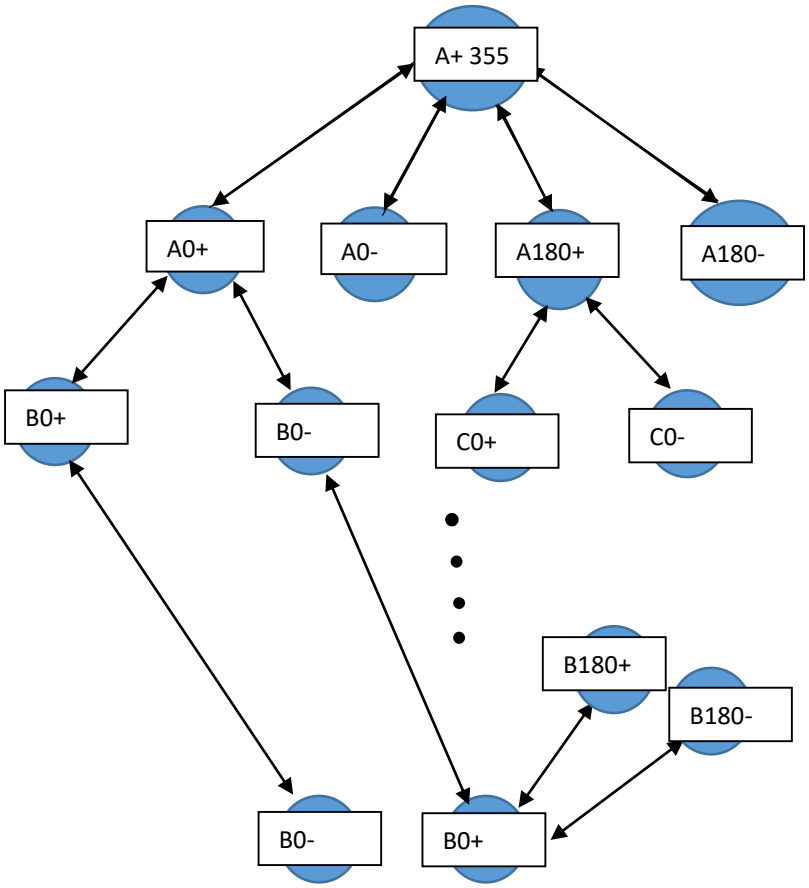
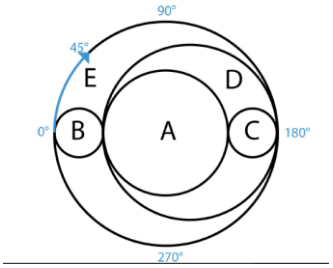
ThisRunMe

I consider every intersection point on a road as a node in a bi-directional cyclic graph. Also a separate node is considered for every orientation. If A has intersections at 0 and 180, then for road A the nodes are A0+,A0-,A180+,A180-. A0+ to A180- implies drone should reverse direction and then move to A180-.

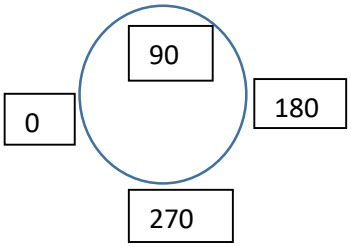




A separate node is created for start and destination.

Two nodes are considered for destination node : for ‘+’ and ‘-‘ as the end direction can be anything based on the chosen small path.

Assigning costs to the edges:

1. The distance functions takes the start node and end node as parameters.
2. 

3. The calculations of distances are performed only for two intersections of same road e.g. A0+ to A180+/A180-. The edge between different roads (essentially an intersection) e.g. A0+ and B180+ is assigned 0 (as it is the same point).
4. Logic:
   CLOCKWISE
   ‘+’ – ‘+’ e.g. A0+ to A180+
   Angle = destination angle- source angle
   180-0 =180
   If angle<360, for e.g. 355 to 360 ,355-360=-5 hence it added with 360 giving angle=5
   Distance is obtained from angle as:
   Dist=(angle)*2*pi*(Radius of road)/360
   If ‘-‘ to ‘+’ then additionally a distance corresponding to 0.3 seconds for reversal is added

   ANTI CLOCKWISE
   Same logic as CLOCKWISE but angle= source angle-destination angle
5. A graph is formed which is the final tree with each node to its intersection nodes with corresponding distances as weights.
   e.g. FinalTree{'A355+': {'A180+': 0.1, 'C0+': 3060.0, 'C0-': 3080.7, 'A}}

Computing Actions and Time:

The drone accelerates at 1m/s^2 till 4m/s and remains constant.

1. The Djiksta Algorithm returns a shortest path based on these edge weights. For e.g. for start:A355+ to end: B0+ it returns: [A355+,A0+,B0+]
2. This path list is taken and pairwise the states are read.
   Algorithm:
1. Start
2. Read the path returned pairwise (node1, node2)
3. If the node 1 is start node: time = (distance-8)/4 + 4
   The drone takes 4 seconds to accelerate from start to 4 m/s during which it covers 8m.
4. If the pairwise nodes are intermediate nodes and their directions are different it needs a reversal:

time+=((dist)/4)+4+4
The drone will have 4m/s velocity which will be brought down to 0 (4 seconds) and accelerated again to 4 m/s. The distance becomes dist+8 when decelerating and that distance is covered during acceleration again.

5. Whenever the two roads are different time is incremented by 0.1 for transfer.
6. The corresponding actions are appended in the actions list
7. Always ‘GO’ action is appended after a TRANSFER

Runme

This code performs the realtime simulation of the drone movement. I had used a Finite State Machine to model the below state diagram. I was able to simulate the drone as the drone as the different states are called for different amount of times. If this code is run it moves the drone from A355+ to B0+ and return the time taken by the drone to complete this. It continuously would display the position of drone.