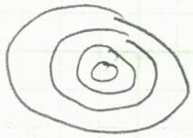
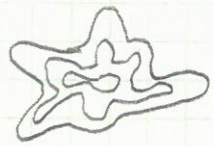


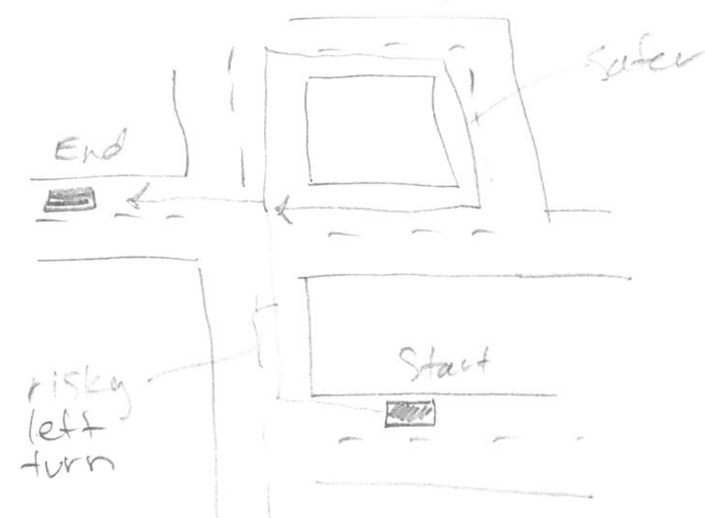
Bread First Search - explores equally in all directions. It is incredibly fast and explores everywhere



Dijkstra's Algorithm - also called uniform cost search, prioritizes which path to search. However, instead of exploring equally, it looks at the cost of taking a certain route and chooses the cheapest first

A* - a modification of Dijkstra's algorithm, but is optimized for a single destination. It uses a heuristic to drive it

Motion Planning

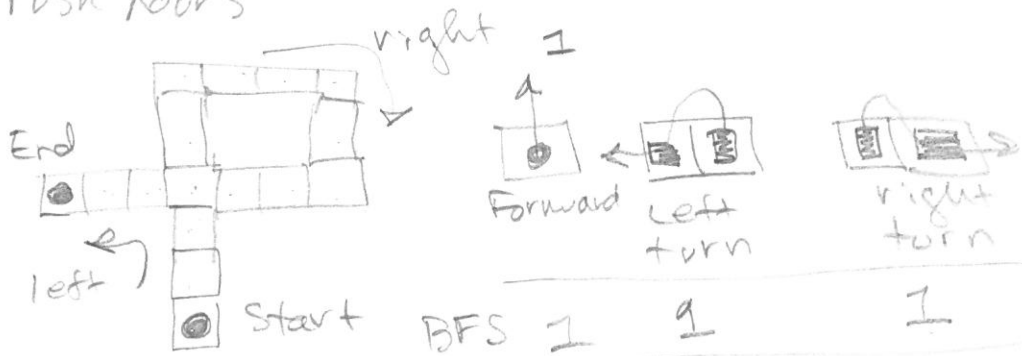


BFS or Dijkstra's Alg (DA) L 2

Give: Map
Goal, Start loc
Cost function

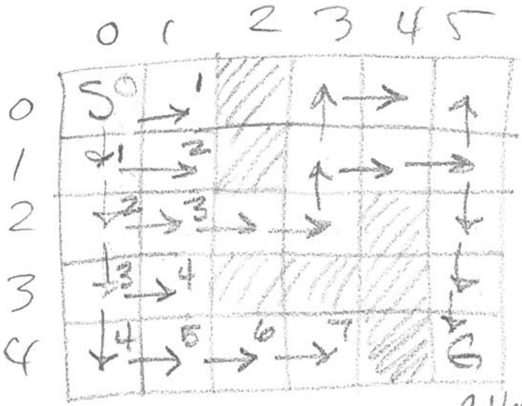
actions
move:
turn:

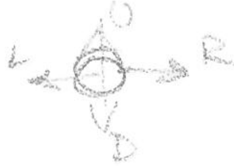
* Fedex, Ups, etc plans routes w/ right turns during rush hours



	Forward	Left turn	Right turn	Cost Left	Cost right
BFS	1	9	1	6	16
DA	1	10	1	15	16
	1	20	1	25	16

- Both BFS + DA give us solutions
- BFS causes us to make a dangerous or time consuming left turn in heavy traffic
- DA must be tuned to avoid the left turn
  - there maybe situations where a left turn is ok or better ... it depends
- Both of these explore the whole map
  - for a static environment, you could precompute all of this





Path Plan - shortes sequence

Cost : 11 actions

gValue ~ gives us the shortest path

not g

open = [0,0] 0

frontier → [1,0] 2 [0,1] 1

↓ expand down and check if goal

[2,0] 2 [1,1] 2

[3,0] 3 [2,1] 3

[4,0] 4 [3,1] 4

[4,1] 5

[4,2] 6

[4,3] 7

[x, y] cost_value

f_value - a function that returns the cost of the move, this will be more complex w/ A*

f_value = cost_value

BFS or DA

1. Expand frontier
   - move in all possible directions
   - calculate cost of move
   - stop when there is no longer any place to explore
2. start from goal and move to start taking the lowest cost path

* Both explore entire enviornment from the start and expand outward in all directions

A* is Similar
A* always expands to goal

- Heuristic Function (h) - often gradient decent

A → B
4 steps
check it!

A 4	3	2
3	2	1
2	1	B 0

← optomistic guess to goal w/o obsticals

$h(x,y) \leq$ distance to goal from x,y

* Doesn't have to be accurate, just guide the robot
* If it was accurate, then you probably already solved it

open [0,0] g-value f-value

f-value = g + h(x,y)

now look @ this when moving

open [4,1] 5 9 g h
g f = 5+4

expand node lower f-val and closer to goal

[4,2] 6,9
[3,2] 7,11 [4,3] 7,9
[3,3] 8,11 [4,4] 8,9 ← this will take you goal
exp
Note: don't expand the open area

Dynamic Programing

Given Map + Goal
Outputs Best path from anywhere

has policy $x, y \rightarrow$ action
- thus if you find yourself somewhere other than where you thought, you know how to navigate
- Good in dynamic environment
- every grid pt tells you direction to go

value function (f) for each cell

$f(x,y) = \min f(x', y') + 1$

hill climbing action

5	4	3	2	
6		2	1	
7		1	0	Goal
Start				

A* Summary

Cost

2	1	G 2
1	S 0	1
2	1	2

hueristic

2	1	G 0
3	S 2	1
3	3	2

hueristic would never let you search in the shaded cells

f_value

4	2	G 2
4	S 2	2
5	4	4

both solutions are the same