CSC411 Spring 2020 Homework 4

Due November 2^{nd} at 11:59pm

This assignment includes both conceptual and code questions. It must be completed individually. You may not collaborate with other students, share code, or exchange partial answers. Questions involving answers or code must be emailed to the instructor or TAs directly or discussed during office hours. Your answers to the conceptual questions must be uploaded to Moodle as a pdf file titled `Assign4-<unityid>.pdf`. Your source code must be submitted as a self-contained zip called `Assign4-<unityid>.zip` All code must be clear, readable, and well-commented. You may only use libraries to provide standard data structures or file parsing. All third party library use must be checked with Dr. Lynch or the TA *before* submission.

For all of the questions you have been provided with two problem files that represent planning problems in the PDDL representation. The problem files specify a problem scenario and actions which can be used to complete it.

Question 1 (20 points)

Consider the following activities of a project and their estimated duration in hours. Each has some prerequisite(s) which need to be completed before that activity can begin.

Activity	Prerequisite	Estimated Duration (days)
Start	-	0
C	Start	5
B	Start	9
P	Start	12
A	C, B, P	17
U	P	8
T	A	12
R	A	5
N	U	6
End	T, R, N	12

1. Represent these ordering constraints as a directed graph relating the activities (as shown in the textbook) .
2. Apply the Critical Path Method to this graph to calculate the earliest possible start time (ES) and latest possible start time (LS) of each activity. Also calculate the slack/float of each activity and annotate each node in the graph with [LS, ES] and the duration of the activity as shown in the slides. Show your work.
3. Identify the Critical path and its' duration. What is the total duration of the project assuming no delay over the estimated duration of the activities?
4. Assuming that the activities start at their Earliest start time (ES) but activity P takes five hours longer to finish than estimated. How would it impact the duration of the project? What if instead of activity P, activity R takes half a day longer than the estimated time?

Question 2 (20 points)

For this question generate two *new* planning problems, not shown in the class notes, textbook, or another source, and encode them in the format supplied. Both must generate graphs with at least 3 action layers. One of the problems must result in a viable plan, the other may not. Submit a graphplan graph for each problem in your pdf and include the files in your zip. You may draw them using any suitable tool.

Question 3 (60 points)

In an approved language of your choice implement the GraphPlan algorithm. Your code must:

1. Load the supplied scenario file and print out the contents to the user.
2. Generate the result graph and write it out to a file.
3. Halt when the graph is stable.

Your code should be called as follows:

```
java -jar GraphPlanGenerate.jar <Infile> <GraphFile>

python Assign4/GraphPlanGenerate.py <Infile> <GraphFile>
```

Where `Infile` Is the scenario file being read and `GraphFile` is the output file that stores the result graph. The Graph File should be in the following format:

`StateLayer: <Depth>`
- `Literals:` L_0, L_1,...
- `Negated Literals:` (L_i, L_j)...
- `Inconsistent Support:` (L_i, L_j)...

`ActLayer: <Depth>`
- `Actions:` A_0, A_1,...
- `Inconsistent Effects:` (A_i, A_j)...
- `Interference:` (A_i, A_j)...
- `Competing Needs:` (A_i, A_j)...

Where `Depth` represents the depth of the layer; `Literals` will be a list of the literals that are true or false in the layer; `Actions` are the set of actions in the layer by name; and the remaining lines list pairs of literals or actions that are mutex.

Question 4 (Extra Credit 40 points)

Having implemented the GraphPlan Algorithm you should now extend it to extract a viable plan from the graph using a standard search algorithm or indicate when no plan can be extracted. This extended version of the code should be called as follows:

```
java -jar GraphPlanExtract.jar <Infile> <GraphFile>

python Assign4/GraphPlanExtract.py <Infile> <GraphFile>
```

If a viable plan can be extracted then your algorithm must print the sequence of actions showing where steps can occur in parallel at the end of the GraphFile along with the duration of the plan. If no viable plan can be extracted then you may simply print `"No Plan"`