CSCI 406: AlgoBOWL

Reminder: You are NOT allowed to consult the internet to solve this problem.

This project directly addresses Learning Objectives 7 (develop a competitive but necessarily suboptimal solution for an NP complete problem) and 9 (function effectively on an algorithm design and implementation team) listed in the syllabus.

1 Problem Description

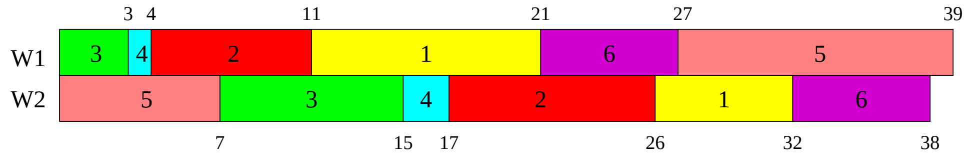
You are given n *jobs*, m *workstations* and an $n \times m$ two-dimensional *task matrix* T of the time each job will spend at each workstation. Each job becomes available at a specified time and may be processed in only one workstation at a time. Each workstation may process only one job at a time. The goal is to assign a start time for each job at each workstation that minimizes the finish time of the last job[1].

Input Format:

The input must be provided in a file in the following format: the first line contains two integers: the first denotes n, the number of jobs, and the second denotes m, the number of of workstations. Each of the n subsequent lines contains the time at which the job becomes available followed by the appropriate row (consisting of m integers) of the task matrix T.

```
6 2
10 10 6
0 7 9
0 3 8
3 1 2
0 12 7
20 6 6
```

The input above describes six jobs numbered 1 through 6 and two workstations numbered 1 and 2. A possible solution is shown below.



The figure can initially appear overwhelming so please take a few minutes to make sure you get it and also appreciate its limitations.

- The first and second rows show the jobs assigned to Workstations `W1` and `W2`, respectively.

[1]To understand why a job can only be processed in one workstation at a time, consider a large automotive garage with three specialized shops. A car may require the following work: replace exhaust pipes and muffler; align wheels; and tune-up. These three tasks may be carried out in any order. However, since the exhaust, alignment and tune-up shops are in different buildings, it is not possible to perform two tasks simultaneously on one car. Additionally, people bring their cars in at varying times of day, and a car cannot be worked on till it is brought in.

- The numbers inside each block indicates which task is being processed. For example, the '3' in the first (green) block allocated to W1 represents the task from Job 3 (which requires 3 units).
- The overall completion time is 39.
- Observe that tasks from the same job do not overlap in time. For example, Job 3 on W1 is performed in time interval $[0, 3]$ while Job 3 on W2 is performed in time interval $[7, 15]$.
- Also observe that all jobs begin execution at or after the time they become available. For example Job 1 becomes available at time 10 and begins execution at time 11.
- Note that in general there could be idle time on workstations (although this is not the case in this example).

Input Restrictions: To simplify the problem, we will require that the number of workstations $m = 3$; the number of jobs n is no more than 1000; jobs become available at integer times in $[0, 50]$; and all task times are integers in $[1, 50]$. Note that the illustrative example above would not be considered valid since it has $m = 2$.

Output Format: Line 1 of your output will contain the overall completion time of your solution. Each of the subsequent n lines corresponds to a job (similar to the input format) and will contain the start times of its m tasks.

```
39
11 26
4 17
0 7
3 15
27 0
21 32
```

Note that this problem is NP-hard, which means that it's unrealistic to expect that your algorithm will compute an optimal solution in a reasonable amount of time.

2 Deliverables

Your group has three tasks:

1. Develop as good an algorithm as you can that accepts a valid input and produces a valid output.
2. Create an input within the parameters specified above that will challenge the other groups.
3. Develop a tool that verifies the other groups' outputs. What does this mean? The purpose of the verifier is to examine the outputs that other groups compute on your input and check that (1) the output meets all of the contraints specified in the problem description and (2) the reported completion time (e.g., 39 in the example) is consistent with task start times in the output. (The verifier is not checking whether the solution provided by the other group is optimal.)