MI Sheet 2

3.2) • Robot starts in the center of the maze facing north → goal is to get them out.

- Can turn it to face north, east, south, west
- Can direct the robot to move forward a certain distance (stops before hitting a wall)

a) Formulate the Problem. How large is the State Space

IASGP

- The State is (x, y, θ) where x, y signify the location and θ is one of the 4 orientations.
  → State space is as large as $N \times 4$ where N is the no. of locations the robot could be in.
  (4 because there are 4 possible orientations for each)

1. Initial State:
   $(0, 0, 90°)$
   - $0, 0$ → Center
   - $90°$ → facing north

2. Actions(S):
   - turn(θ), $\theta \in \{0, 90°, 180°, 270°\}$
     → turns the robot to face E, N, S, W depending on θ (respectively)
   - Move(d), $d \geq 0$
     → moves the robot $\min(d, w)$ units forward where w is the distance to the nearest wall in the forward direction

both can be executed in any State S.

decided by θ

3. Succesor Function

Describing Result(S,A)

Let $S_t = (x_t, y_t, \theta_t)$ be the Current State

Next State after turn(θ):

$$S_{t+1} = (x_t, y_t, \theta)$$

Next State after move(d):

$$S_{t+1} = (x_t + d\cos\theta_t, y_t + d\sin\theta_t, \theta_t)$$

($d = \min(w, d)$ as described earlier)

4. Goal Test

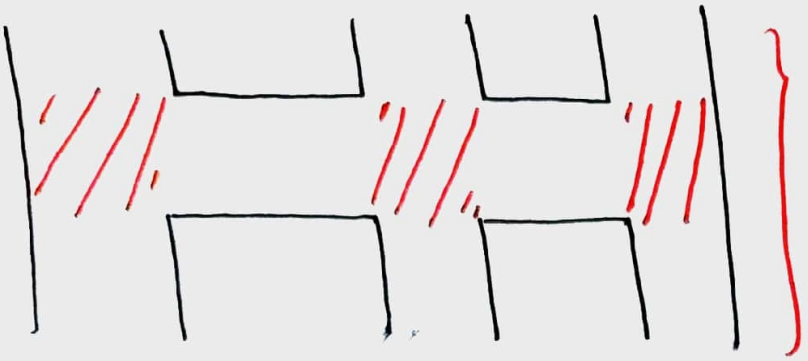
$S_t = G$ where $G = (x_g, y_g, \theta)$ Such $(x_g, y_g) \notin$ Maze

5. PathCost

- The cost Step $C(S_t, a, S_{t+1})$ Can be taken as 1 whenever a is turn(θ) and d whenever a is move(d).
- The Path Cost is the Sum of the Step Costs For all Steps along the Path

b) Reformulate the Problem and State Space Size If we only need to turn at corridor intersections

- Revisit Actions(S) and State that turn(θ) is only Possible if (x,y) of the Current State is an intersection location



- Shaded in red are corridor intersections (anything else is a corridor)
  - * We can be facing any direction in a corridor intersection
  - * Meanwhile in a corridor can only face along it (in both directions) depending on how it was entered (broadly)

Hence, State Space Size is

$4I + 2(N - I)$

- $4I$ → no. of intersection locations
- $2(N-I)$ → Corridor locations

c) Only action we need is to "move in any of the 4 directions until intersection Point is reached" (turning)

→ In this case we can define move() as our only action that moves the agent to the nearest intersection Point in any of the four directions. (and drop the turn action)

- Result(S_t, a) where $S_t = (x, y)$ is $S_{t+1} = (x_i, y_i)$ where (x_i, y_i) is the nearest intersection Point

- We don't need to keep track of the orientation because our move can move in any of the 4 directions regardless to current orientation. It's also not part of the goal test.

d) Simplifications we made while abstracting real world

- Discrete locations
- Discrete orientations
- assumed Perfection actions (turns/moves)
- Ignored if move can take a long time

3.3. Give a Precise Complete Problem Formulation for the Following

a) Using 4 Colors, you have to Color a Planar map so that no adjacent regions share Color

I: ([None, None, ..., None]) // call the list C

none of the regions are initially colored.

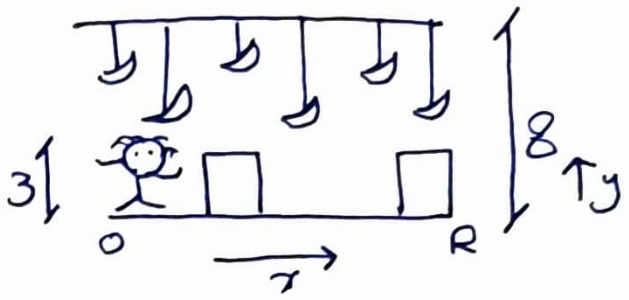
G: C[i] != None For $0 \leq i \leq n-1$
C[i] != C[j] For all i,j corresponding to adj. regions

A: Colorize(i, Color)
→ Colors region i using Color
→ Can be always Performed

S: Result(S, a) when a = Colorize(i, Color)
S = (C)
is (C_n) where $C_n[K] = C[K] \; \forall \; K \neq i$
and $C_n[K]$ = Color Otherwise

P: let C(S, a, S') be always 1. The Path Cost is the Sum of step costs along it.

b) • 3-foot-tall monkey is in a room where bananas are suspended from 8-foot-tall ceiling
• goal is to get some bananas
• room has two stackble, movable, climable 3 foot crates



I: $(0, 0, 0)$ bottom left of the room with 0 banana
x_0 y_0 b_0

G: $b_0 = 1$ (generaly N): Capture a banana

A:

move_l() → // 1 unit left if the position is free
move_r() → ...
Push_Crate() → Push 1 unit left if monkey is next (@ left)
Pull_Crate() → ...
Climb_Crate() → Climb Crate if monkey is next to it
Stack_Crate() → Stack them if ... and they're next to...
grab_banana() → grab banana if there's one along monkey's height
...

S: Skipped due to too many details

P: Let $C(S, a, S') = 1$ So Path Cost is no of actions

c) • We have access to a Program that takes a pile of input records and outputs illegal input record if any of them is illegal
→ Want to discover which record is illegal

I: All input records in a suspected list (L) and Program output (P) as "illegal"

A: • Send a subset L' of the records (Len(L)-1) to the Program & get its output (a)

S: Result(S, a)
→ when the action a is done while L has a record or more and P is illegal or legal
. update P with the programs output
. if P is legal, set L = L - L'
else, set L = L'

G: L has size 1 (that must be the illegal record)

P: Cost is 1 per action $\xrightarrow[\text{Sum}]{}$ Path cost

d) Three jugs measuring 12, 8, 3 gallons and a water faucet. (capacity[])

→ Can empty jugs out from one to another or onto the ground

→ Can fill them up from the faucet

• Need to measure exactly one gallon.

I: (Content) where Content = [0, 0, 0] (all jugs empty)

G: There exist $j \in \{0,1,2\}$ where Content[j] = 1

A:

Action	Condition
• Fill jug j from faucet	• Content[j] < capacity[j]
• Empty jug j into jug K	• j ≠ K, Content[j] > 0, Content[K] < capacity[K]
• Empty jug j into ground	• Content[j] > 0

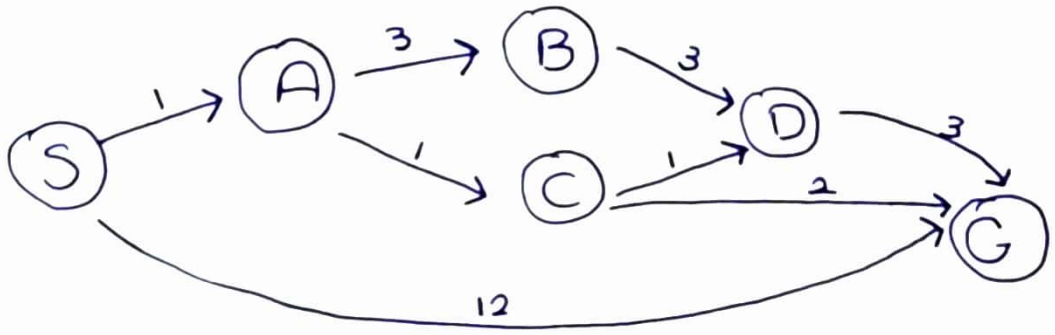
(Global State)

S:

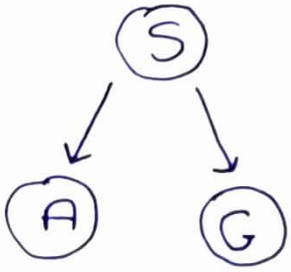
Action	newState
Fill_Faucet(j)	Content[j] = capacity[j]
Empty_Jug(j,K)	...
Empty_Jug_Ground()	Content[j] = 0

PathCost: Sum of StepCosts where StepCost is 1 ∀ actions (1+)

- Apply BFS, DFS, UCS to



BFS:



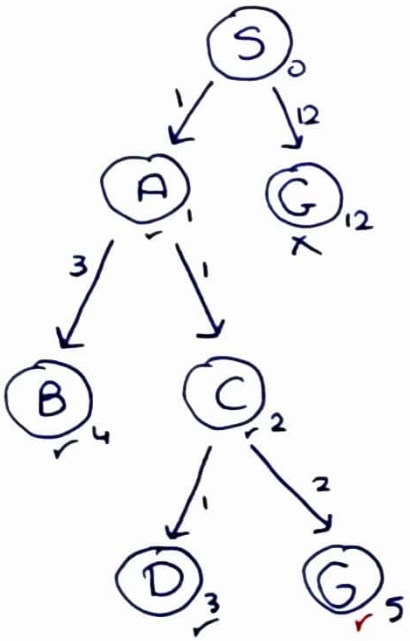
- Frontier = [S]

Frontier = [A] then before inserting G goal test is successful

- Explored = [S]

- Solution Path: S → G

UCS:



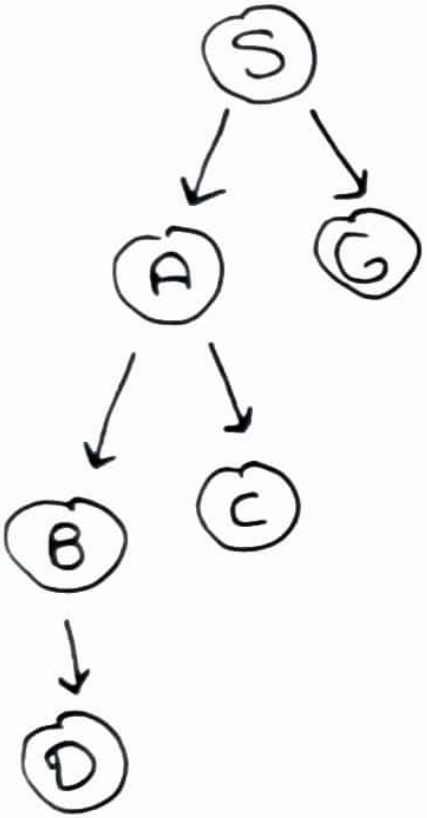
Expand	Frontier
	[S]
S	[A, G]
A	[C, B, G]
C	[D, B, G]
D	[B, G]
B	[G]
G	Done.

explored list

Solution Path: [S → A → C → G]
Cost: 4

DFS:

→ will assume graph search and goal test after Frontier exit



	Expand	Frontier
		[S]
Explored list	S	[A, G]
	A	[B, C, G]
	B	[D, C, G]
	D	[C, G]
	C	[G]
	G	Done

Sol. Path: S → G