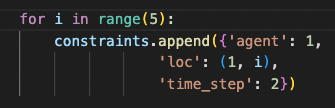
**Project on Multi-Agent Path Finding (MAPF) for the Robotics seminar (236824)**

**Name: Ron Benchetrit**

1. **Task 1: Implementing Space-Time A\***
   1. **Searching in the Space-Time Domain**
2. I have extended the state to support the spatial dimension. In practice this means that a node now also contains a variable “time\_step”.
3. The “time\_step” of the root is zero and for child it is one bigger than its parent.
4. In order to support “wait” actions I have added a fifth option and modified the “move” function.
   1. **Handling Vertex Constraints**
5. In order to support constrains when looping for child nodes we check if it is constrained using the “is\_constrained” function and prune the child If that is the case.
6. A constrains table is initialized for each agent in the “a\_star” function. The table maps from a time step to a list of verex or edge constraints (since each agent may have multiple constraints for each time step) and is implemented by an hash table for fast search.
7. The function “is\_constrained” just chech if a constrain exists in the constrains table. As stated earlier, is is an efficient search because the table is implemented by a hash table.
   1. **Adding edge Constraints**
8. Implemented the same as 1.2 only that now we also support edge constraints.
   1. **Handling Goal Constraints**
9. When initializing the constrains table we also check for the constraint with the latest time step (the maximal key in the hash table) and store it as variable.
10. We changed our goal test condition to check that we have exceeded the time step of the latest constraint.
11. That way we cannot violate any constraints with the cost of a longer search and a longer plan.
12. At time step 10, agent 0 moved the left cell and then came back to the goal at the next time step.
    1. **Designing Constraints**
13. For simplicity and easy implementation, I decided to constrain agent 1 from being at row 1 at time step 2.



This set of constraints forces agent 1 to clear the way for agent 0 such that agent 0 is able to go straight to its goal while agent 1 goes down (south) at time step 2 and then up and right to its goal.

The total sum of cost is 8.

