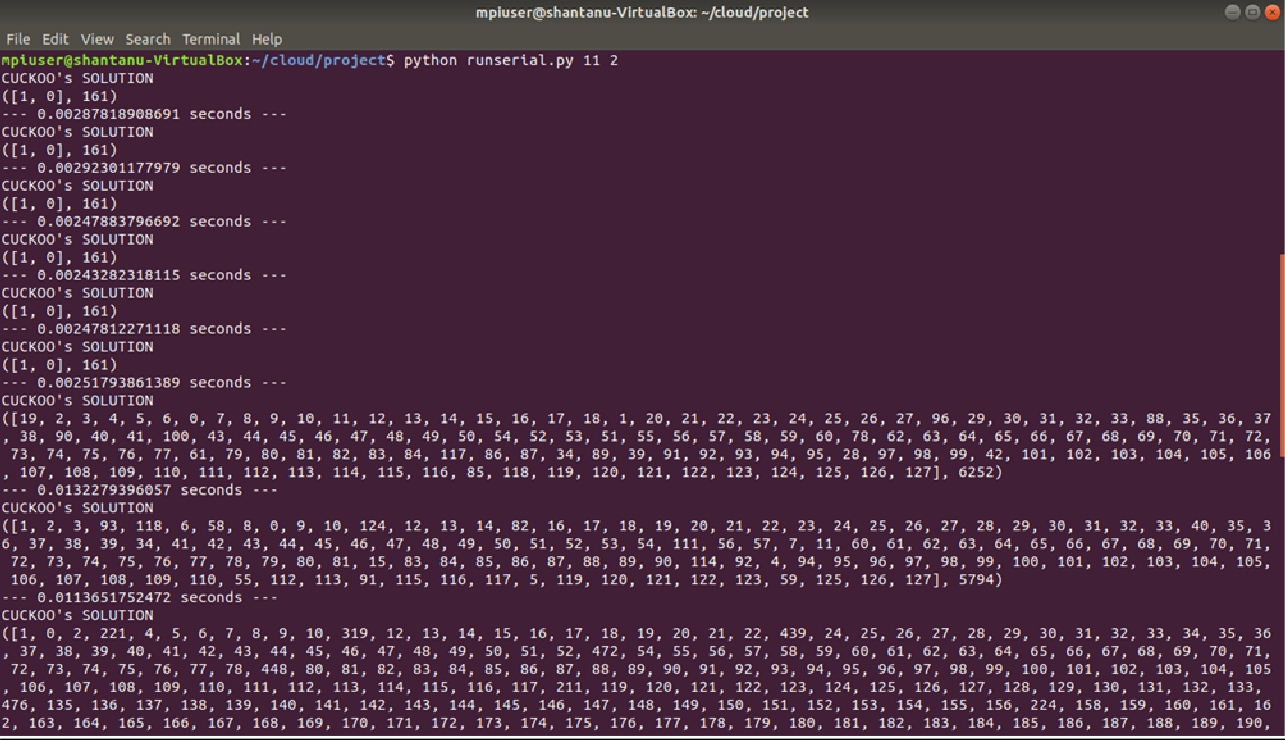


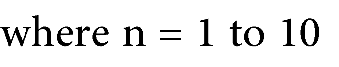
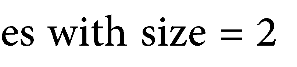
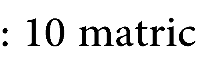
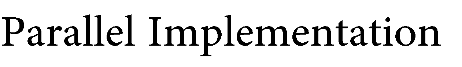


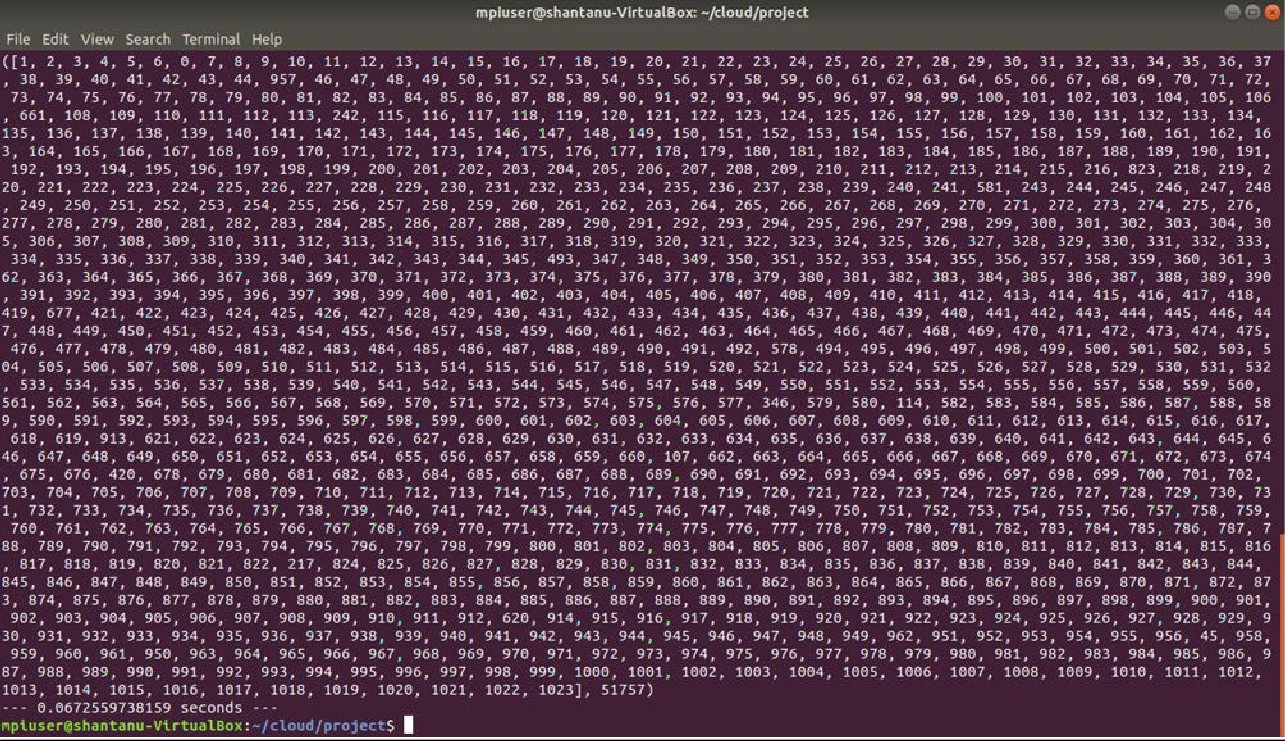


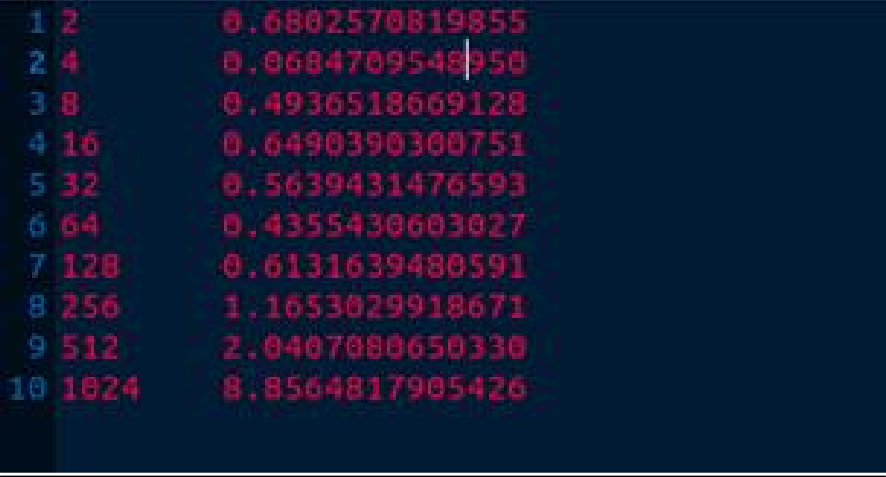


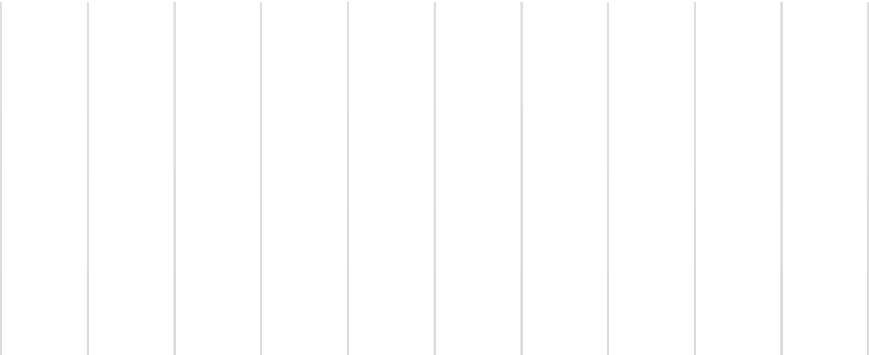












128 256 512 1024

64

32

16

8

4

2

10

9

8

7

6

5

4

3

2

1

0

Parallel

Serial

**Comparison**

