REPORT

Team Members:

1. Name: Madhuri Bandari

Email Id: bandari.24@wright.edu

2. Name: Venkata Sireesha Marella

Email Id: marella.22@wright.edu

3. Name: Naga Deepthi Doddaka

Email Id: doddaka.3@wright.edu

Pseudocode:

```
1. To obtain the list of jobs (JobID, start, finish), parse the input.
2. Arrange the tasks according to their deadlines (a self-serving tactic).
3. Start two lists: one for jobs that are linked to machine M1, and another for machine M2.
4. Work your way through the sorted jobs:
a. Employ M1 to plan a job if possible (its start time >= M1's present free time).
b. If not, see if the task can be scheduled on M2 (provided that its start time is greater than M2's available free time).
5. After work is scheduled, update the machine's (M1 or M2) free time.
6. Output all planned jobs as well as the jobs that are allocated to each machine.
```

Greedy Heuristic Used:

Jobs are chosen by the algorithm based on when they finish (earliest first). This guarantees that a machine is as free as feasible at all times, enabling the scheduling of more work later. Jobs that can be scheduled on M1 are given priority; if not, they are scheduled on M2, if space is available. This method ensures that the number of non-overlapping jobs on both machines is maximized.

Informal Argument for Correctness:

We make sure that jobs have the largest feasible window of time to schedule them in the future by scheduling jobs with the earliest end time first. Because arranging a job with an earlier end time does not conflict with other tasks that may be booked later, this avaricious decision is the best one. We additionally guarantee effective task distribution between the two machines by giving M1 priority. The number of scheduled jobs is maximized using this strategy.

Computational Complexity:

1. Sorting the jobs by their end times takes ---> **O(n log n)**, where n is the no. of jobs/duties.

2. Iterating through the duties to allocate them takes **O(n).**

Therefore, overall time complexity (T.C) of the greedy algorithm is **O(n log n)**

Test Cases

In my solution, **input.txt**, **output.txt** and **assignment2.py** python program files check the output of the given input by only using single argument by using command line and helps in generating the output.txt file in the same working directory.

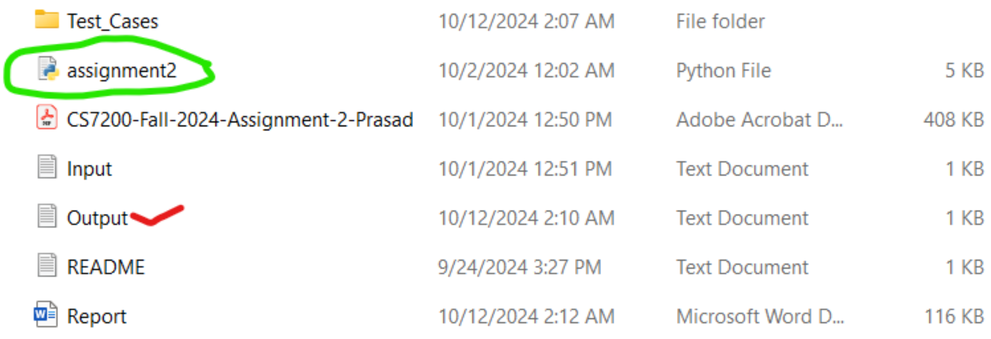
By using below Command, it automatically generates output.txt in same directory:

<u>Test Cases:</u>

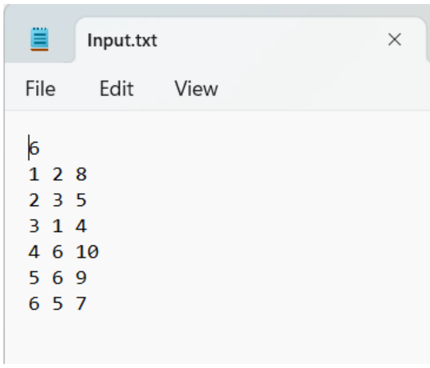
n = 6 Input.txt, Output.txt

Command line:

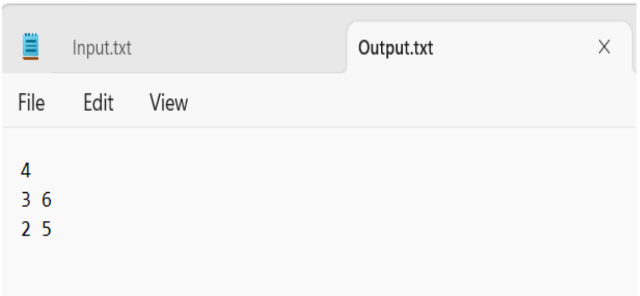
python assignment2.py Input.txt



Test_Cases	10/12/2024 2:07 AM	File folder	
assignment2	10/2/2024 12:02 AM	Python File	5 KB
CS7200-Fall-2024-Assignment-2-Prasad	10/1/2024 12:50 PM	Adobe Acrobat D...	408 KB
Input	10/1/2024 12:51 PM	Text Document	1 KB
Output	10/12/2024 2:10 AM	Text Document	1 KB
README	9/24/2024 3:27 PM	Text Document	1 KB
Report	10/12/2024 2:12 AM	Microsoft Word D...	116 KB



```
6
1 2 8
2 3 5
3 1 4
4 6 10
5 6 9
6 5 7
```



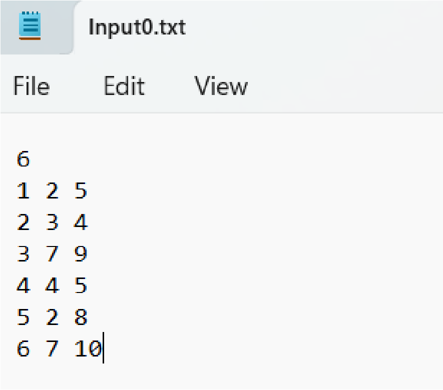
```
4
3 6
2 5
```

Examples:

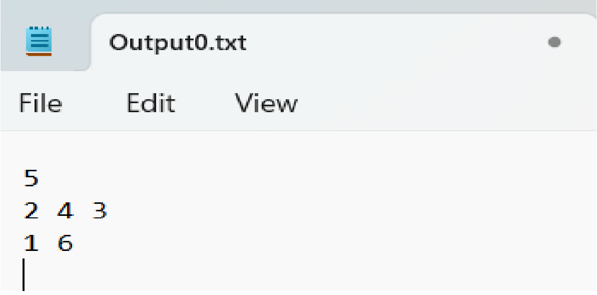
n = 6 Input0.txt, Output0.txt

Command line:

python assignment2.py Input0.txt



```
6
1 2 5
2 3 4
3 7 9
4 4 5
5 2 8
6 7 10
```



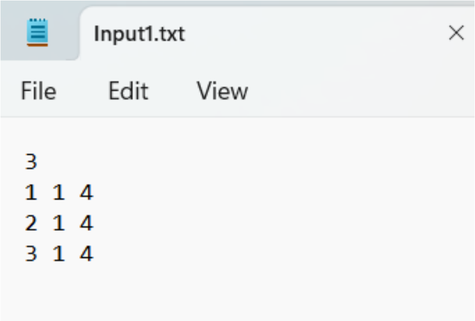
```
5
2 4 3
1 6
```

----

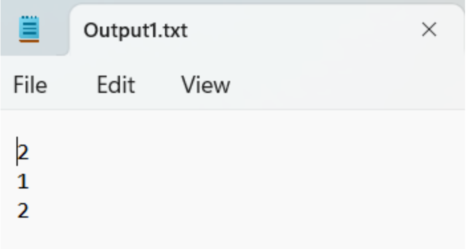
n = 3 Input1.txt, Output1.txt

Command line:

python assignment2.py Input1.txt



```
3
1 1 4
2 1 4
3 1 4
```



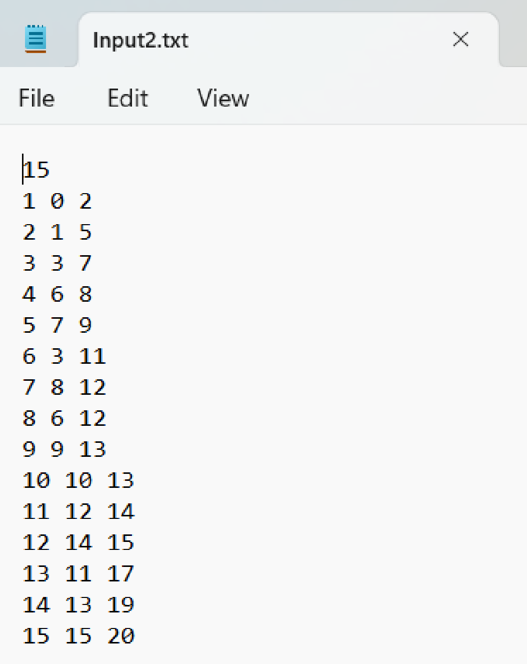
```
2
1
2
```

----

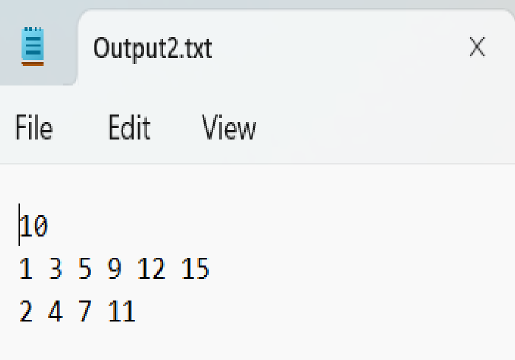
n = 15   Input2.txt, Output2.txt

Command line:

python assignment2.py Input2.txt



```
15
1 0 2
2 1 5
3 3 7
4 6 8
5 7 9
6 3 11
7 8 12
8 6 12
9 9 13
10 10 13
11 12 14
12 14 15
13 11 17
14 13 19
15 15 20
```



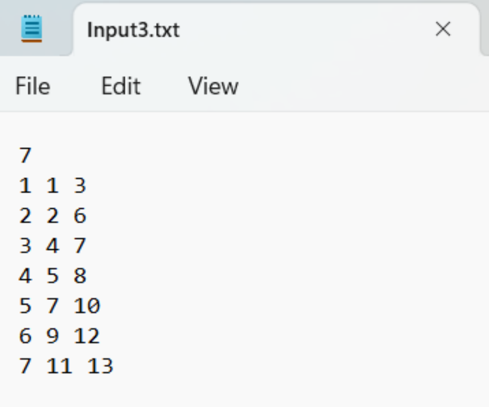
```
10
1 3 5 9 12 15
2 4 7 11
```

---

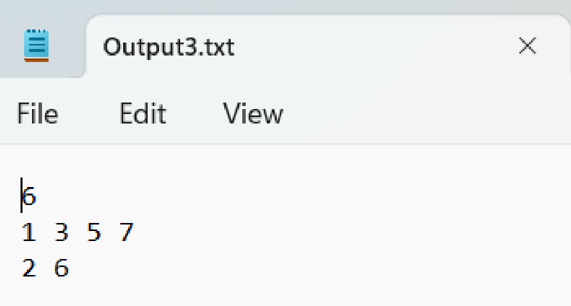
n = 7 Input3.txt, Output3.txt

Command line:

python assignment2.py Input3.txt



```
7
1 1 3
2 2 6
3 4 7
4 5 8
5 7 10
6 9 12
7 11 13
```



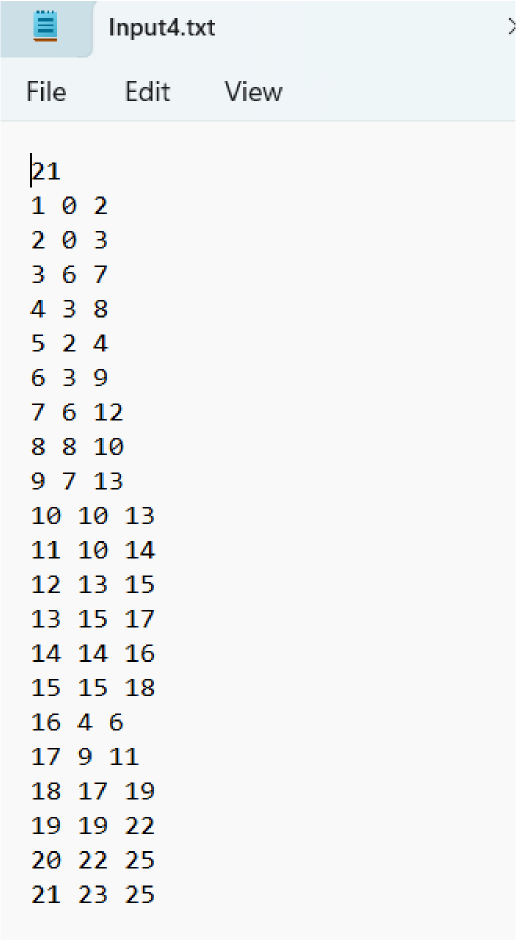
```
6
1 3 5 7
2 6
```

---

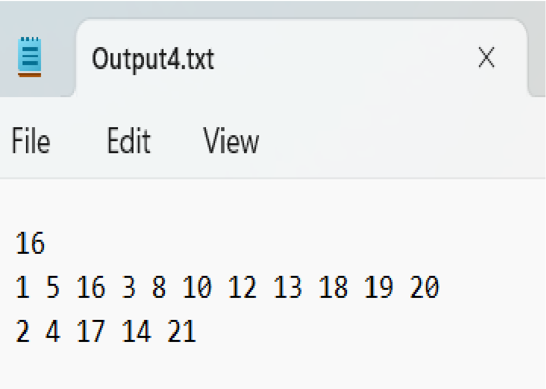
n = 21 Input4.txt, Output4.txt

Command line:

python assignment2.py Input4.txt

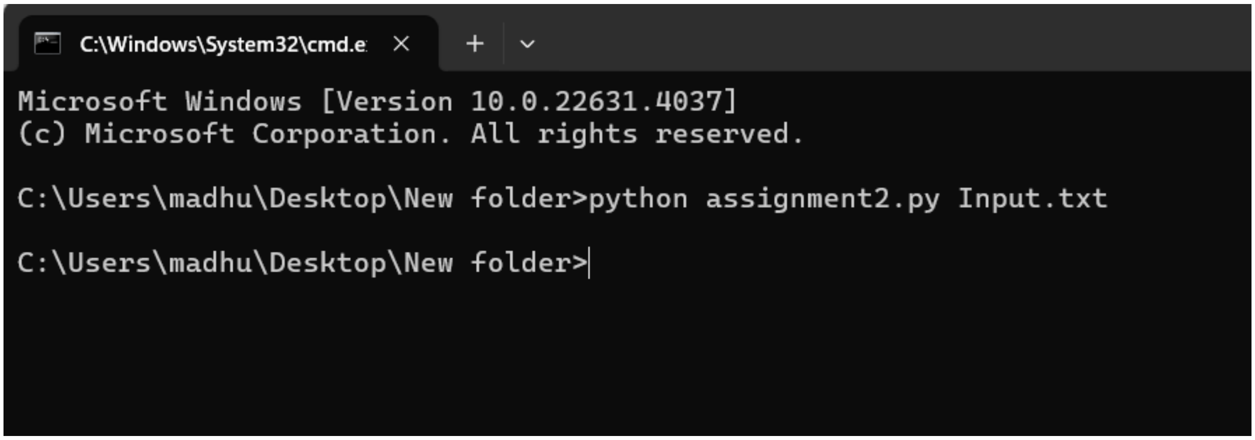


```
21
1 0 2
2 0 3
3 6 7
4 3 8
5 2 4
6 3 9
7 6 12
8 8 10
9 7 13
10 10 13
11 10 14
12 13 15
13 15 17
14 14 16
15 15 18
16 4 6
17 9 11
18 17 19
19 19 22
20 22 25
21 23 25
```



```
16
1 5 16 3 8 10 12 13 18 19 20
2 4 17 14 21
```

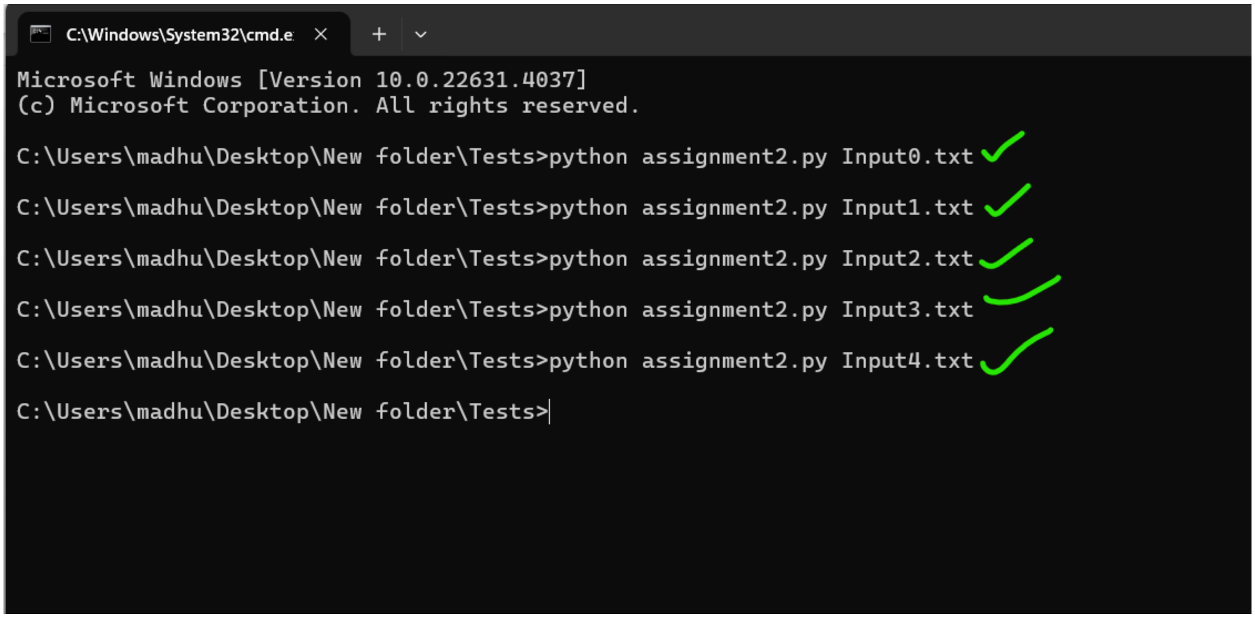
Results Screenshot’s (assignment2.py)



```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\madhu\Desktop\New folder>python assignment2.py Input.txt

C:\Users\madhu\Desktop\New folder>
```



```
Microsoft Windows [Version 10.0.22631.4037]
(c) Microsoft Corporation. All rights reserved.

C:\Users\madhu\Desktop\New folder\Tests>python assignment2.py Input0.txt ✓

C:\Users\madhu\Desktop\New folder\Tests>python assignment2.py Input1.txt ✓

C:\Users\madhu\Desktop\New folder\Tests>python assignment2.py Input2.txt ✓

C:\Users\madhu\Desktop\New folder\Tests>python assignment2.py Input3.txt ✓

C:\Users\madhu\Desktop\New folder\Tests>python assignment2.py Input4.txt ✓

C:\Users\madhu\Desktop\New folder\Tests>
```