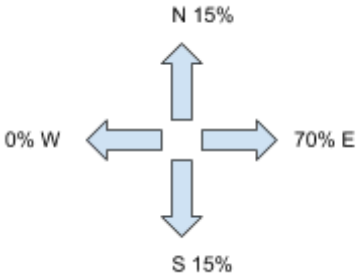
Lab 6: Markov decision process

The task for this assignment is to create a path planner to help a pizza chain deliver food to their customers using drones. As this is a pilot program they started in a small town of size 6 blocks from west to east by 6 blocks from north to south.

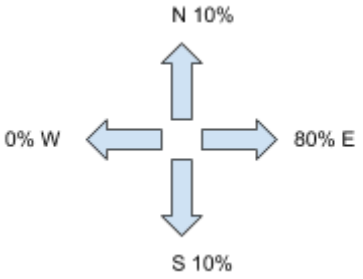
There are a few things you need to be aware of:

- Rival pizza places owns some of the blocks and they will shoot down the drones if they fly over them.
- The company pays to charge the drone battery so the shorter the flight, the better.
- Wind can be strong and the drone not always move in the direction it wants to move.
- All drones come with a special propulsion system that requires extra battery but reduces the impact of the wind in the flight.
- Drones can move only in 4 directions (south, west, north, east).
- Drones can not fall from the map so they bounce against the limits.

Due to the wind, every time the drone attempts to move in one direction it has a 70% chance of succeeding and 30% chance of going sideways (15% each side).
For example: if it decides to go E, it has the following probability associated.



The special propulsion system improves the results as follows but it doubles the battery consumption when it is on.

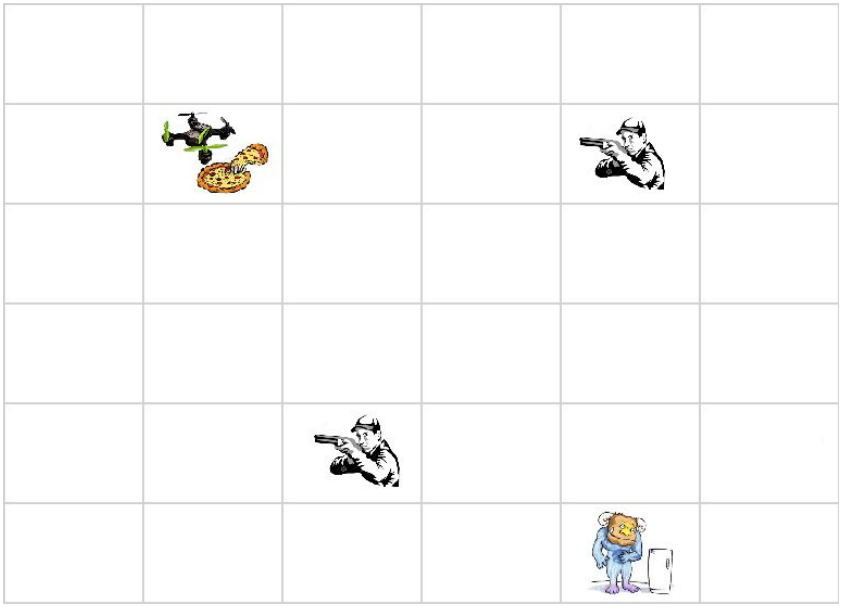


If the drone happens to fly over a block that belongs to a rival company at any given time, we can assume the drone we will be shot down and lost (and that costs the company money).
In addition the planner should discount rewards based on how far in the future they are (i.e. use gamma).

Your program will receive a 6x6 map and must return the recommended actions for the drone in each block.
The values you will find in the map are:

0 - Empty block
1 - The pizza shop
2 - The hungry customer
3 - Rival pizza places

Example: the following map:



Will be represented by the following array:

map[0]= {0,0,0,0,0,0};
map[1]= {0,1,0,0,3,0};
map[2]= {0,0,0,0,0,0};
map[3]= {0,0,0,0,0,0};
map[4]= {0,0,3,0,0,0};
map[5]= {0,0,0,0,2,0};

And the possible actions are:

1 - South with special propulsion OFF
2 - West with special propulsion OFF
3 - North with special propulsion OFF
4 - East with special propulsion OFF
5 - South with special propulsion ON
6 - West with special propulsion ON
7 - North with special propulsion ON
8 - East with special propulsion ON

Problem: Policies
For this task you are to complete one function:

- def drone_flight_planner (map, policies, values delivery_fee, battery_drop_cost, dronerepair_cost, discount)

The function receives:

- A bidimensional structure called map containing the location of all elements.
- An empty bidimensional structure called policies that is required to be filled up with the recommended actions for each block.
- An empty bidimensional structure called values that is required to be filled up with the expected utility of each block under optimal action.
- A float called delivery_fee with the value of the reward for delivering the food to the hungry customer.
- A float called battery_drop_cost is the negative reward for each step the drone is flying with special propulsion OFF (i.e living cost), the negative reward with special propulsion on is 2*battery_drop_cost.
- A float called dronerepair_cost with the cost of replacing the shot down drone.
- An float called discount which is the discount of future values (i.e. gamma)

The function must return:

- The expected utility of the job.

The function must modify:

-The “policies” structure, by entering the optimal policy for each block.

-The “values” structure, by entering the expected utility of each block under optimal policy.

Your code should use python3 and should not import any extra modules

Considerations:

- There are only two exits: getting shot or delivering the food.
- Fees should be considered positive rewards and costs negative rewards.
- Make sure your solution iterates until the values converge (the margin for correctness will be +/-0.1% during grading)
- If there is a tie in values due to actions, give priority to non-special actions, if there is still a tie give priority in the following way: south>west>north>east.