COMP 472: Artificial Intelligence
State Space Search part #3
State Space Representation video #1

- Russell & Norvig – Sections 3.1-3.3

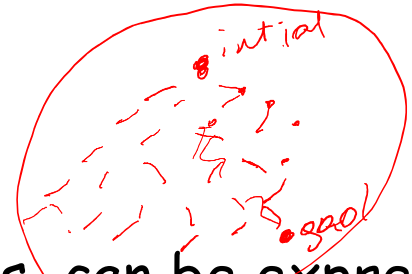
Today

1. State Space Representation

video #1

2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

Motivation



1970

- Many AI problems, can be expressed in terms of going from an initial state to a goal state
  - *Ex: to solve a puzzle, to drive from home to Concordia…*

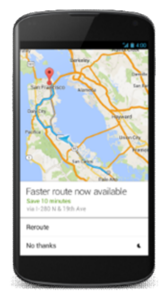
initial

goal

8-puzzle

15 moves

200 moves



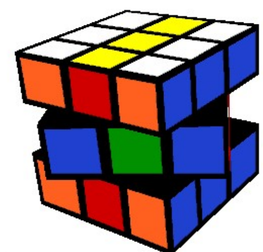
Google Maps

home

left right

straight right left

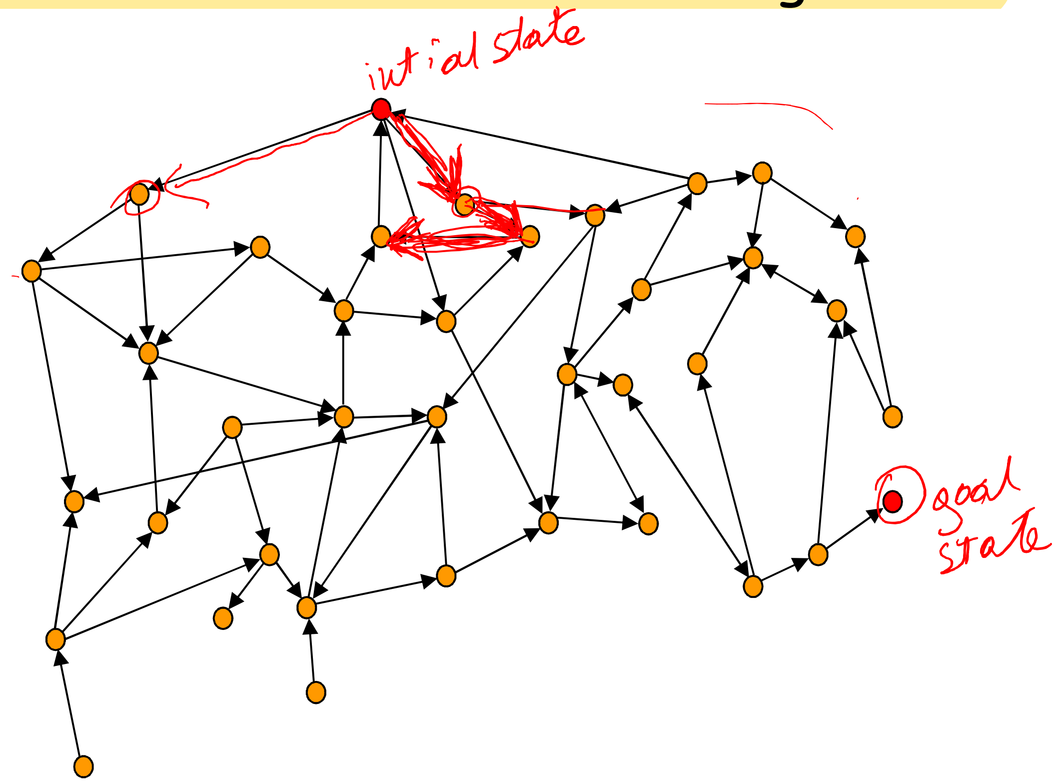
Concordia



Rubik's cube

Motivation

- Often, there is no direct way to find a solution to go from the initial state to a goal state
- but we can list the possibilities and search through them



Example: 8-Puzzle

State: Any arrangement of 8 numbered tiles and an empty tile on a 3x3 board

8	2	empty
3	4	7
5	1	6

Initial state

1	2	3
4	5	6
7	8	

Goal state

there are several standard goals states for the 8-puzzle

1	2	3
4	5	6
7	8	

1	2	3
8		4
7	6	5

...

(n^2-1)-puzzle

8	2	
3	4	7
5	1	6

8-puzzle

4

4

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

15-puzzle

....

15-Puzzle

Invented in 1874 by Noyes Palmer Chapman …

but Sam Loyd claimed he invented it!

Sam Loyd

see Sam Loyd's book of puzzles:
https://archive.org/stream/CyclopediaOfPuzzlesLoyd/Cyclopedia_of_Puzzles_Loyd#mode/2up

https://en.wikipedia.org/wiki/Sam_Loyd

State Space

- Problem is represented by:
  1. Initial State
     - starting state
     - ex. unsolved puzzle, being at home
  2. Set of operators / moves / actions
     - actions responsible for transition between states
  3. Goal test function
     - Applied to a state to determine if it is a goal state
     - ex. solved puzzle, being at Concordia
  4. Path cost function
     - Assigns a cost to a path to tell if a path is preferable to another

	cost	cost
up	1	3
down	1	1
left	1	2
right	1	4
	least nb of moves	lowest cost

goal(...) → T

→ F

goal(...) = T

- State space:
  - the set of all states that can be reached from the initial state by any sequence of action

initial

comp 352

- Search algorithm:
  - how the search space is visited

Example: The 8-puzzle

goal

8	2	
3	4	7
5	1	6

= false

1- Initial state

goal

1	2	3
4	5	6
7	8	

= True

Goal state

2- Set of operators:
  blank moves up, blank moves down, blank moves left, blank moves right

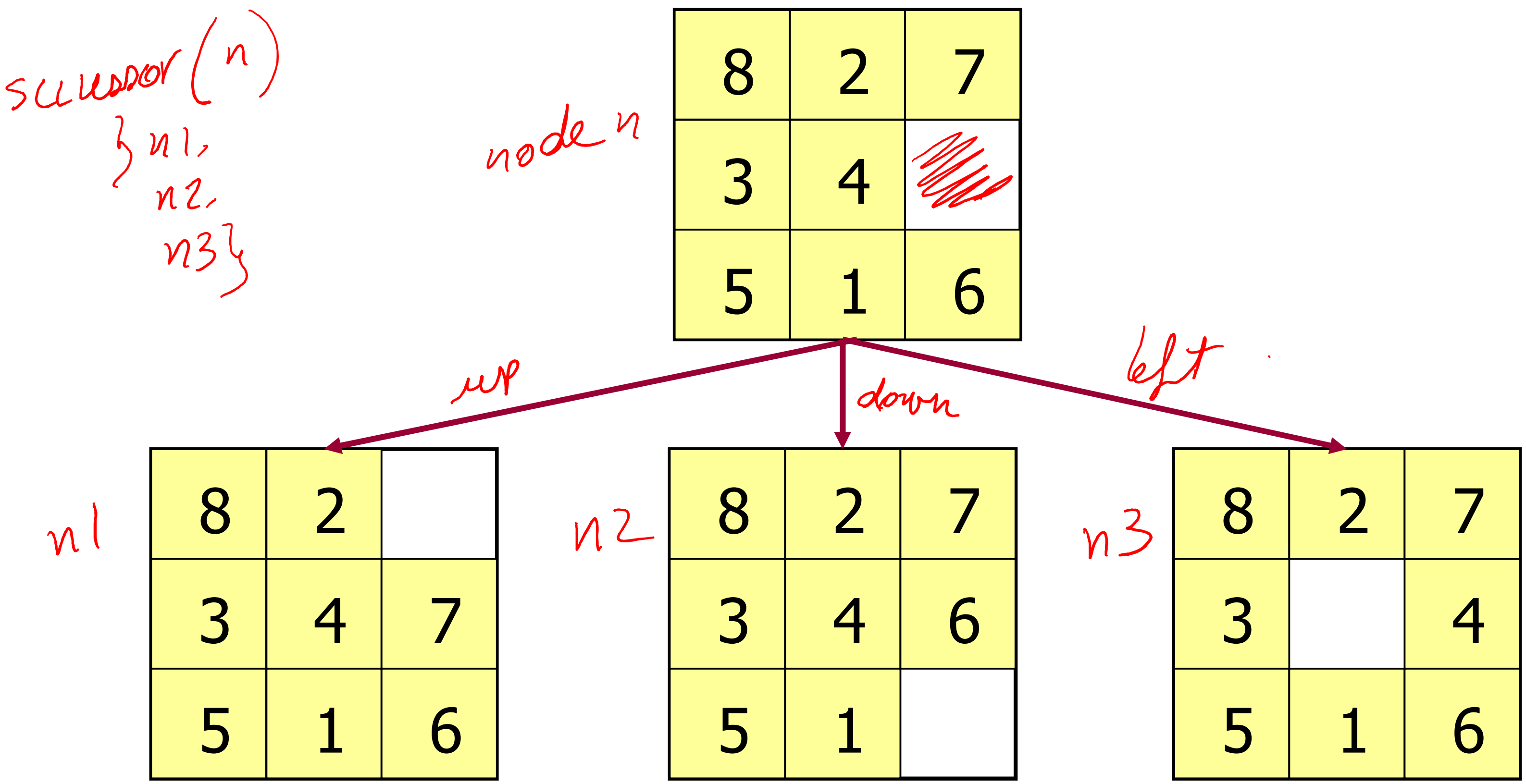
3- Goal test function:
  state matches the goal state

4- Path cost function:
  each movement costs 1
  so the path cost is the length of the path (the number of moves)

source: G. Luger (2005)

8-Puzzle: Successor Function

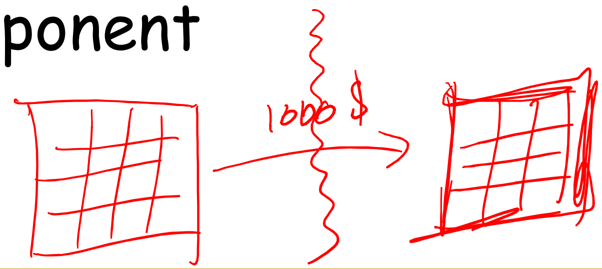
{children

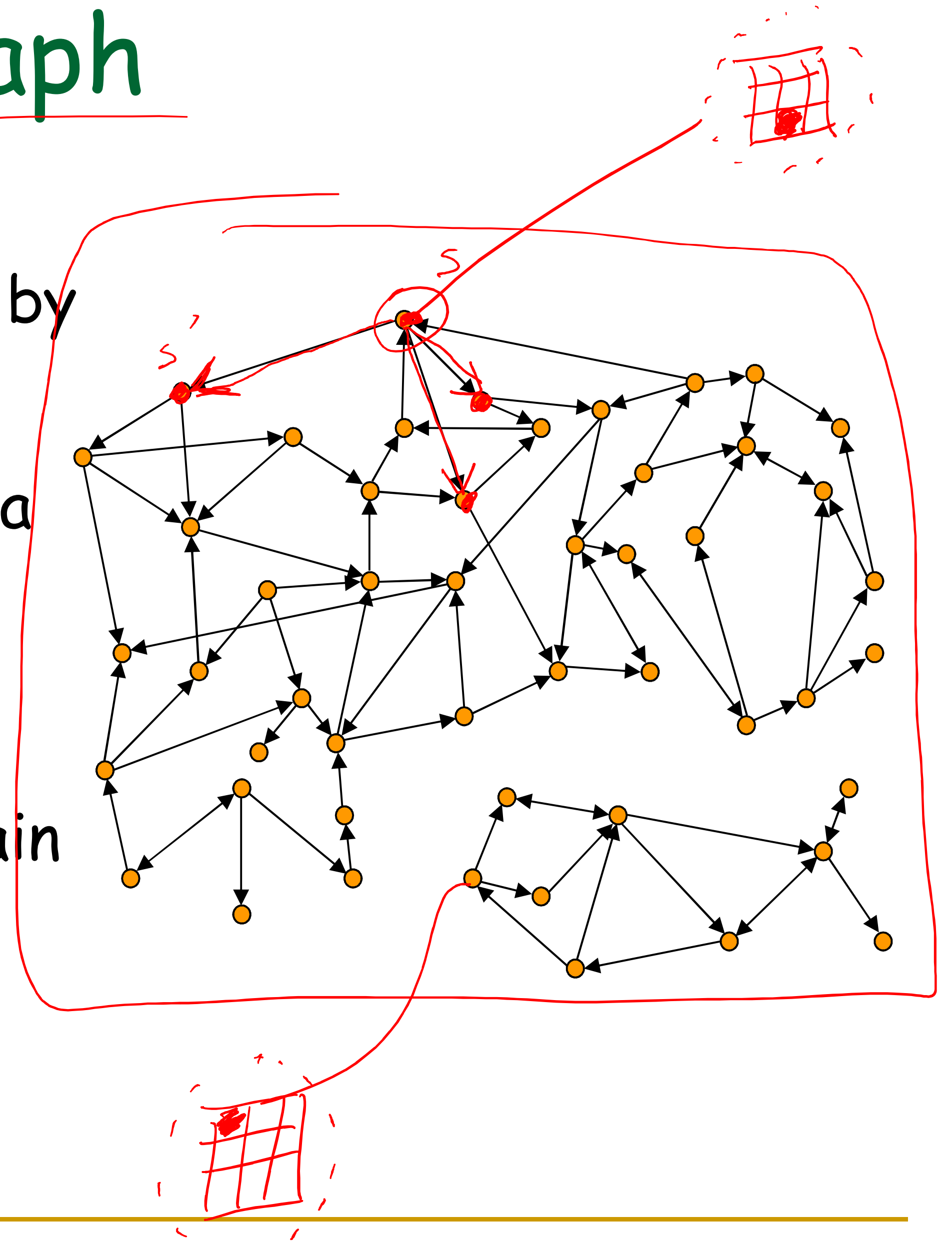


Search is about the exploration of alternatives

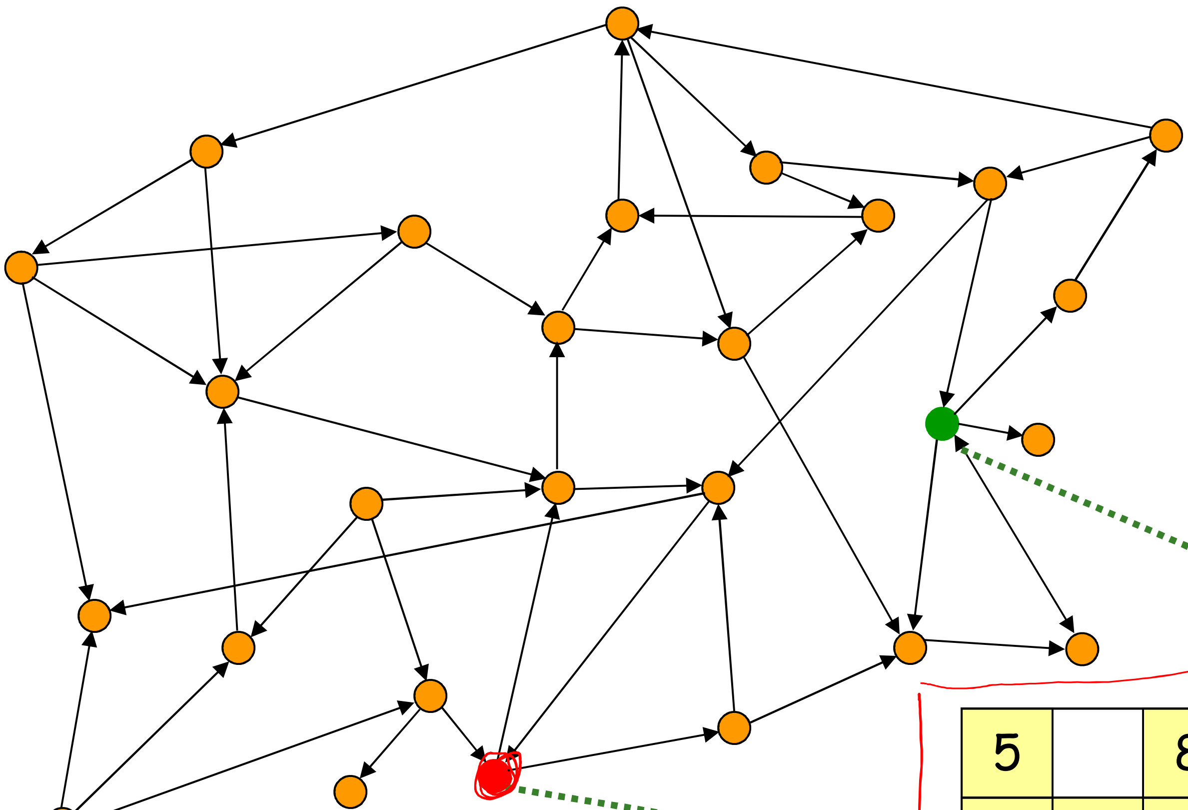
State Space Graph

- Each state is represented by a distinct node
- An arc (or edge) connects a node s
  to a node s' if
  s' ∈ SUCCESSOR(s)
- The state graph may contain more than one connected component





Just to make sure we're clear…



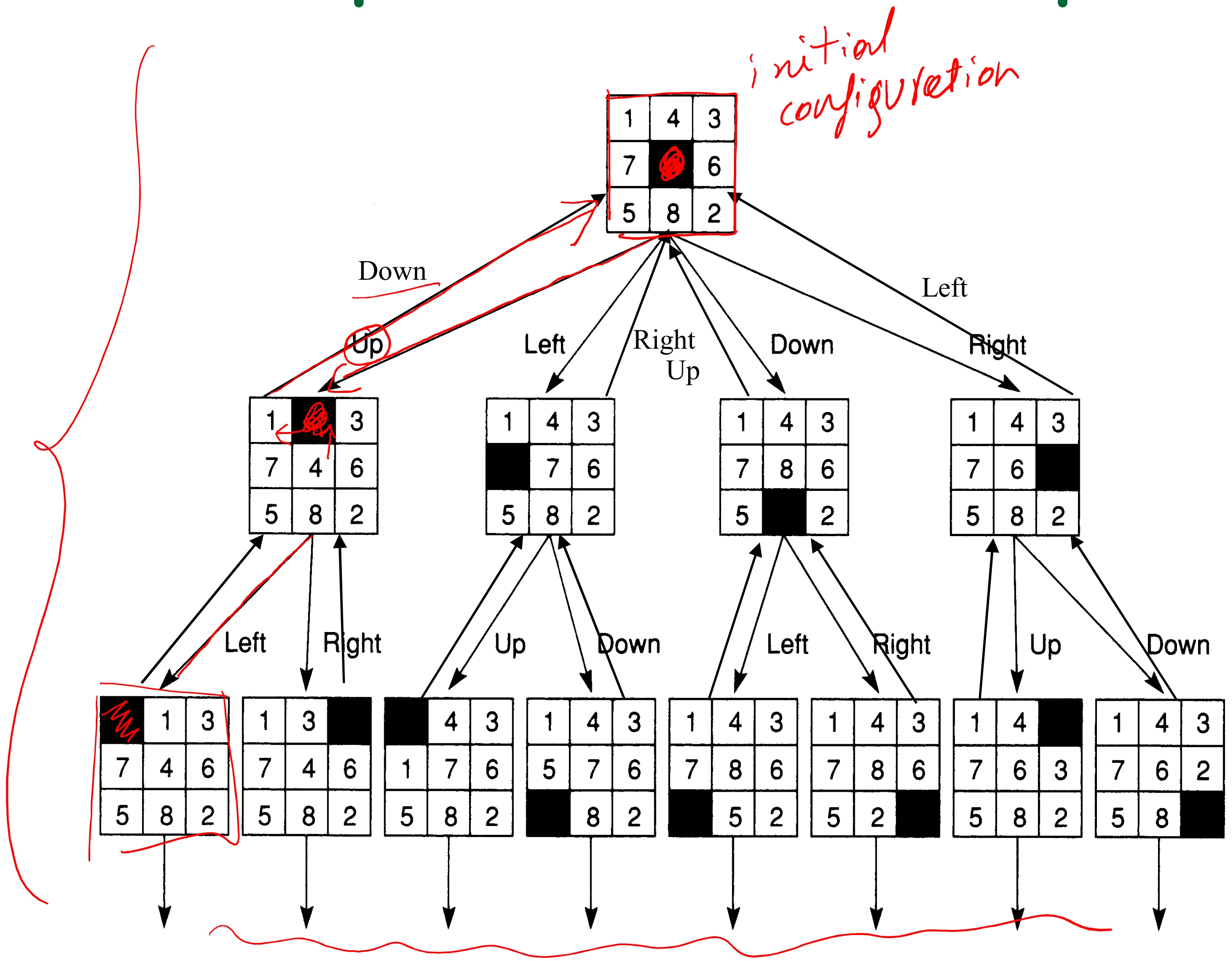
5		8
4	2	1
7	3	6

Initial state

1	2	3
4	5	6
7	8	

Goal state

State Space for the 8-puzzle



source: G. Luger (2005)

Size of state spaces

- For the (n-1)-puzzle:
  - Nb of states:
    - 8-puzzle --> 9! = 362,880 states
    - 15-puzzle --> 16! ~ 2.09 x 10^{13} states
    - 24-puzzle --> 25! ~ 10^{25} states
  - At 100 millions states/sec:
    - 8-puzzle --> 0.036 sec
    - 15-puzzle --> ~ 55 hours
    - 24-puzzle --> > 10^9 years

5

5

goal(state) → T
→ F

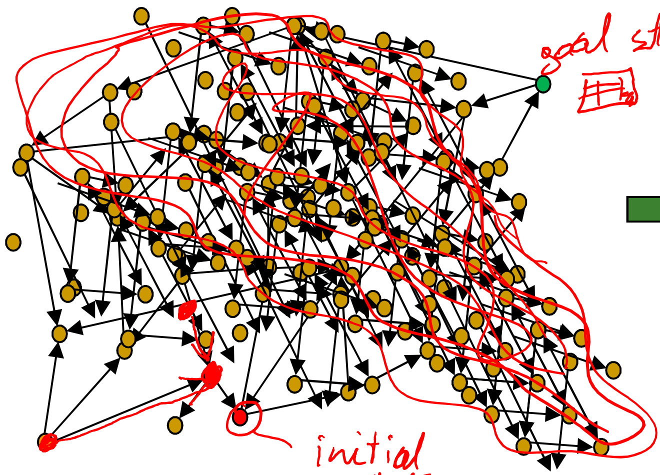
- For real problems:
  - state spaces are way too large to be generated in advance and searched after
  - so it is generated dynamically <u>while</u> searching.

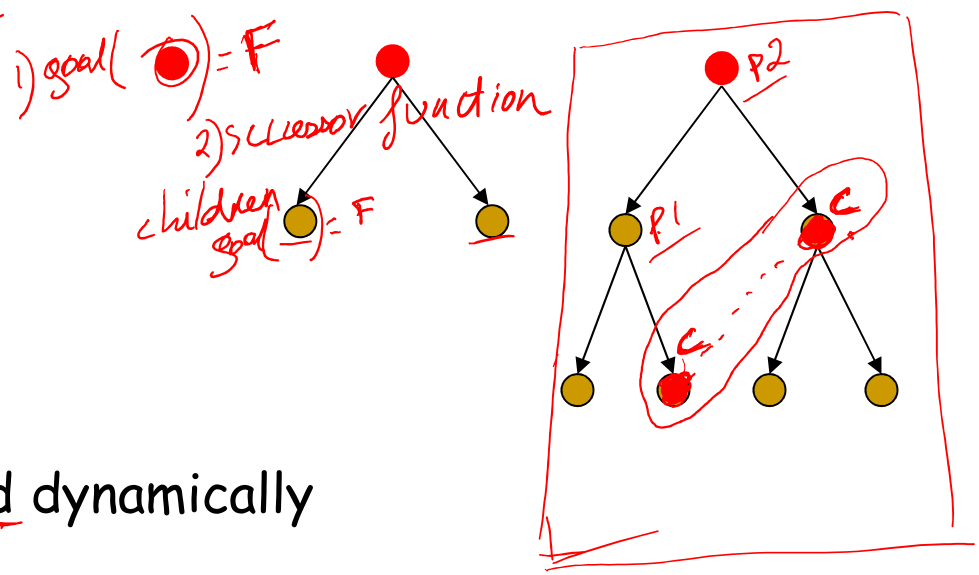
goal(•) → T
→ F

successor

s(•)

State Space Graph as a Search Tree



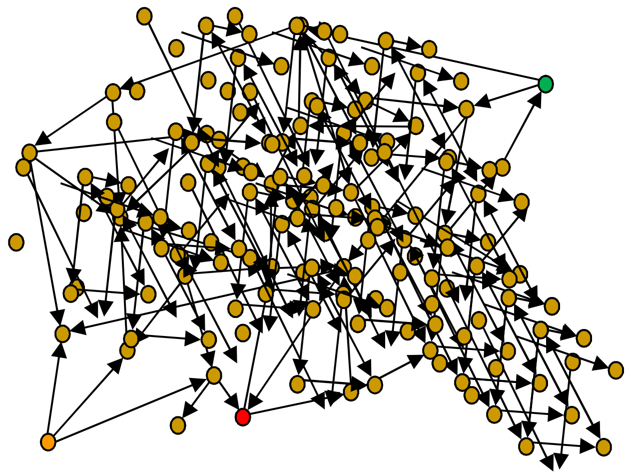


Search tree

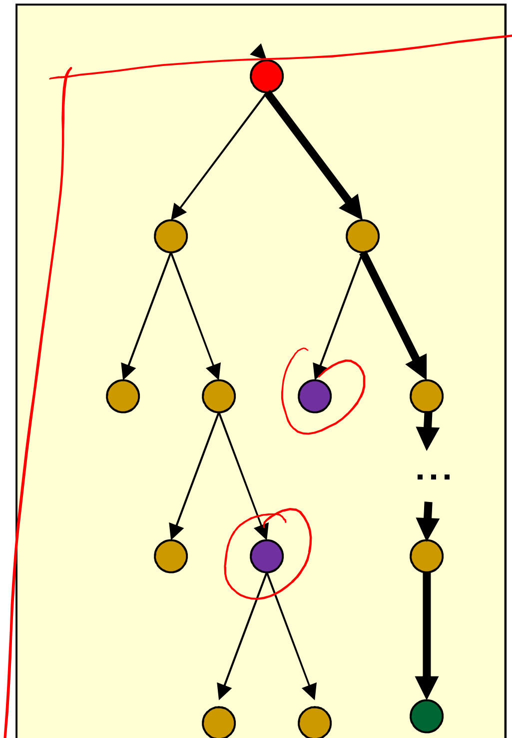
- The state space is generated dynamically while searching.
  - we explore a node:
    - if it the goal, stop and trace back the path from the initial node
    - if it is not the goal, then generating its successors/children and explore these recursively
  - to avoid cycles, the search algorithm will check for duplicate nodes.

solution path

State Space Graph as a Search Tree



Theoretical State Space Graph



Dynamically created
Search tree

So now, we just need an efficient search algorithm

Today

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

Up Next

1. State Space Representation
2. State Space Search — video #2
   a) Overview
   b) Uninformed search — COMP 352
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search — novelty
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary