Demcon festival challenge

# Introduction

Like most scheduling problems, I expect also this one to be [NP-complete](https://en.wikipedia.org/wiki/NP-completeness). So it is possible to solve by use brute force but that would require working out all possibilities.

An option is to create a deterministic algorithm that solves the problem but might not have the best result. Now by introducing some stochastic process on the inputs, we can create a multitude of solutions and select the best one. This will still not be the perfect solution but in most cases it will be good enough.

# Datastructures

We can recognize the following data / objects:

* Show = name + starttime + endtime
* Stage = sorted list of shows (0..n)
* Schedule = list of stages (0..m)

These are used for the classes CShow, CStage and CSchedule.

# Deterministic Algorithm

The deterministic algorithm I have chosen, works as follows:

1. Give each show its own stage.
2. Try to combine stages where the shows don’t conflict.

The latter is done by comparing a stage with the others and merge when possible. This is done by starting with the last stage and checking with all previous stages. Then we take the second last stage and do the same, all up to the first stage. In fact, we could also start with the first stage, doesn’t make a difference.

# Stochastic Process

The randomization is done by changing the order of the stages after assigning each show its own stage.

For this the [Fisher-Yates](https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle#The_modern_algorithm) algorithm is used.

# Selecting the best schedule

Currently, the selection is simply based on the amount of stages needed. However, more factors could be used.

# Showing the schedule

For showing the schedule, I use a Canvas. The class CDrawSchedule draws the necessary elements on the canvas. After that, the canvas is resized. Of course, we can make it more fancier but that was not the assignment 😊.

# Miscellaneous

Some remarks about the implementation:

* Reading the file containing the shows, it kept very simple and does NOT check anything. So a wrong file format will result in a crash.
* Same applies to write the resulting schedule.
* It is possible to read new files and to rerun the scheduling. Helps with testing but also to see the results of changing the number of iterations.
* The get an indication of the performance, a stopwatch was build in. The performance is quite good. With the given example of 30 shows, 10000 iterations only take some 0.3 seconds in debug mode on an i5 processor.

# The Challenge

|  |  |
| --- | --- |
| |  | | --- | | Demcon has decided to expand its activities, and to organize a musical festival. After being an expert in mechatronic systems engineering, the move to the entertainment business seems a rather logical choice.  For the festival 30 acts are hired, such as the Demcon band and other popular acts. Unfortunately, each band has a very tight schedule, and is only able to play at a fixed timeslot. This makes the planning difficult, and the festival organizer needs to know how many stages to prepare for the event.  Your job is to help the festival organization. You are given a list of shows, each with a start and end time. The start and end times are provided as an offset from the start of the festival since the festival goes on non-stop. For example, the first three shows are given like this:  show\_1 36 39 show\_2 30 33 show\_3 29 36  show\_1 is scheduled from 36 hours after festival start and will play for 4 long hours up to (and including) hour 39 after festival start.  It can be seen that show\_1 and show\_3 overlap, since they both play at hour 36. Also show\_2 and show\_3 overlap, so they cannot share a stage.  Your task is to create a planning program which takes the list of shows, and their start and end times, and creates a planning.  Good luck! | |

|  |  |
| --- | --- |
| |  | | --- | | **SPECIFICS & RULES** | |

|  |  |
| --- | --- |
| |  | | --- | | * Please write your solution in Python/C++/Matlab. * Please add information (in the README) on how to execute the code. * Your solution should be able to output a schedule explaining where and when each show will be. How you do this, is up to you. Just know that we do like a good-looking and well-constructed output ;) * We’ve included an example list below for you to get started, but your code should be able to handle any schedule. Additionally, feel free to use another input format or file type, as long as you clearly explain what the input should look like. * We will evaluate your solutions based on inventiveness, efficiency, and good coding practices. * We will only accept submissions in the form of a link to a Github repository. * Please submit to [communication@demcon.com](mailto:communication@demcon.com) * We’ve added some instructions below if you’re unfamiliar with Github. | |

|  |  |
| --- | --- |
| |  | | --- | | **HOW TO USE GITHUB** | |

|  |  |
| --- | --- |
| |  | | --- | | 1. In your internet browser, navigate to [github.com](http://github.com/) 2. Enter your email address, and press ‘sign up for GitHub’ 3. Press continue, create a password and enter a username 4. Solve the puzzle to show that you are a human 5. Verify your email address and customize your Github experience 6. Create a new project by pressing ‘start a project’ 7. Give your repository a name, and make sure it is set to public. Press ‘create repository’ 8. In quick startup, press ‘create new file’ or ‘upload existing file’ 9. Enter the code for this challenge here 10. When you are ready to submit, go to the <>code section of your repository and press ‘code’ and copy the HTTPS link. Enter this link into your submission. | |

|  |  |
| --- | --- |
| |  | | --- | | **EXAMPLE INPUT** | |

|  |  |
| --- | --- |
| |  | | --- | | show\_1 29 33 show\_2 2 9 show\_3 44 47 show\_4 26 30 show\_5 15 20 show\_6 8 15 show\_7 2 9 show\_8 30 34 show\_9 1 9 show\_10 20 28 show\_11 1 4 show\_12 2 11 show\_13 26 29 show\_14 5 10 show\_15 37 44 show\_16 27 35 show\_17 36 39 show\_18 4 10 show\_19 35 44 show\_20 22 30 show\_21 15 20 show\_22 42 46 show\_23 6 9 show\_24 19 23 show\_25 31 38 show\_26 37 41 show\_27 30 36 show\_28 14 21 show\_29 5 13 show\_30 33 36 | |