CSC384 A1 Answers

Name: Cai Lingfeng
Student Number: 1001931573
CDF Account: c4cailio

- **Question1 fuelmaze.py**

1. Heuristics

The heuristics I designed will consider the fuel station. If the current fuel status is not able to reach the goal station, it will compute the manhattan distance from current position to one of the fuel stations, and the manhattan distance from that fuel station to the goal. Finally, plus one cost for fueling. The **minimal** sum will be chosen. It only considers situations where refuel once is needed before reaching the goal.

- *hval=min(hval, man_dis_pos_to_station+man_dis_station_to_goal + 1)*

The heuristics is admissible. When the robot is lacking of fuel, it must go to one of the fuel stations to refuel before going to the goal station. The minimal-cost path (from current position to fuel, and from fuel to goal) will be chosen. Because of factors of the obstacles and potential multiple refuels, the estimate cost is always less than or equal to actual cost.

2. Test cases

- Fuelmaze1 Testcase

Fuelmaze 1, ***Uniform heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 39, Goal state:
<Action="Move Right",index=1138,g=39,h=0,f=g+h=39:S[(X, Y, fuel) = (8,8,9)],(From Node1059)>

Search time = 0.010000000000000002, nodes generated = 1193, nodes expanded = 494, nodes cycle check pruned = 683
==========

Fuelmaze 1, ***Manhattan-Distance heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 39, Goal state:
<Action="Move Right",index=919,g=39,h=0,f=g+h=39:S[(X, Y, fuel) = (8,8,9)],(From Node918)>

Search time = 0.010000000000000009, nodes generated = 922, nodes expanded = 391, nodes cycle check pruned = 498

==========

Fuelmaze 1, ***Custom heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 39, Goal state:
<Action="Move Right",index=906,g=39,h=0,f=g+h=39:S[(X, Y, fuel) = (8,8,9)],(From Node905)>

Search time = 0.01999999999999999, nodes generated = 909, nodes expanded = 384, nodes cycle check pruned = 489

➢ Fuelmaze2 Testcase

Fuelmaze 2, ***Uniform heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 21, Goal state:
<Action="Move Right",index=514,g=21,h=0,f=g+h=21:S[(X, Y, fuel) = (8,8,0)],(From Node480)>

Search time = 0.0, nodes generated = 517, nodes expanded = 195, nodes cycle check pruned = 319
==========

Fuelmaze 2, ***Manhattan-Distance heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 21, Goal state:
<Action="Move Right",index=345,g=21,h=0,f=g+h=21:S[(X, Y, fuel) = (8,8,0)],(From Node344)>

Search time = 0.010000000000000009, nodes generated = 348, nodes expanded = 129, nodes cycle check pruned = 187
==========

Fuelmaze 2, ***Custom heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 21, Goal state:
<Action="Move Right",index=77,g=21,h=0,f=g+h=21:S[(X, Y, fuel) = (8,8,0)],(From Node76)>

Search time = 0.0, nodes generated = 80, nodes expanded = 30, nodes cycle check pruned = 13

➢ Fuelmaze3 Testcase

Fuelmaze 3, ***Uniform heuristic, A* with cycle checking***
==========
Search Successful! (strategy 'astar with full cycle checking') Solution cost = 24, Goal state:
<Action="Move Up",index=910,g=24,h=0,f=g+h=24:S[(X, Y, fuel) = (7,1,6)],(From Node821)>

Search time = 0.009999999999999995, nodes generated = 968, nodes expanded = 343, nodes cycle check pruned = 590
==========

Fuelmaze 3, ***Manhattan-Distance heuristic, A* with cycle checking***

==========

Search Successful! (strategy 'astar with full cycle checking') Solution cost = 24, Goal state:
<Action="Move Up",index=421,g=24,h=0,f=g+h=24:S[(X, Y, fuel) = (7,1,6)],(From Node417)>

Search time = 0.009999999999999995, nodes generated = 423, nodes expanded = 154, nodes cycle check pruned = 216

==========

Fuelmaze 3, ***Custom heuristic, A* with cycle checking***

==========

Search Successful! (strategy 'astar with full cycle checking') Solution cost = 24, Goal state:
<Action="Move Up",index=421,g=24,h=0,f=g+h=24:S[(X, Y, fuel) = (7,1,6)],(From Node417)>

Search time = 0.010000000000000009, nodes generated = 423, nodes expanded = 154, nodes cycle check pruned = 216

My heuristics generally does better when the robot needs to fuel once before reaching the goal. The nodes generated is generally less than uninformed and manhattan-distance heuristics. Because of the small test cases, the search time might in some cases take longer though.

- **Question2 taskscheduling.py**

1. Heuristics

The heuristics will loop in the remaining jobs-to-be-done. For each job-to-be-done, the heuristics will loop in (time, cost) tuples of that job. If current state time plus job duration exceeds the deadlines, the maximum cost for passing the deadline will be chosen as heuristics value for this job. All jobs-to-be-done heuristics values will add up as the total heuristics value.

2. Admissibility Proof

For each job-to-be-done, if the current state time plus job duration exceeds one of the deadlines, the cost of that deadline cannot be avoided at this state, no matter what actions to take. Since the maximum penalty will apply, the maximum cost (if exceeding the corresponding deadline) will be chosen as the heuristics value for this job. The heuristics is always less or equal than the actual cost, because the actual cost will contain this unavoidable cost. The total heuristics is still admissible because it contains all unavoidable cost of all jobs-to-be-done.