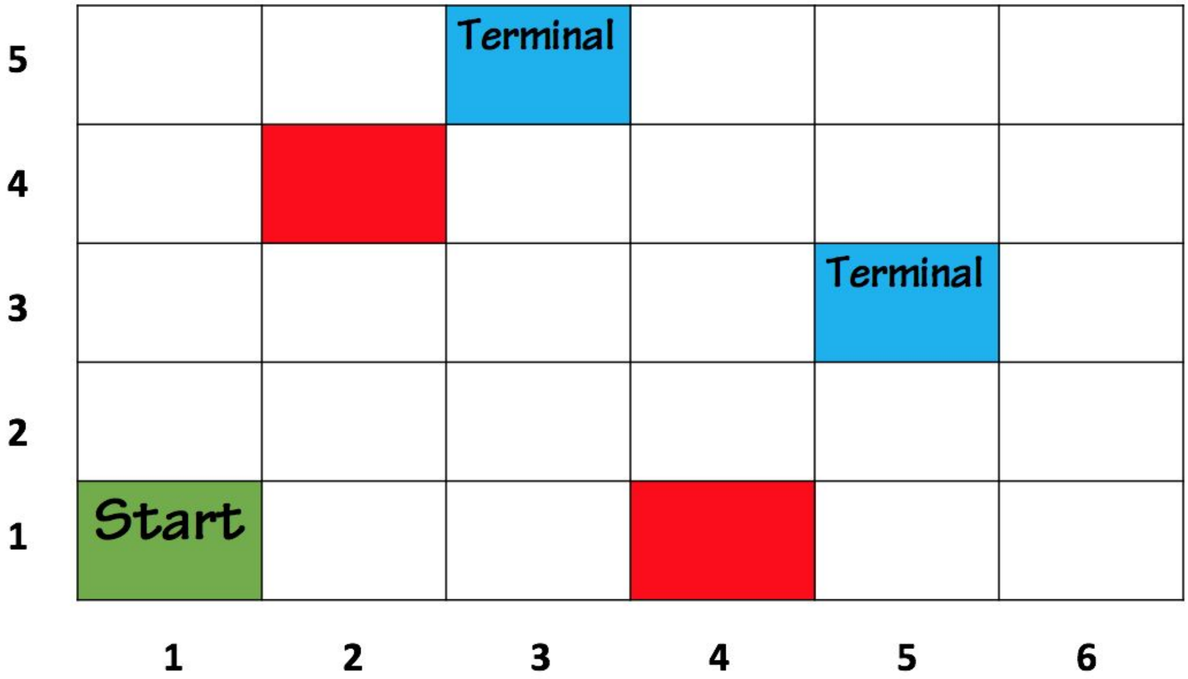
Markov Decision Process (MDPs)

Problem description

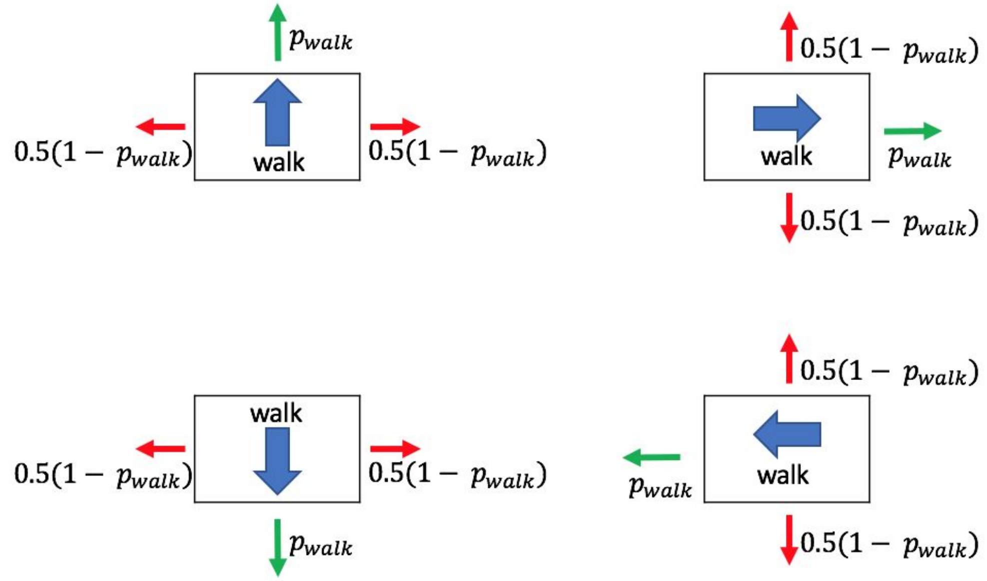
GridWorld is a 2D rectangular grid of size $(N_{rows}, N_{columns})$ with an agent starting off at one grid cell, moving from cell to cell through the grid, and eventually exiting after collecting a reward. This grid environment is described as follows:

- **State space:** GridWorld has $N_{rows} \times N_{columns}$ distinct states. We use s to denote the state. The agent starts in the bottom-left cell (row 1, column 1, marked as a green cell). There exist one or more terminal states (blue cells) that can be located anywhere in the grid (except the bottom-left cell). There may also be walls (red cells) that the agent cannot be moved to. An instance of this grid is shown in the figure below.

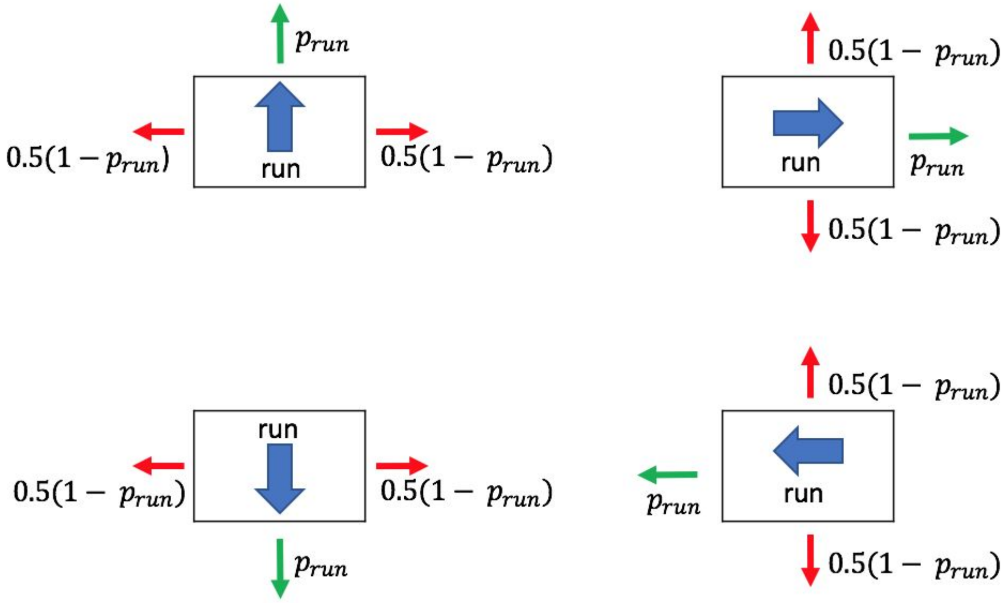


- **Actions:** At every non-terminal state, the agent can either walk or run in any of the four directions (Up, Down, Left, and Right), which results in 8 possible actions: “Walk Up”, “Walk Down”, “Walk Left”, “Walk Right”, “Run Up”, “Run Down”, “Run Left”, “Run Right”. At the terminal state, the only possible action is “Exit”. We use $A(s)$ to denote the set of all possible actions at state s.

- **Transition model**: GridWorld is stochastic because the actions can be unreliable. In this environment, action “Walk *X*” (*X* can be Up, Down, Left, or Right) moves the agent one cell in the X direction with probability p_{walk}, but with probabilities $0.5\,(1-p_{walk})$ and $0.5\,(1-p_{walk})$ moves the agent one cell at angles of 90° and -90° to the direction *X*, respectively.

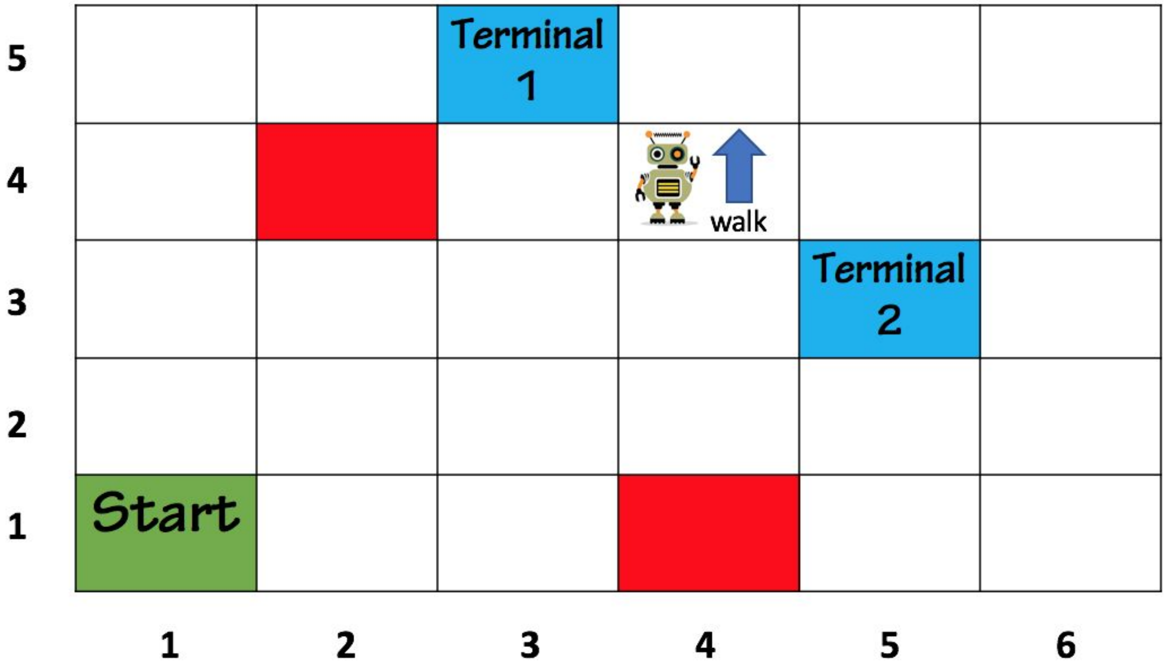


Furthermore, action “Run *X*” moves the agent two cells in the *X* direction with probability p_{run}, but with probabilities $0.5\,(1-p_{run})$ and $0.5\,(1-p_{run})$ moves the agent two cells at angles of 90° and -90° to the direction *X*, respectively.

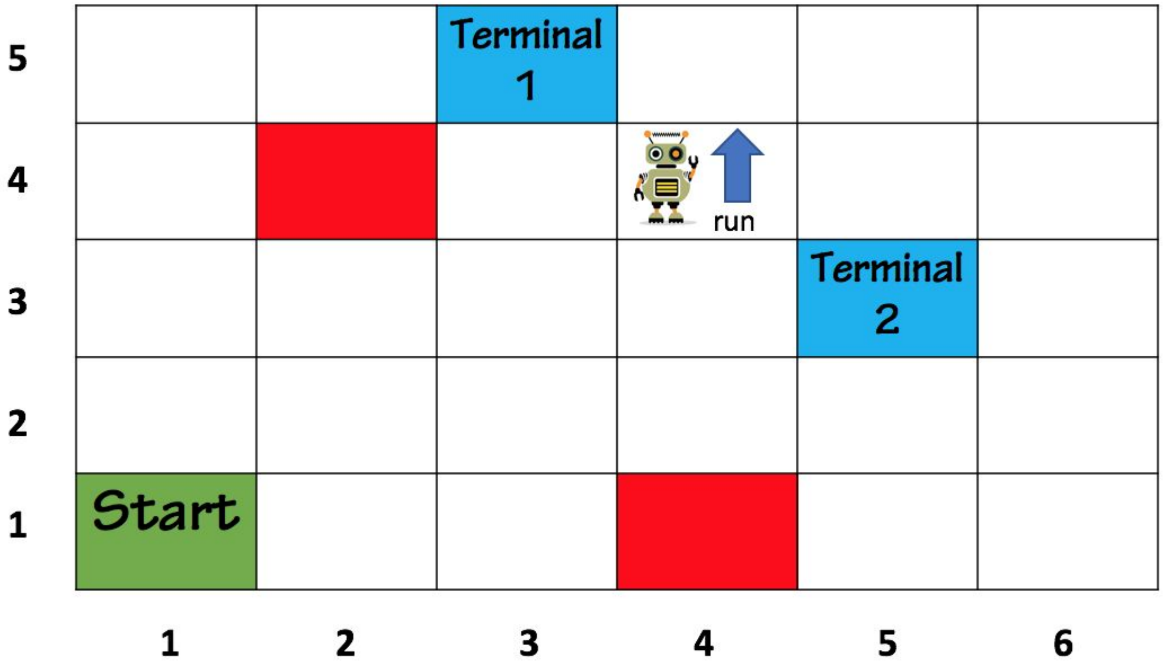


If moving in a particular direction causes the agent to bump into a wall, the movement fails, and the agent stays in the same cell "i,j". We write $P(s'|s,a)$ to denote the probability of reaching state s' if action a is done in state s. The following examples illustrate the environment dynamics:

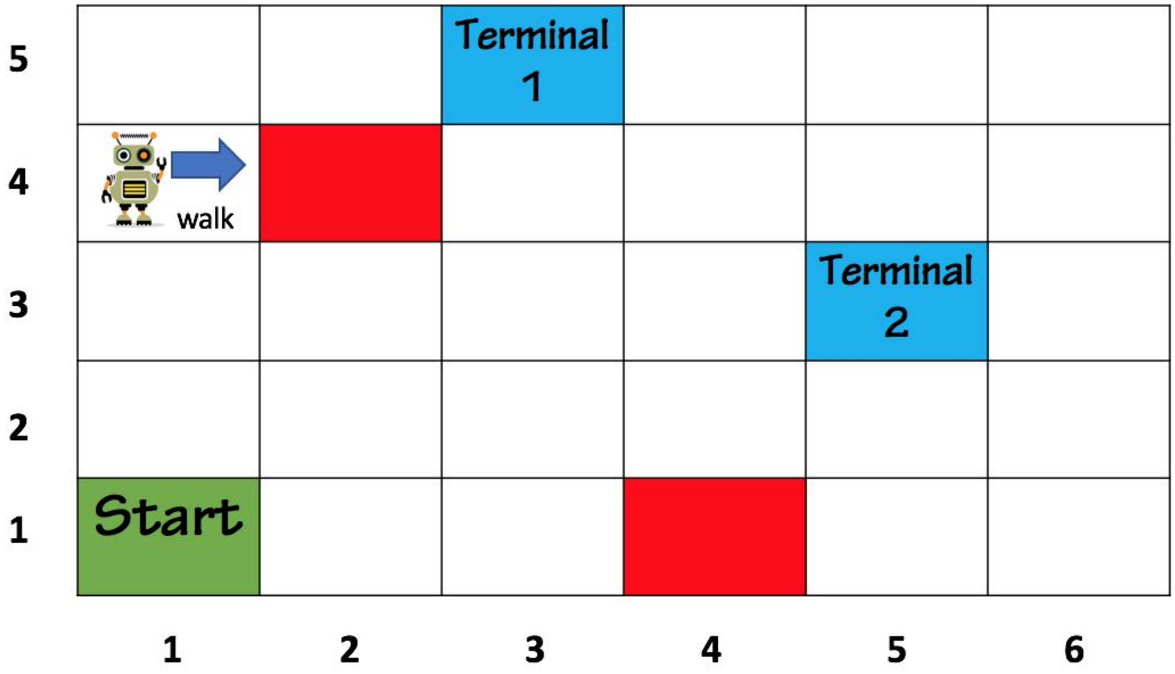
- Assume that the agent chooses action “Walk Up” at “4,4” as shown in figure below. This action moves the agent to (5,4) with probability p_{walk}, but with probability $0.5\,(1 - p_{walk})$, it moves the agent right to “4,5”, and with probability $0.5\,(1 - p_{walk})$, it moves the agent left to "4,3”.



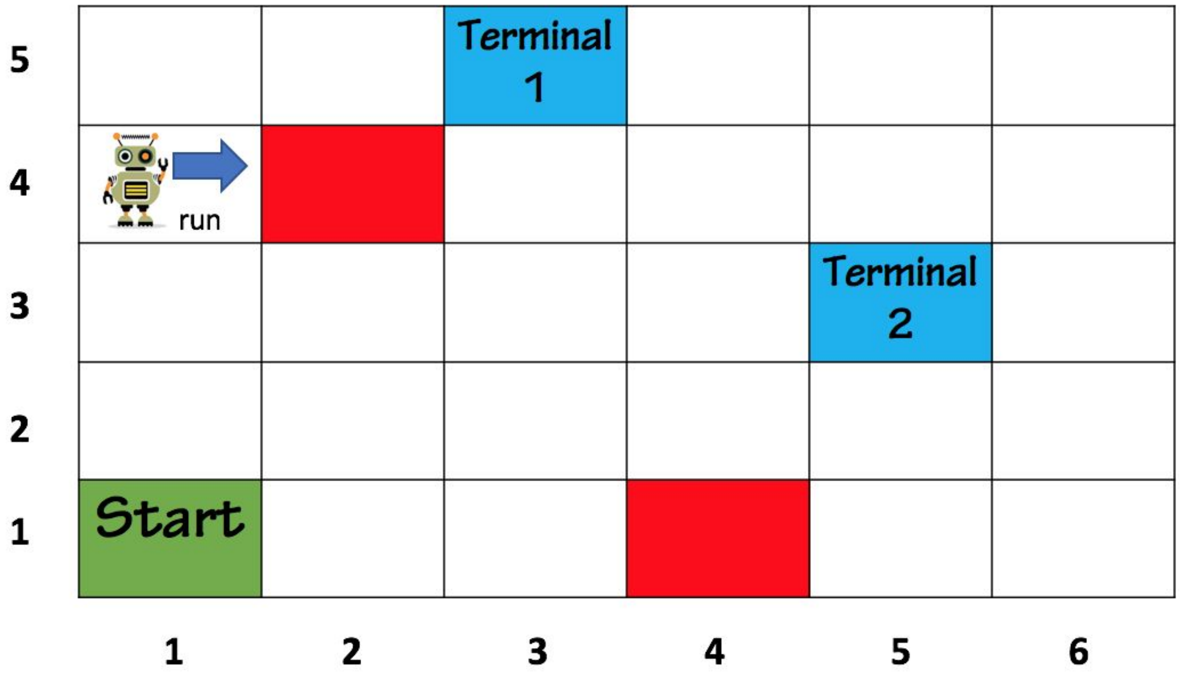
- Assume that the agent chooses action “Run Up” at “4,4” as shown in figure below. This action moves the agent two cells up, but because it causes the agent to bump into a wall, the agent stays at “4,4” with probability p_{run}. With probability $0.5\,(1 - p_{run})$, the agent moves two cells right to “4,6”. Finally, this action moves the agent two cells left with probability $0.5\,(1 - p_{run})$, but because of the wall at “4,2”, the agent stays at “4,4” with probability $0.5\,(1 - p_{run})$. Hence, as a result of this action, the agent moves to “4,6” with probability $0.5\,(1 - p_{run})$ and stays at “4,4” with probability $p_{run} + 0.5\,(1 - p_{run})$.



- ○ Assume that the agent chooses action “Walk Right” at “4,1” as shown in figure below. Then, the agent moves to “5,1” and “3,1”, each with probability $0.5\,(1-p_{walk})$ and stays at “4,1” with probability p_{walk}.



  - Assume that the agent chooses action "Run Right" at "4,1" as shown in figure below. Then, the agent moves to "2,1" with probability $0.5\,(1 - p_{run})$ and stays at "4,1" with probability $p_{run} + 0.5\,(1 - p_{run})$.



- **Rewards:** When the agent takes action *a* in state s, it receives a reward, $R(s, a)$. For all non-terminal states, *s*:
  - *R*(*s,*Walk *X*) = R_{walk} (a constant).
  - *R*(*s,*Run *X*) = R_{run} (a constant).

  Furthermore, if there are K terminal states, the reward in the *k*th terminal state, *s* is *R*(*s,*Exit) = $R_{terminal}(k)$ (a constant).

The agent prefers receiving its reward as soon as possible. Therefore, it applies a discount factor, γ, to future rewards. For example, if it runs once and then exits from the *k*th terminal state, it will receive a total reward of $R_{run} + \gamma R_{terminal}(k)$. If it instead walks twice and then exits from the *k*th terminal state, it will receive a total reward of $R_{walk} + \gamma R_{walk} + \gamma^2 R_{terminal}(k)$.

Problem definition

Finding the action that the agent should take at each state to maximize its expected reward.

Given constraints:

1. Ties between the possible actions are broken based on the following preference order:
   "Walk Up" > "Walk Down" > "Walk Left" > "Walk Right" > "Run Up" > "Run Down" > "Run Left" > "Run Right".
2. The grid will contain at most 1,000,000 cells.
3. As the wall states are never reached, it does not matter what the optimal actions are at these states. However, for consistency, printing "None" as the optimal action for a wall state.

Input File format:

<GRID SIZE> includes two positive integers separated by a comma "," where the first one is the number of rows (N_{rows}) of the grid, and the second one is the number of columns ($N_{columns}$) of the grid.

<WALL CELLS NUMBER> is a number (greater than or equal to zero) that specifies the number of wall cells in the grid.

<WALL CELLS POSITION> contains <WALL CELLS NUMBER> lines where each line includes 2 values separated by a comma ",". The two values in each line indicate the row and column numbers of the gird where a wall is located.

<TERMINAL STATES NUMBER> is a number (greater than or equal to one) that specifies the number of terminal states in the grid.

<TERMINAL STATES POSITION and REWARDS> contains <TERMINAL STATES NUMBER> lines where each line includes 3 values separated by a comma ",". The first two values in the k-th line indicate the row and column numbers of the grid where the k^{th} terminal state is located, and the third value (a float) denotes $R_{terminal}(k)$.

<TRANSITION MODEL> contains two probability values separated by a comma ",", where the first value is p_{walk} and the second value is p_{run}.

<REWARDS> contains two floating-point numbers separated by a comma "," where the first value is R_{walk} and the second value is R_{run}.

<DISCOUNT FACTOR> which is a number in the interval [0,1].

Output File format:

<OPTIMAL ACTION GRID> contains N_{rows} lines where each line includes $N_{columns}$ strings separated by a comma ",". The j-th string at the i-th line should indicate the optimal action (one of "Walk Up", "Walk Down", "Walk Left", "Walk Right", "Run Up", "Run Down", "Run Left", or "Run Right" for non-terminal states; "Exit" for terminal states; or "None" for wall states) at state "$N_{rows} - i + 1, j$".