AI Assignment1 (theory)

Anshuman Suri, 2014021

1. (a) Autopilot System
   - i. PEAS:
     - A. Performance Measure: Safety, Fuel spent, minimize Turbulence, Number of traffic rules broken
     - B. Environment: Air (different zone of the atmosphere, depending on type of aircraft), air traffic, birds, weather
     - C. Actuators: Flaps of wings, tail flap, engines, rear boosters, lights, tires, air pressure stabilizing devices, GPS
     - D. Sensors: Engine rotation speed sensors, barometer, odometer, magnetometer, cabin air pressure sensor, level sensor
   - ii. PAGE:
     - A. Perceives: Engine rotation speed, proximity , weather, temperature , speed, altitude, location, magnetic orientation
     - B. Actions: Increase turbine speed, decrease turbine speed, deploy wheels, change orientation of wing flaps, air conditioning to maintain temperature and air pressure
     - C. Goals: Maintain safety, minimize fuel spent, minimize turbulence, obey air traffic rules
     - D. Environment: Air (different zone of the atmosphere, depending on type of aircraft), air traffic, birds, weather

   (b) Surgery Performing Robot
   - i. PEAS:
     - A. Performance Measure: Blood loss, pain, resources exhausted, time taken, success rate
     - B. Environment: Body on which surgery performed, lighting conditions, doctors helping in surgery
     - C. Actuators: Rotors, pistons, robotic fingers, medical equipment attached to robot, electric pulse machines,etc
     - D. Sensors: Camera, X-ray, blood pressure sensor, pulse sensor, ultrasound sensor
   - ii. PAGE:
     - A. Perceives: blood pressure, pulse, brain activity, blood flow, responsiveness, internal images
     - B. Actions: cut skin, fire laser, patch organs, go deeper, control sensor settings
     - C. Goals: perform surgery in minimum possible time and blood loss, without compromising on success rate of surgery
     - D. Environment: Body on which surgery performed, lighting conditions, doctors helping in surgery

   (c) Amazon Go
   - i. PEAS:
     - A. Performance Measure: Loss of Amazon because of undetected items taken away by customers, minimize cost of setup and its maintenance
     - B. Environment: Signal noise from electronic equipment, people, items already with people
     - C. Actuators: Gates to open only when transaction synced
     - D. Sensors: Depth sensor, cameras, RFID sensors, proximity sensors
   - ii. PAGE:
     - A. Perceives: images, depth, sounds, signals (RFIDs)
     - B. Actions: Close gates on detecting shoplifters, link people with their accounts
     - C. Goals: Minimize discomfort of people hopping, detect shoplifters, minimize maintenance and running cost
     - D. Environment: Signal noise from electronic equipment, people, items already with people

2. This problem can be solved using a greedy search algorithm, where the nodes and edge weights vary per iteration.

   For every missile (which is essentially a node), we maintain the following information:

   (a) It's angle with the positive X-axis, which can be calculated as:
   $tan^{-1}(\frac{y_i}{x_i})$

   (b) It's x and y coordinates at time t, which can be calculated as:
   $(x'_i, y'_i) = (x_i + \frac{y_i}{x_i} * s_i * t, y_i + \frac{y_i}{x_i} * s_i * t)$

   The two cost functions *g()* and *h()* are defined as:

   (a) *g(i)* : if the rotation speed of the cannon is c,
   g(i) = $|\theta_{current} - \theta_i| * c$

(b) *h(x)* : the heuristic function here is the time (if not stopped) for the missile to strike,

h(i)= $\frac{\sqrt{x_i^2+y_i^2}}{s_i}$

Using these two cost functions and summing them up, we run this algorithm. There is a slight change in it however: after the missile defense system has rotated to an angle, that specific node is removed from consideration from the next iteration (as it has been destroyed). The time t and θ_i are re-calculated per iteration, depending on the current state of the cannon. At any point where the function h becomes ¡=0 for any node, we terminate. The number of nodes left at that point subtracted from the number of nodes is the beginning is the required answer.

At every step, we recompute both costs for all nodes, which requires $O(N)$ time per step. In the worst case, all missiles are destroyed. The time complexity of this algorithm thus changes to $O(N^2)$. We need to store information for all states, thus the space complexity is $O(N)$.

3. For generality, let M be the number of cells in the game. The game board can be modelled as a graph, where every cell is a node and every node (except the last 5) has 3 edges (excluding snakes). If we include the snakes as well, we get $O(3M)$ edges from the board moves, and $O(N)$ edges from the snakes, which gives a graph which has $O(M)$ nodes and $O(M+N)$ edges.

   If we pick the first cell as source node, then running a *BFS* algorithm on this reduced graph will return the shortest path from that starting node to any cell in this graph. This is because of the property of *BFS*, which returns the shortest path from the start node to any given node in the graph. Thus running a *BFS* and calculating the shortest path from start to end cell will give the shortest sequence. In case it is not reachable, we can return -1 to indicate that no such path exists.

   The sequence from this graph is equivalent to the game playing, as every edge simulates a possible move from a given node. Moreover, since the reduced *BFS* tree has only one edge between 2 nodes (no cycle either), a node a to a node b can have only one of the moves 2,3,6 as the edge, so it is valid as well.

   The algorithm is thus as follows:

```
Data: Board game size with details of snakes
Result: Shortest sequence from start to end, -1 if none exists
Reduce game board with snakes to graph as described above;
Run BFS on this reduce graph from first cell node;
if path exists to last cell node then
    return Back-tracked path sequence;
else
    return-1;
end
```

   The *BFS* algorithm for a Graph *G(V,E)* has a running time complexity of $O(|V|+|E|)$, and space complexity $O(|V|)$. Since we have a graph with $O(M)$ nodes and $O(M+N)$ edges, the running time complexity is $O(M+N)$ and space complexity of is $O(M)$

4. Assume that $g(t) > h^*(s)+\epsilon$

   That means that the current path to the goal state has at least one node that is part of the optimal path but not part of the current path. Let that node be n

   $f(t) = g(t) + 0$ (as the heuristic takes the value 0 at the goal state)

   Since the node n is not picked before t, it must be because:

   $f(t) <= f(n)$ (because of how A* works)

   $g(t) <= f(n)$

   ie, $g(t) <= g(n) + h(n)$

   ie, $g(t) <= g(n) + h^*(n) + \epsilon$ (as $h(n) <= h^*(n) + \epsilon$) - *(i)*

   Now, because $h^*()$ gives the optimal path cost, we can say that:

   $g(n) + h^*(n) >= h^*(s)$

   ie $g(n) >= h^*(s) - h^*(n)$ - *(ii)*

   This is because otherwise, we could take a path via n which would give a shorter path, contradicting the definition of $h^*()$

   Using *(i)* and *(ii)* ,we get:

   $g(t) <= h^*(s) - h^*(n) + h^*(n) + \epsilon$

   ie, $g(t) <= h^*(s) + \epsilon$

   However, this contradicts our initial claim. Thus, the initial claim must have been false. So, the flowing statement holds true:

   $g(t) <= h^*(s) + \epsilon$