Artificial Intelligence

SCALAB
Grupo de Inteligencia Artificial

Universidad Carlos III de Madrid

2017-2018

Uninformed Search – Exercises

Recap: Search and problem representation

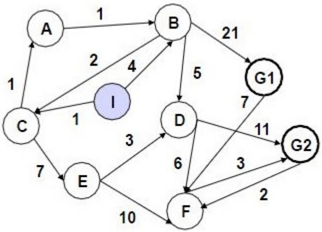
- The basic two elements to solve a problem are:
  - Its representation
  - The search of its solution
- Problem representation
  - Many problems of common practical interest have such a huge search space that they cannot be explicitly represented
  - Methods to implicitly represent these search spaces are needed. Also, we need efficient search methods for these spaces.
- State-space representation
  - Allows a formal problem definition: transform a given situation in a desired one using a set of allowed operators.
  - Allows a search solution: explore the state-space to find a path from inicial state to a goal state.

Recap: Search

- Search is a problem solution general mechanism
- Uninformed search
  - Thorough search of the search space until a solution is reached
  - Does not have knowledge to guide the search
  - Does not have knowledge regarding the domain
- Informed/Heuristic search
  - Explore promising paths
  - Knowledge included: hints to cut down the search process and make it more efficient

Example Breadth First Search I

State Space

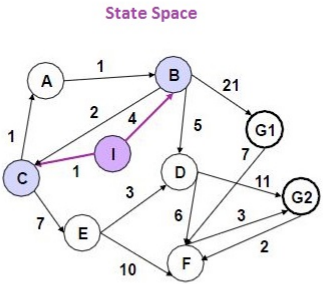


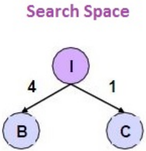
Complete and optimal

Search Space

Open nodes list : I

Example Breadth First Search II

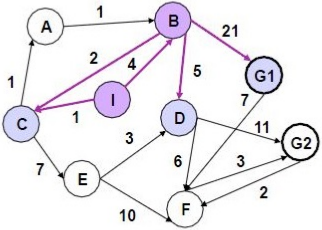




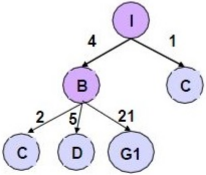
Open nodes list : C B

Example Breadth First Search III

State Space



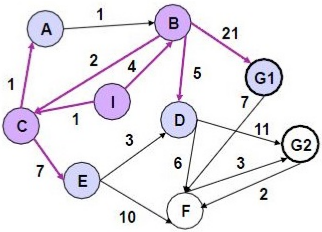
Search Space



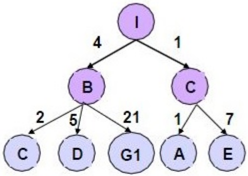
Open nodes list : G1 D C C

Example Breadth First Search IV

State Space

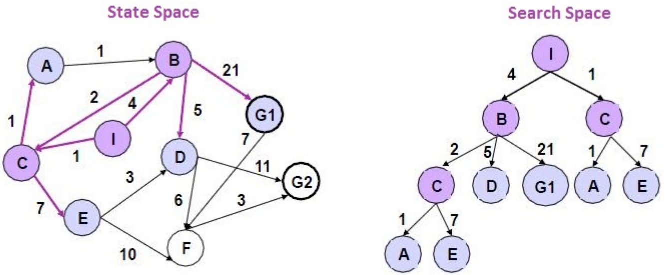


Search Space



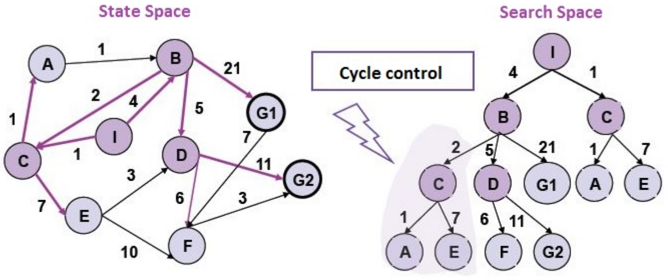
Open nodes list : E A G1 D C

Example Breadth First Search V



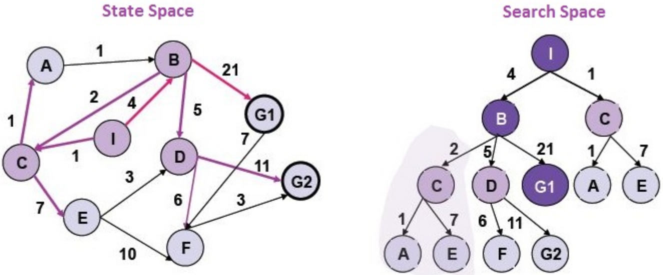
Open nodes list : E A E A G1 D

Example Breadth First Search VI



Open nodes list : G2 F E A E A G1

Example Breadth First Search VII

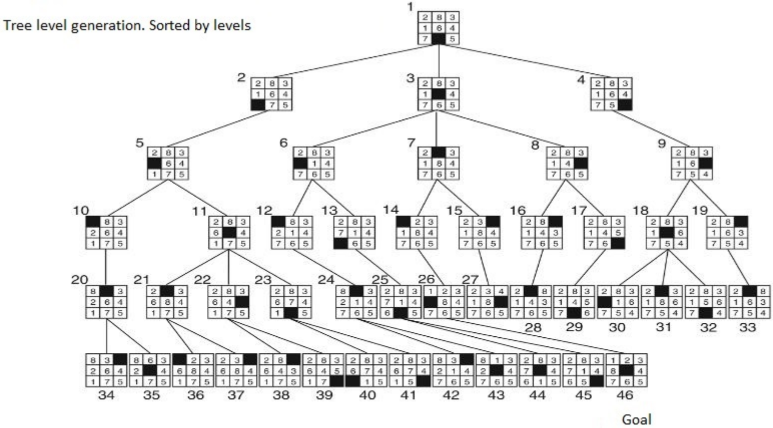


Expanded nodes (in order)	: I B C C D G1

Generated nodes (in order)	: I B C C D G1 A E A E F G2

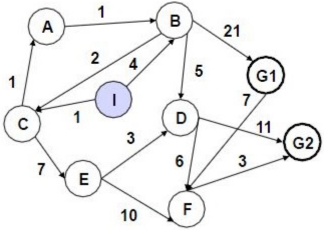
Solution path	: I B G1	Cost	: 4+21 = 25

Example Breadth First Search VIII



Example Dijkstra Search I

State Space

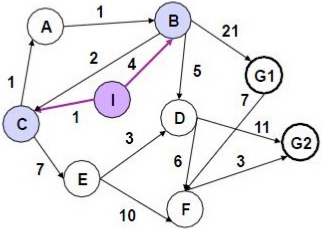


Search Space

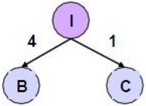
Priority list of open nodes : I(0)

Example Dijkstra Search II

State Space



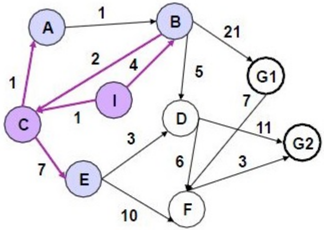
Search Space



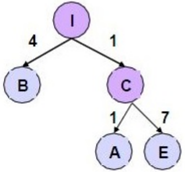
Priority list of open nodes : B(4) C(1)

Example Dijkstra Search III

State Space

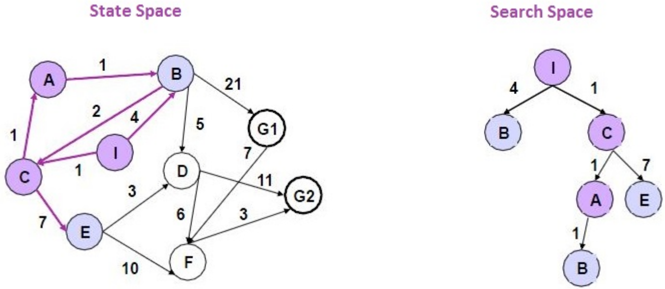


Search Space



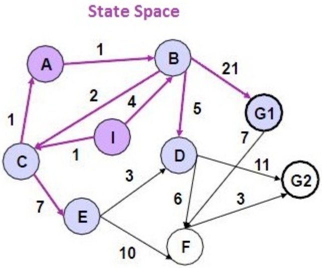
Priority list of open nodes	: E(8) B(4) A(2)

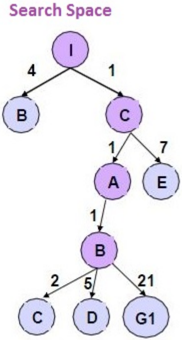
Example Dijkstra Search IV



Priority list of open nodes : E(8) B(4) B(3)

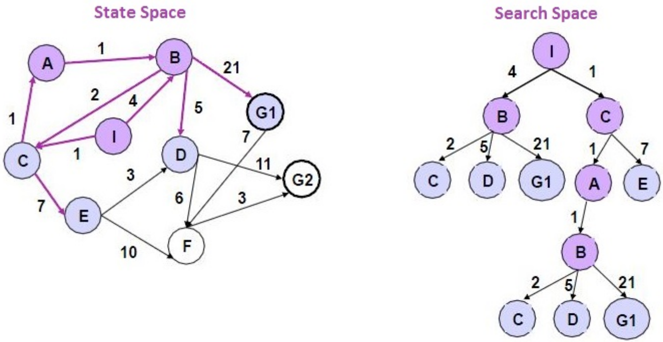
Example Dijkstra Search V





Priority list of open nodes : G1(24) E(8) D(8) C(5) B(4)

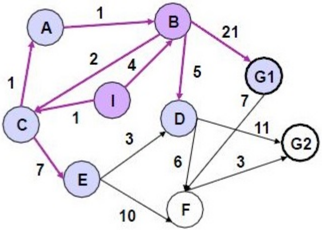
Example Dijkstra Search VI



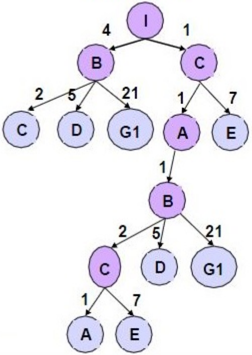
Priority list of open nodes	: G1(25) G1(24) D(9) E(8) D(8) C(6) C(5)

Example Dijkstra Search VII

State Space



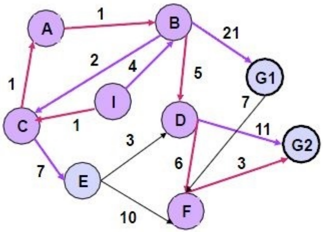
Search Space



Same dynamic a few steps more

Priority list of open nodes	: G1(25) G1(24) E(12) D(9) E(8) D(8) C(6) A(6)

Example Dijkstra Search VIII



Expands a lot of nodes

Note cycle problem ->
Use cycle control

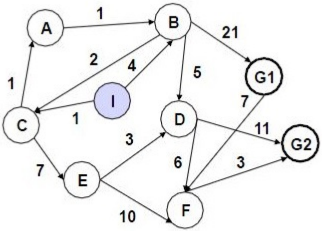
However, with cycle control maybe we do not reach optimal solution.
In this example B (son of A) would not be generated...

Complete and optimal

Solution path : I C A B D F G2	**Cost** : 1+1+1+5+6+3 = 17

Example Depth First Search I

State Space

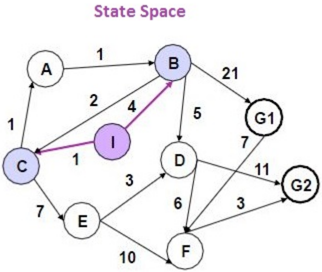


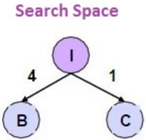
Search Space

Not complete
Not optimal

Open nodes list : I

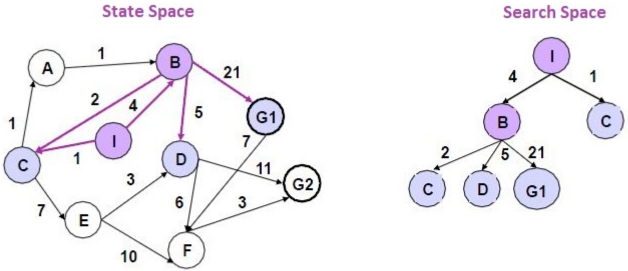
Example Depth First Search II





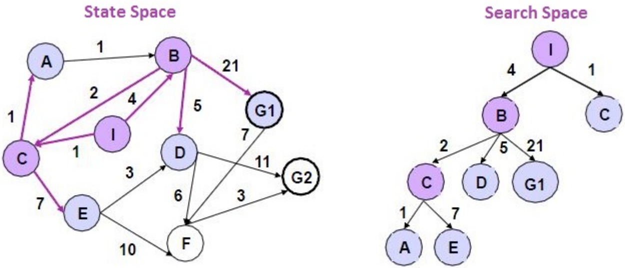
Open nodes list : B C

Example Depth First Search III



Open nodes list : C D G1 C

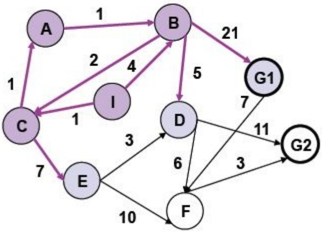
Example Depth First Search IV



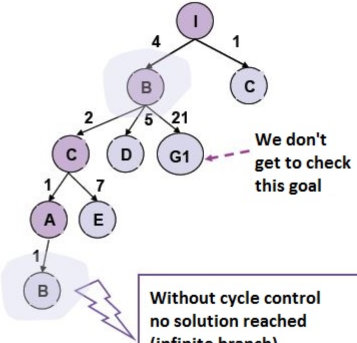
Open nodes list	: A E D G1 C

Example Depth First Search V

State Space



Search Space

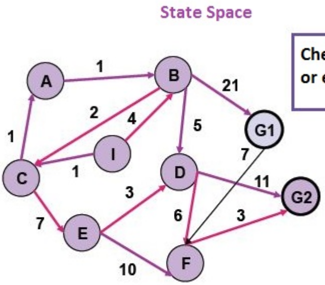


Open nodes list : B E D G1 C

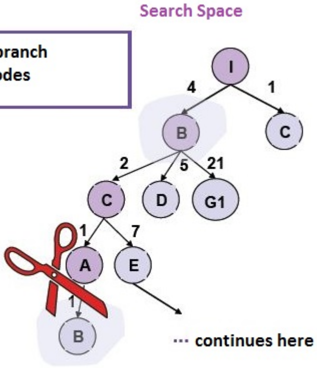
Closed expanded nodes : I B C A

Without cycle control no solution reached (infinite branch)

Example Depth First + cycle control Search I



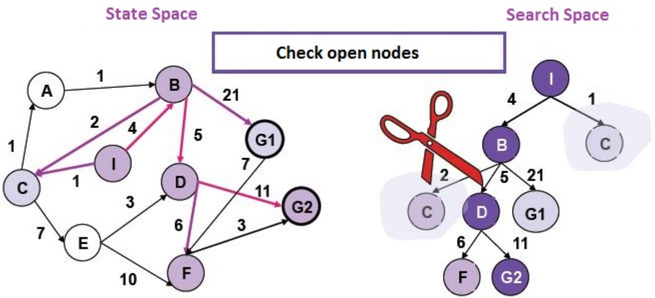
Check current branch
or expanded nodes



Expanded nodes : I B C A E D F G2

Solution path : I B C E D F G2 **Cost** : 4+2+7+3+6+3 = 25

Example Depth First + cycle control Search II

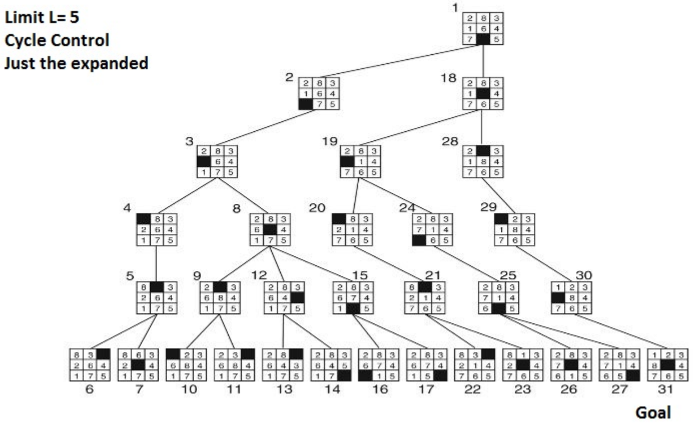


Expanded nodes (in order) : I B D F G2

Open nodes (in order) : I B C D G1 F G2

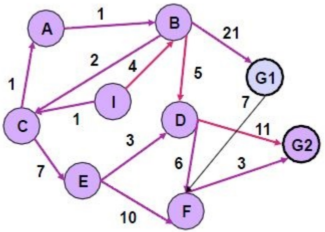
Solution path : I B D G2 **Cost** : 4+5+11 = 20

Example Depth First Search + depth limit

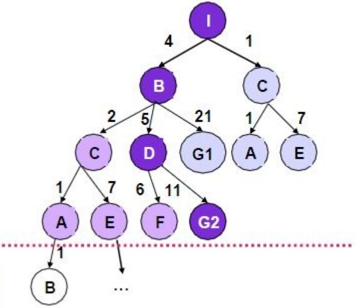


Example Depth First Search + depth limit=3

State Space



Search Space

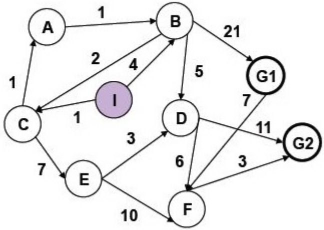


Expanded nodes : I B C A E D F G2

Solution path : I B D G2 **Cost** : 4+5+11 = 20

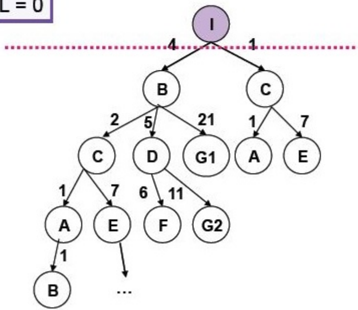
Example Iterative Search I

State Space



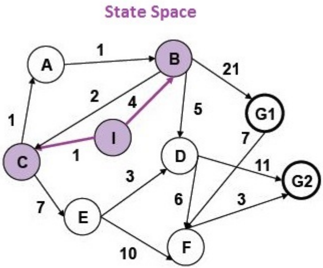
Search Space

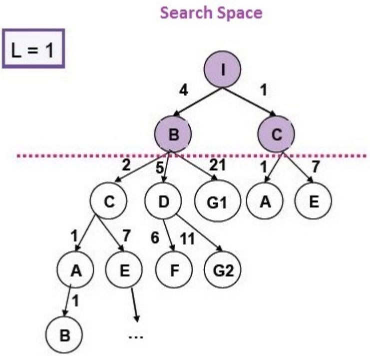
L = 0



Expanded nodes : I

Example Iterative Search II



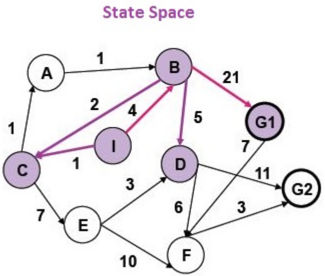


Optimal and Complete

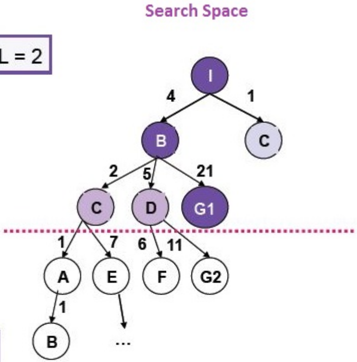
Expanded nodes : I I B C

Combination of BFS and DFS = 1 step of breadth 1 step of depth

Example Iterative Search III



L = 2

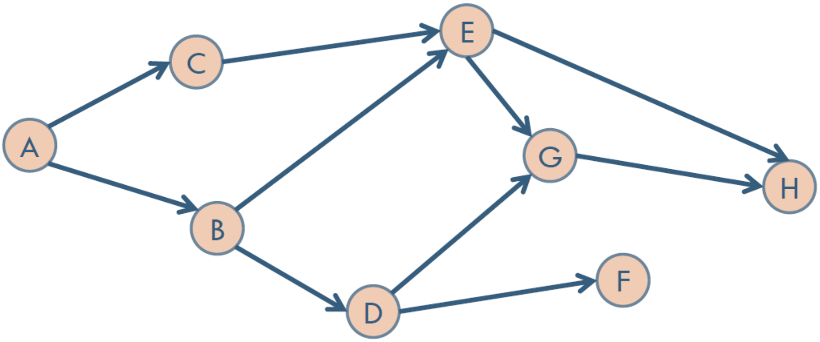


Expanded nodes : I I B C I B C D G1

Solution path : I B G1 **Cost** : 4+21 = 25

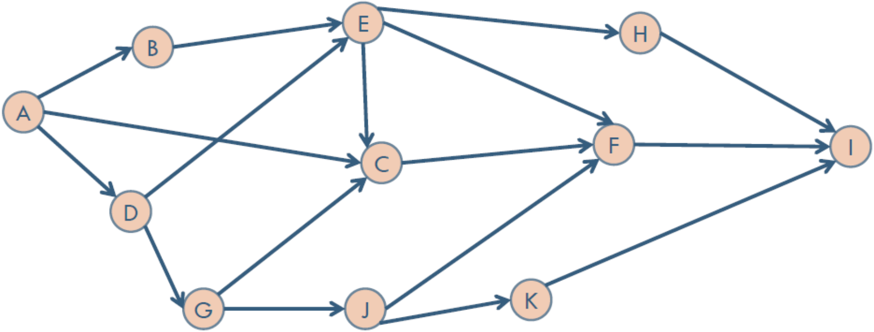
Exercise 1: Breadth First Search and Depth First Search

Apply BFS and DFS to the following graph. Indicate all the details (open list, generated nodes, expanded nodes) for each step. For child nodes pick an alphabetical order for expansion. Initial State=A, Goal state=H.



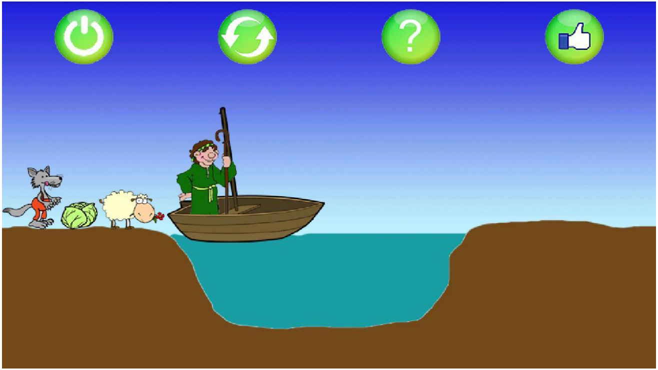
Exercise 2: Shortest path

Find the shortest path (less edges) between A and I



Exercise 3: the wolf-sheep-cabbage problem

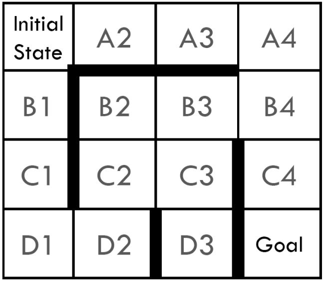
- Problem:
  A man owns a wolf, a sheep and a cabbage. He is on a river bank with a boat that can carry him with only one of his goodies at a time. The man wants to reach the other bank with his wolf, sheep and cabbage, but he knows that wolves eat sheep, and sheep eat cabbages, so he cannot leave them alone on a bank.



Exercise 3: the wolf-sheep-cabbage problem

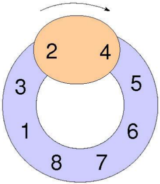
- Questions:
  1. How to represent this problem as a search problem?
     - How do you represent each state? What are the initial and goal states?
     - What are the operators?
  2. What states can be generated from the initial state?
  3. Continue the search up to 5 moves (homework).

Exercise 4: maze



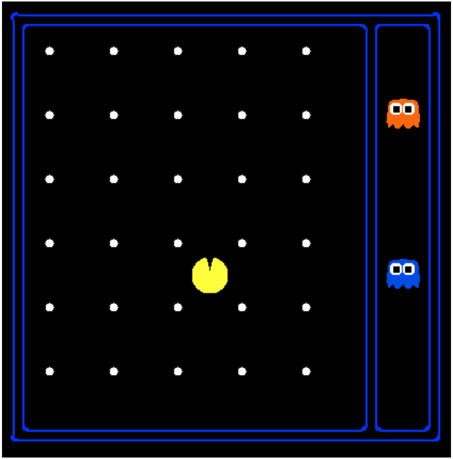
- The aim is to find a path to the goal
- Operators: Move up, move left, move down, move right
- Assuming the order given for the operators, use breadth first search to solve the maze problem. Mark every cell with the number of expanded node
- Apply depth first search for the same problem
- Change the order of operators to find a faster solution

Exercise 5: top spin



- Permutation game in which we rotate the base to exchange the order of numbers
- Goal state: numbers are ordered from 1 to 8 with the pair (1 2) in the base
- Operators: move numbers to the right, move numbers to the left, rotate base to switch order of the numbers in it
- Apply DFS to solve the top-spin problem, use a maximun depth of d=3
- Apply breadth first for the same problem
- Apply DFS to solve the top-spin problem, use a maximun depth of d=4. How many nodes are expanded in this case?

Exercise 0: State Space



Exercise 0: State Space

1. Which are the possible states in the previous Pacman domain? how many there are?
2. Suppose that in this domain we we define a problem where Pacman starts in an initial position and must reach a final position
   - Is this a search problem?
   - What is the state space? What size it has?
   - Which state is the initial? and the final/s?
   - Which actions do we have? and how are they?
3. Suppose that now we define a problem where Pacman has to eat all the food balls
   - Is this a search problem?
   - What is the state space? What size it has?
   - Which state is the initial? and the final/s?
   - Which actions do we have? and how are they?