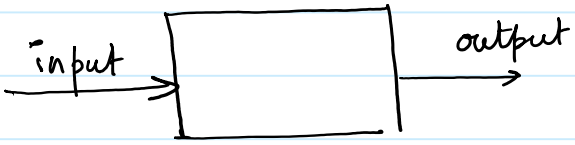
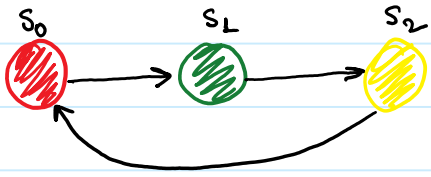
Systems



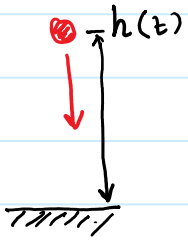
State: It says what is the current condition of the system.



Finite state system: A system with finitely many states.

infite state system:

The state is the height of the free falling ball.



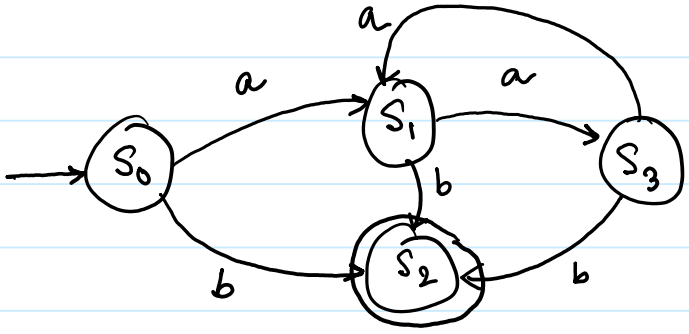
There are infinite possibilities.

3.1, 3.100029, ...

Finite state machines: denoted by $(Q, \Sigma, \delta, q_0, F)$

Q is the set of states.
Σ " " " " action/input
δ " " transition relation that dictates how one state moves to another upon applying an input

q_0 is the initial state.
F is the final state(s).



$Q = \{S_0, S_1, S_2, S_3\}$, $\Sigma = \{a, b\}$
$q_0 = \{S_0\}$, $F = \{S_2\}$

δ (current state, action) = next state

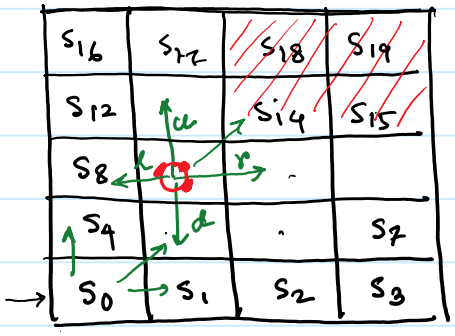
$\delta(S_0, a) = S_1$
$\delta(S_0, b) = S_2$

$\delta(S_1, a) = S_3$
$\delta(S_1, b) = S_2$

$\delta(S_2, a) = \phi$
$\delta(S_2, b) = \phi$

$\delta(S_3, a) = S_1$
$\delta(S_3, b) = S_2$

Robotics Path Planning:



$Q = \{s_0, \ldots, s_{19}\}$

$q_0 = \{s_0\}$

$F = \{s_{14}, s_{15}, s_{18}, s_{19}\}$

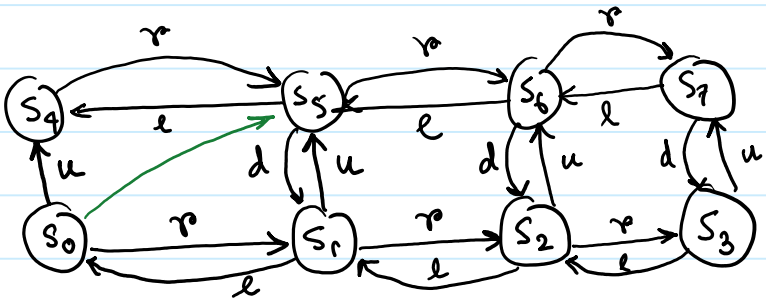
$\delta(s_0, u) = s_4$

$\delta(s_0, r) = s_1$

$\delta(s_1, u) = s_5$

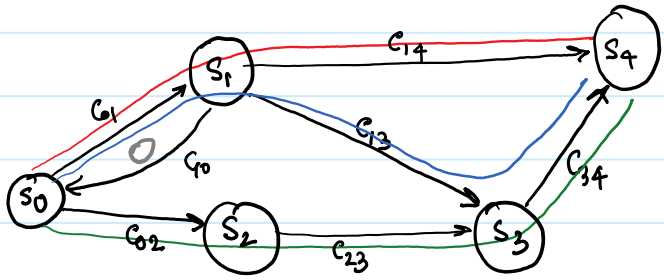
$\delta(s_1, r) = s_2$

$\delta(s_1, l) = s_0$



next \ current	s_0	s_1	s_2	s_3
s_0	X	l	X	X
s_1	r	X	l	X
s_4				
s_5				

Graphs and Network:



find the shortest ("least cost") path from one node to another.

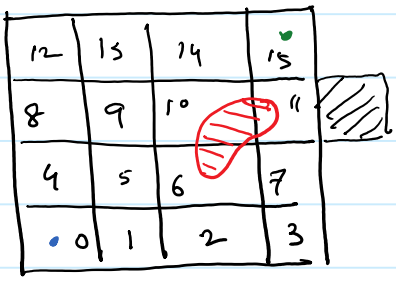
Example: find path from s_0 to s_4

	cost
$s_0\ s_1\ s_4$	$c_{01} + c_{14}$
$s_0\ s_1\ s_3\ s_4$	$c_{01} + c_{13} + c_{34}$
$s_0\ s_2\ s_3\ s_4$	$c_{02} + c_{23} + c_{34}$
$s_0\ s_1\ s_0\ s_2\ s_3\ s_4$	
$s_0\ s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_4$	
⋮	

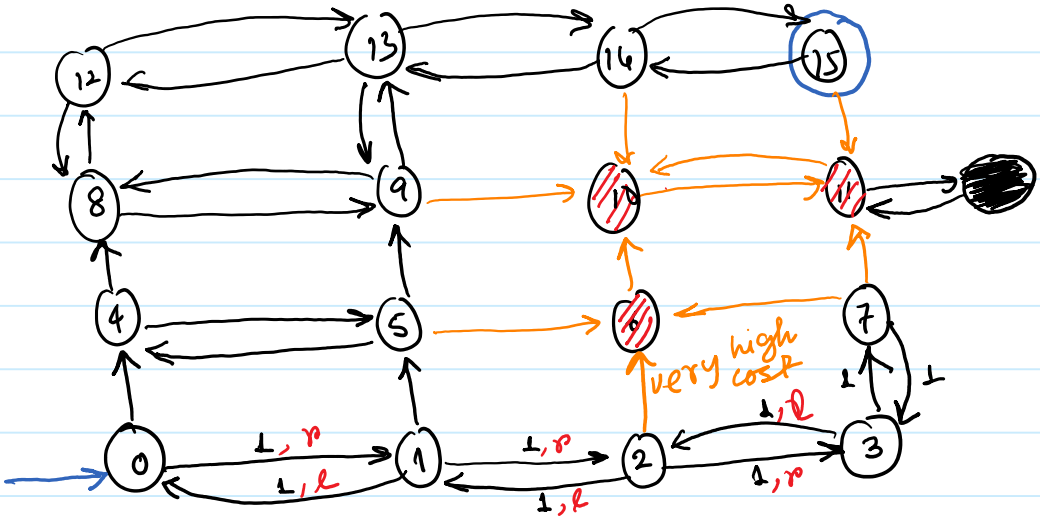
Dijkstra's algorithm for shortest path planning:

A^* is widely used for path planning in robotics.

Robotics path planning using Dijkstra's:

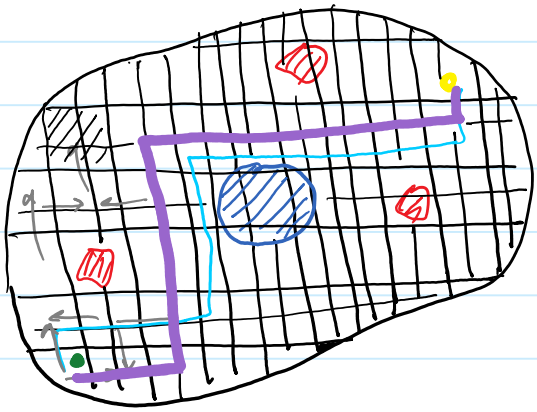


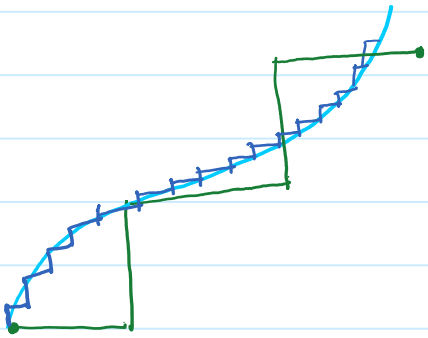
Step: find a FSM equivalent → that gives the graph

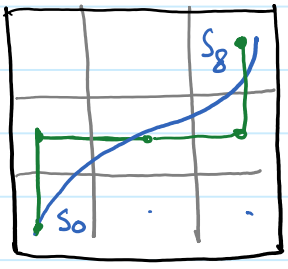


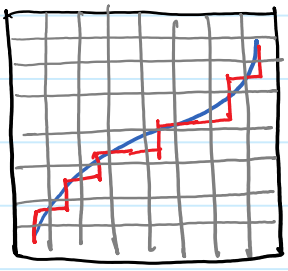
Find a path from s_0 to s_{15}

Problem:









✓

This has only 9 states

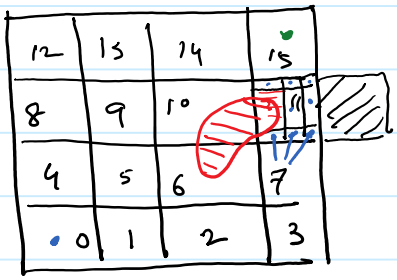
✓

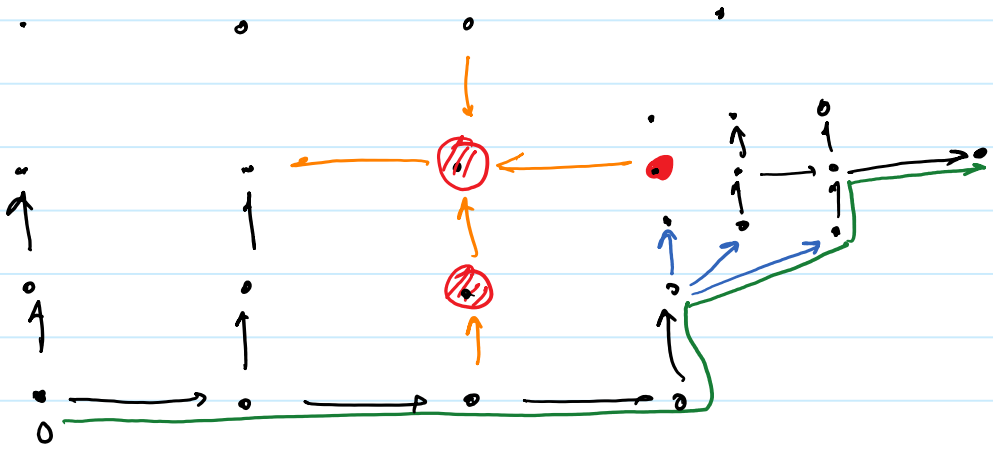
this resolution gives better path but it has too many states

Computation and optimality trade-off:

Higher resolution ⟹ better path, but high computation

Lower resolution ⟹ coarser path, but less time to compute.


12
13
14
15
8
9
10
11
4
5
6
7
0
1
2
3


0