Artificial Intelligence

SCALAB

Grupo de Inteligencia Artificial

Universidad Carlos III de Madrid

2017-2018

Uninformed Search – Exercises

Recap: Search and problem representation

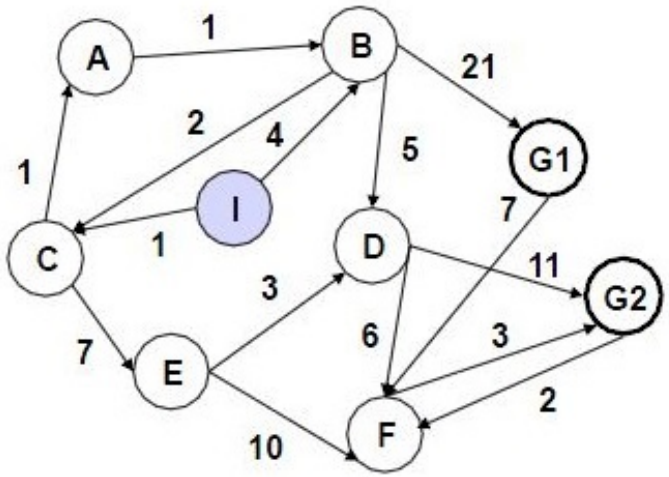
- The basic two elements to solve a problem are:
  - Its representation
  - The search of its solution
- Problem representation
  - Many problems of common practical interest have such a huge search space that they cannot be explicitly represented
  - Methods to implicitly represent these search spaces 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



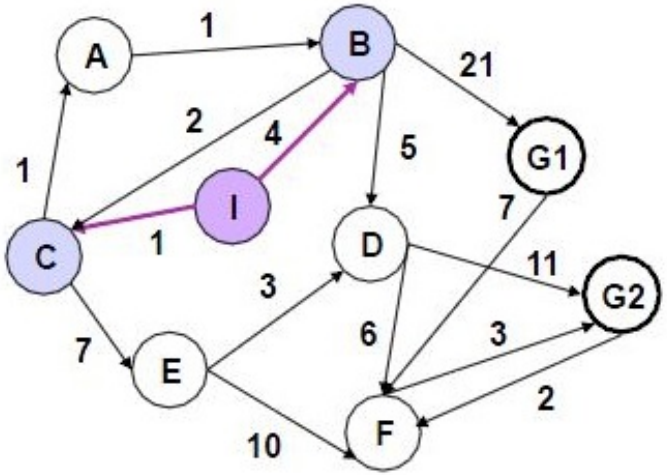
Search Space

Complete and optimal

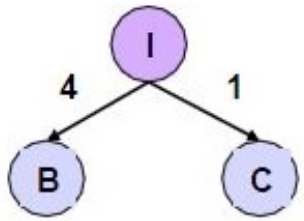
Open nodes list : I

Example Breadth First Search II

State Space



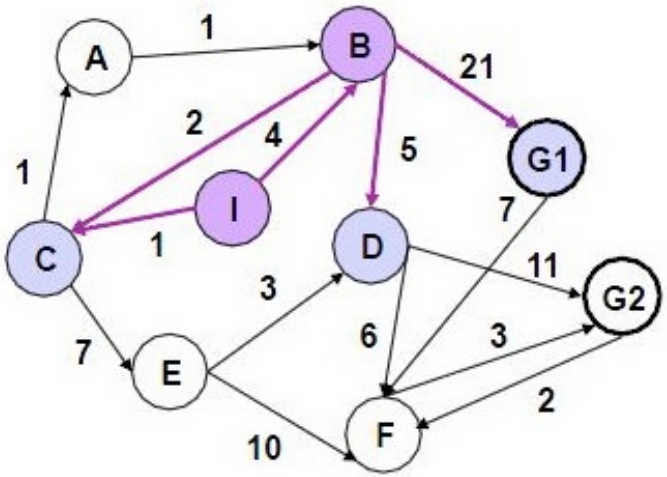
Search Space



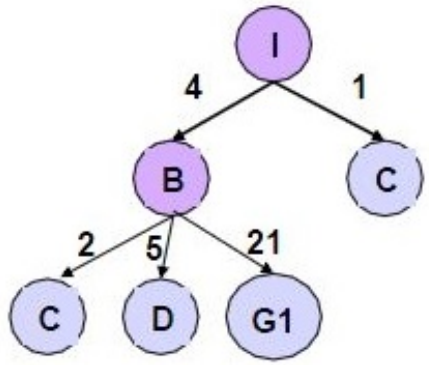
Open nodes list : C B

Example Breadth First Search III

State Space



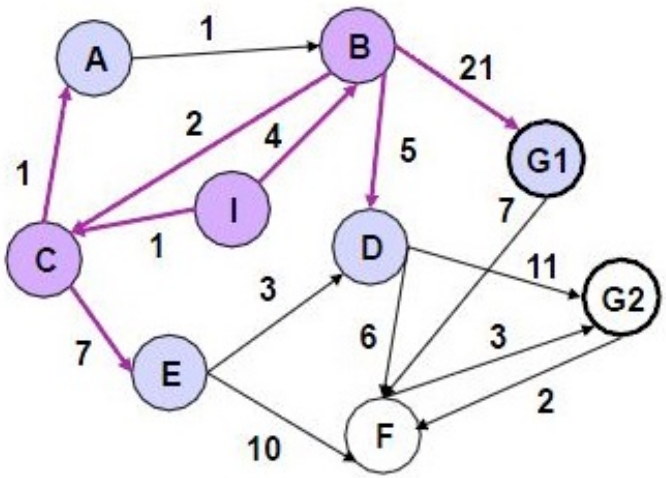
Search Space



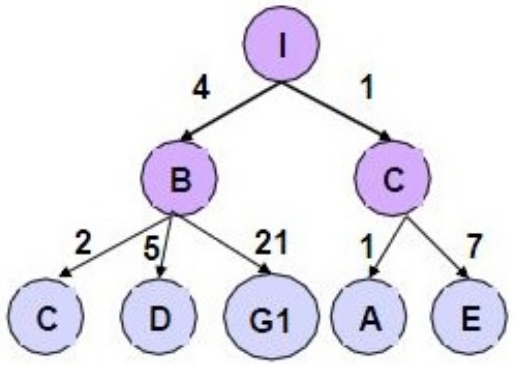
Open nodes list : G1 D C C

Example Breadth First Search IV

State Space

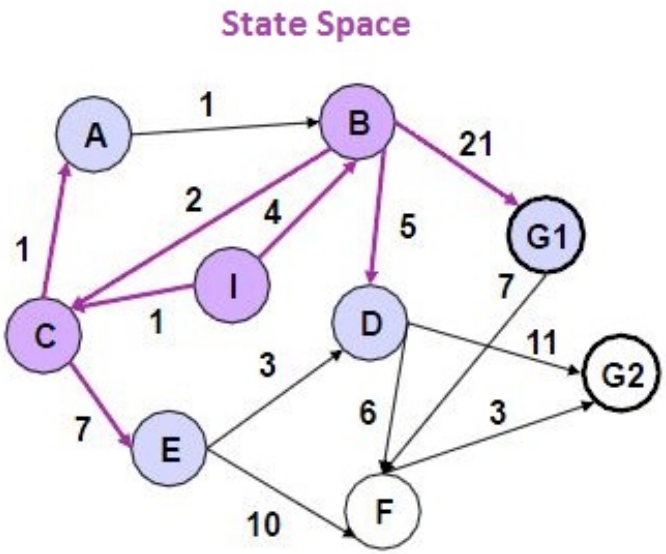


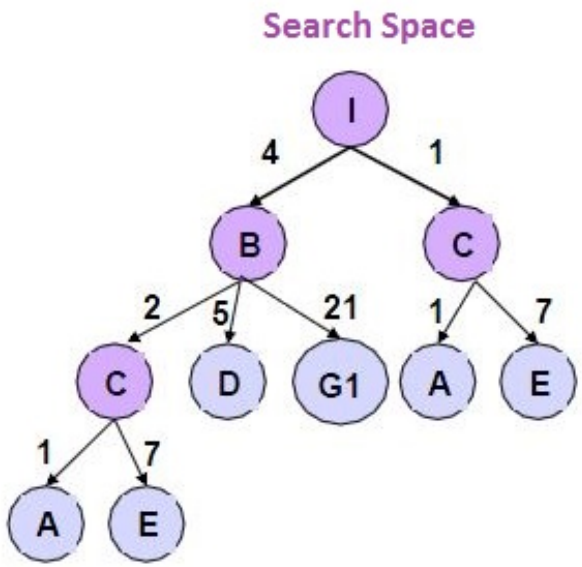
Search Space



Open nodes list : E A G1 D C

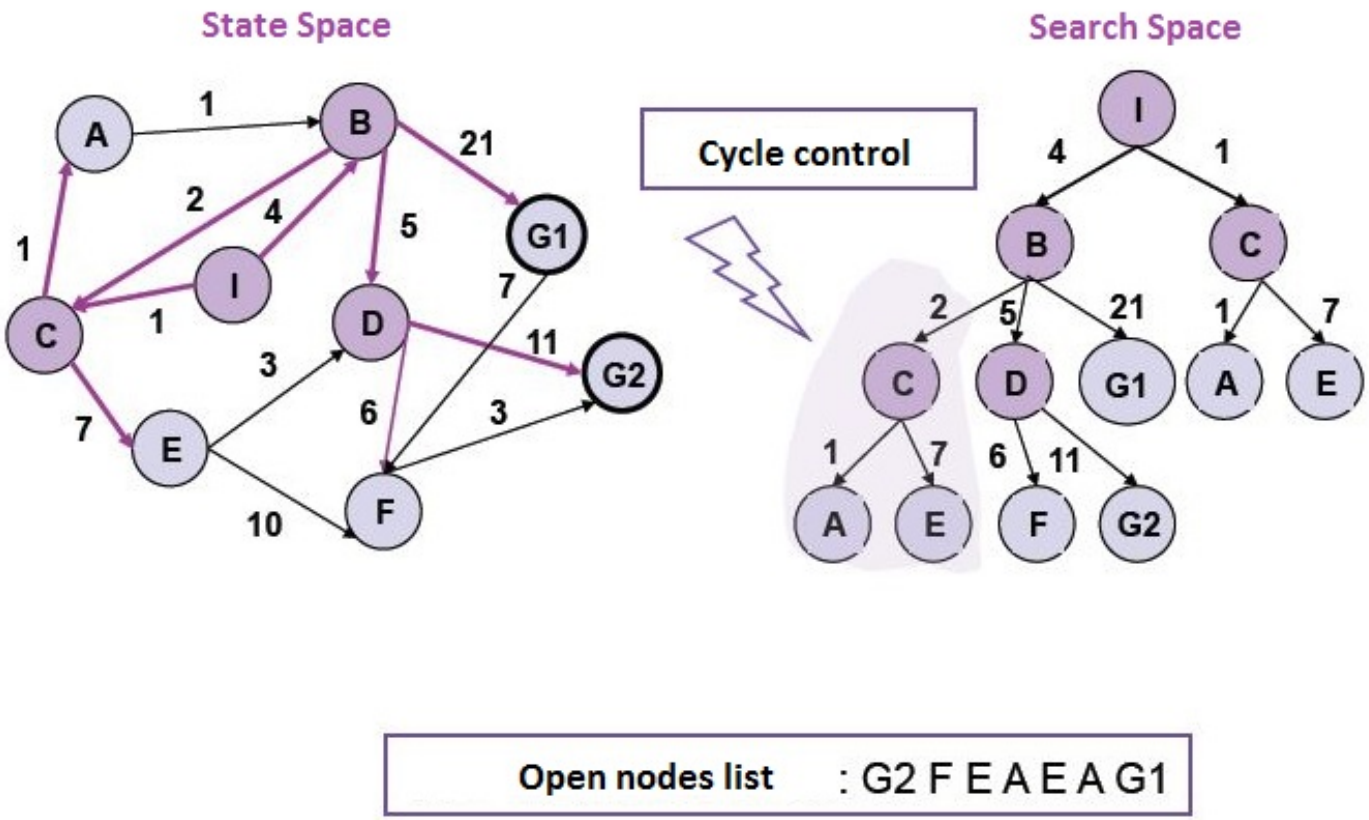
Example Breadth First Search V



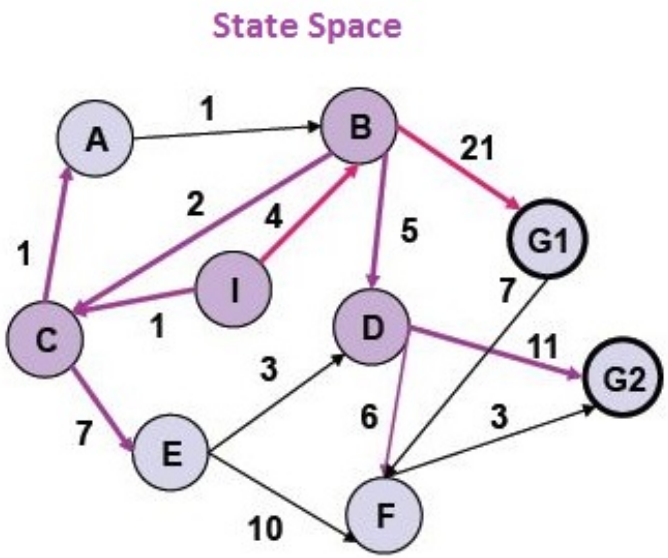


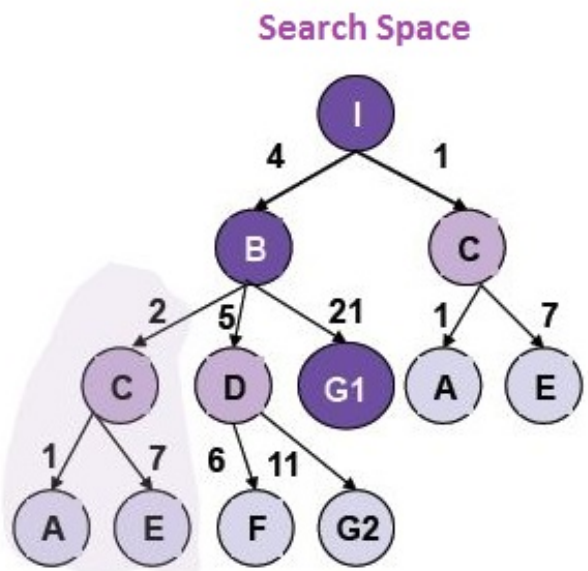
Open nodes list : E A E A G1 D

Example Breadth First Search VI



Example Breadth First Search VII



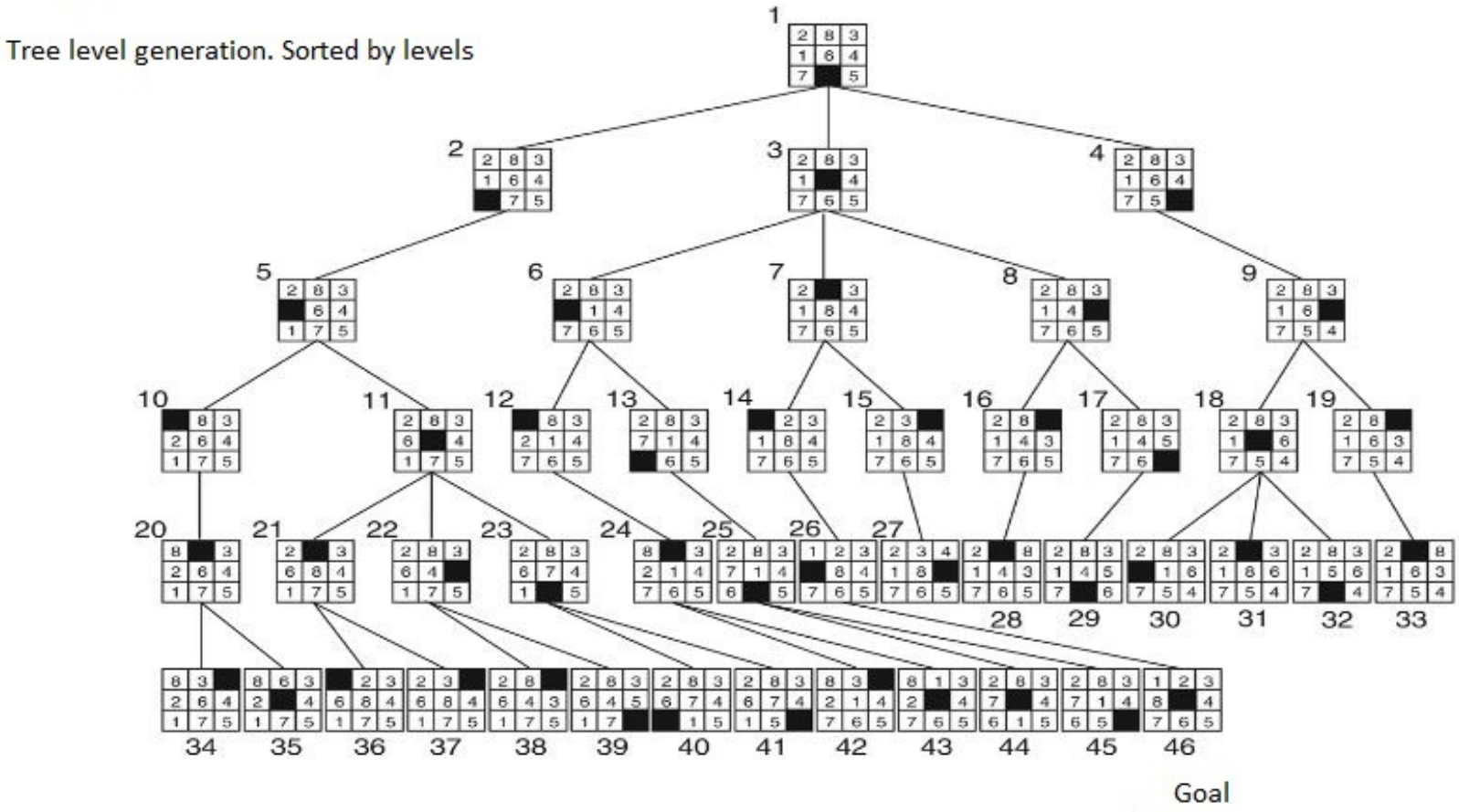


Expanded nodes (in order) : I B C C D G1

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

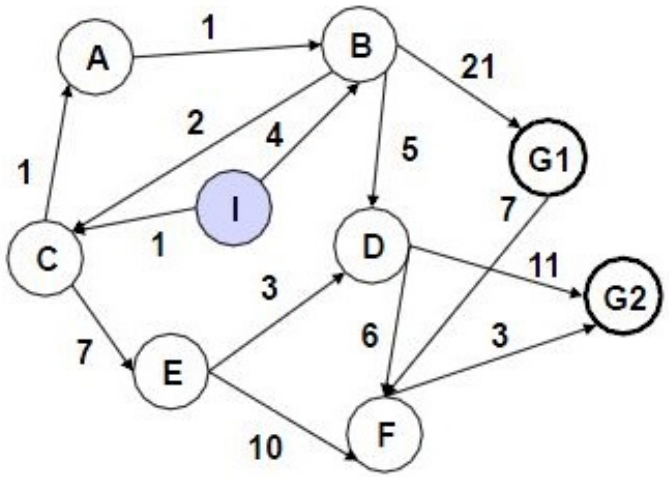
Solution path : I B G1 **Cost** : 4+21 = 25

Example Breadth First Search VIII



Example Dijkstra Search I

State Space

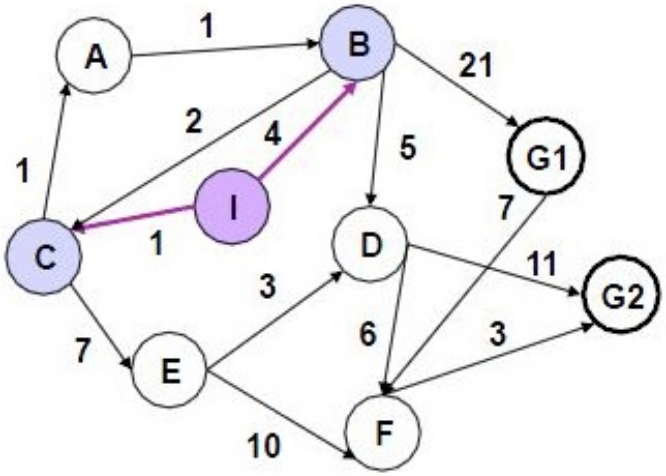


Search Space

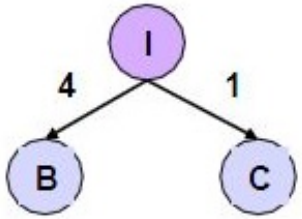
Priority list of open nodes	: I(0)

Example Dijkstra Search II

State Space



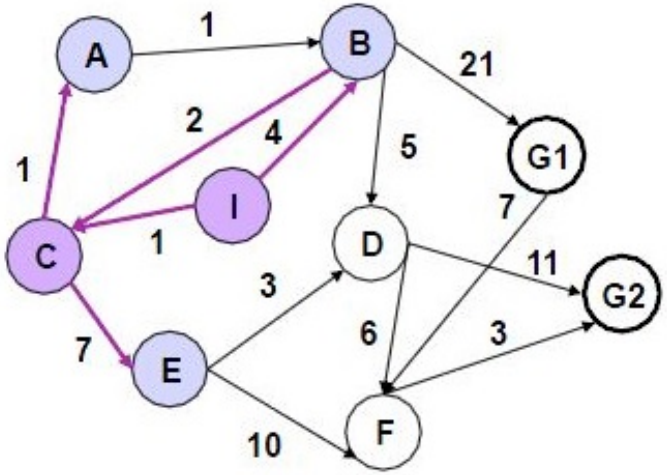
Search Space



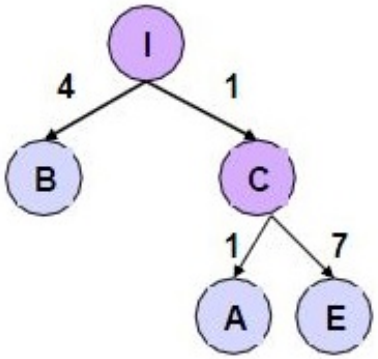
Priority list of open nodes : B(4) C(1)

Example Dijkstra Search III

State Space



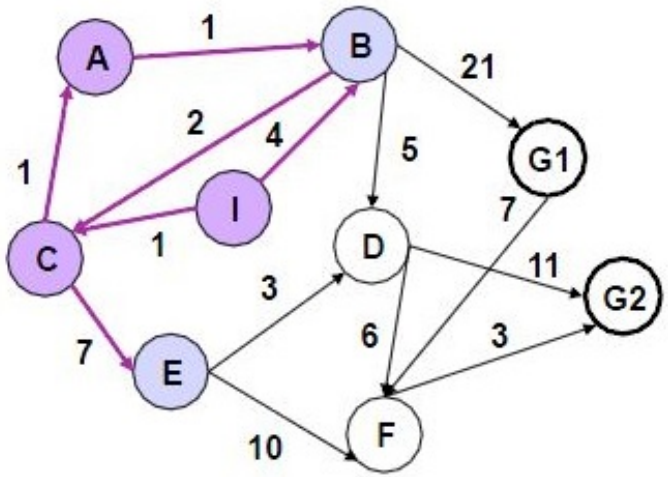
Search Space



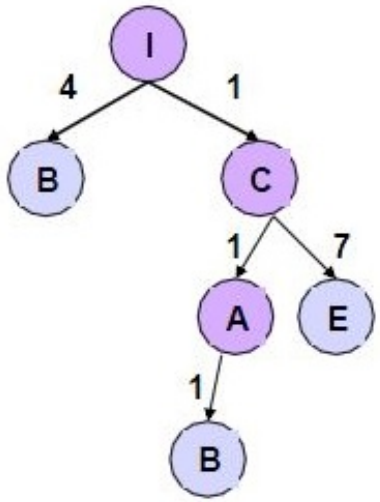
Priority list of open nodes	: E(8) B(4) A(2)

Example Dijkstra Search IV

State Space

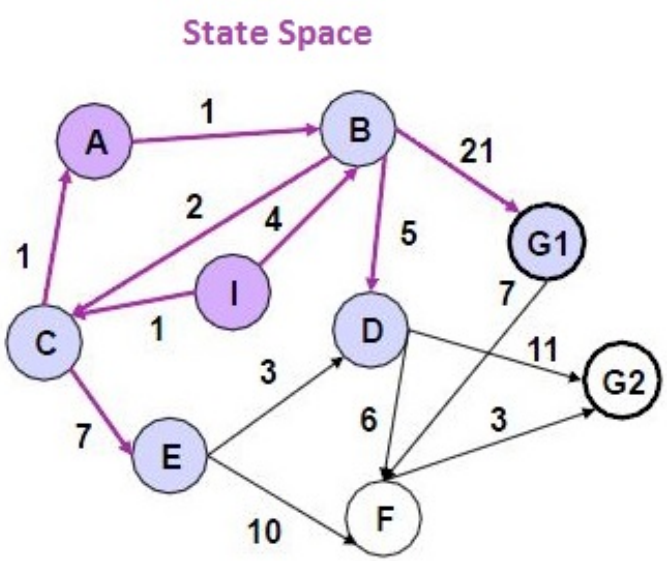


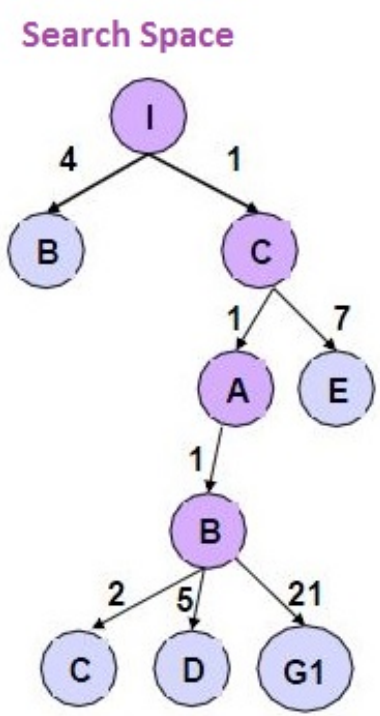
Search Space



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

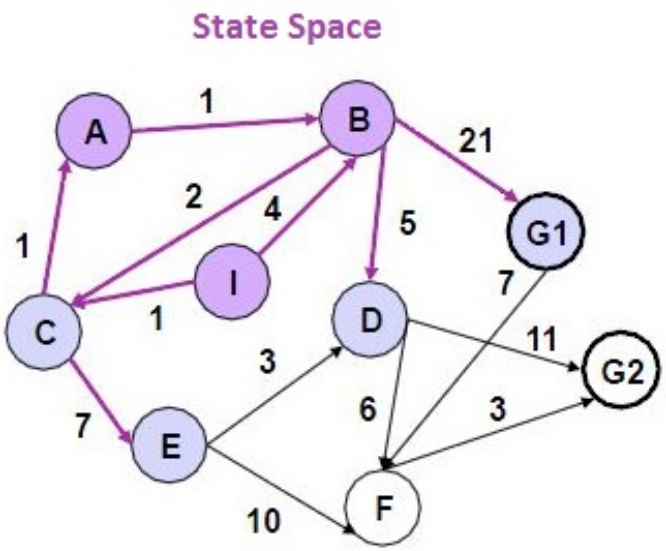
Example Dijkstra Search V

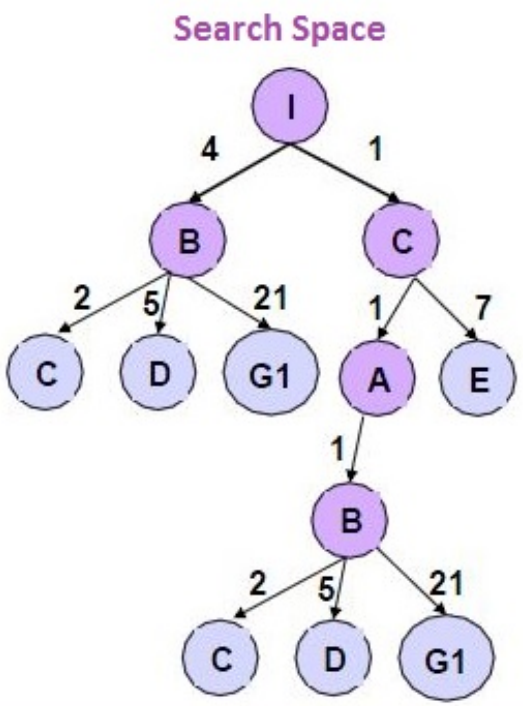




Priority list of open nodes : G1(24) E(8) D(8) C(5) B(4)

Example Dijkstra Search VI

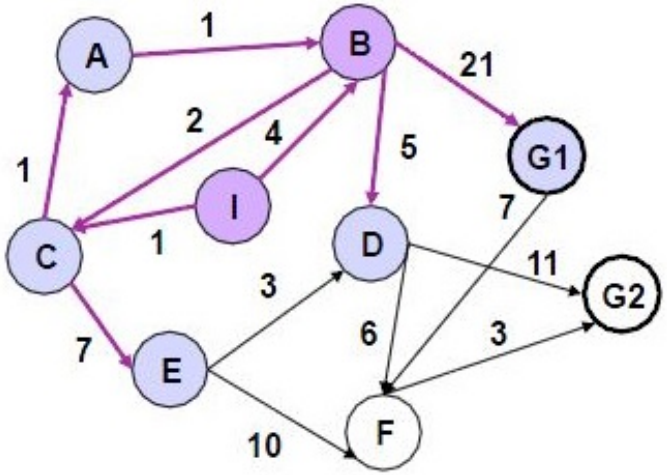




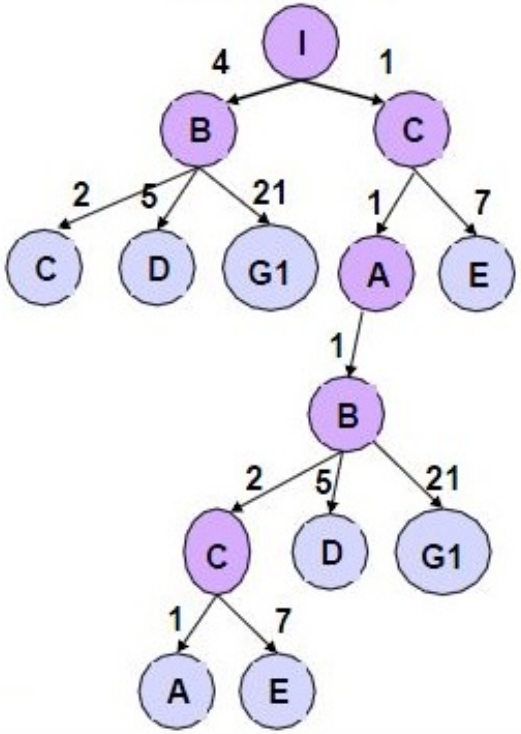
Priority list of open nodes : G1(25) G1(24) D(9) E(8) D(8) C(6) C(5)

Example Dijkstra Search VII

State Space



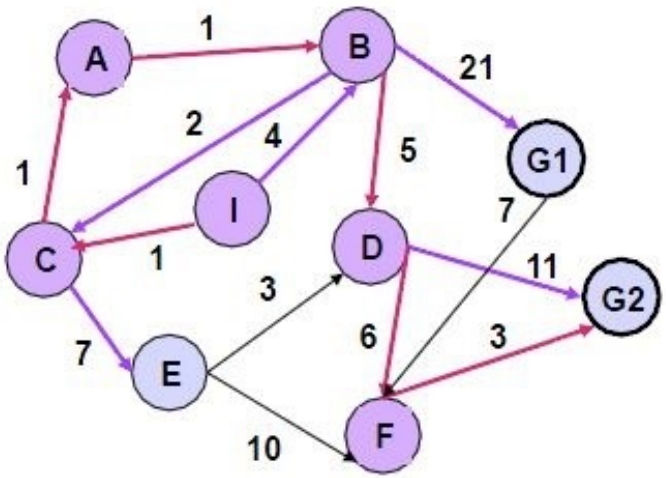
Search Space



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

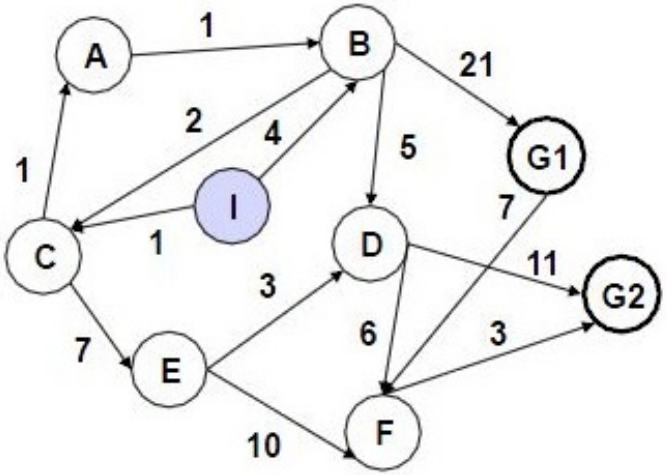
However, with cycle control maybe we do not reach optimal solution. In this example B (son of A) would not be generated...

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



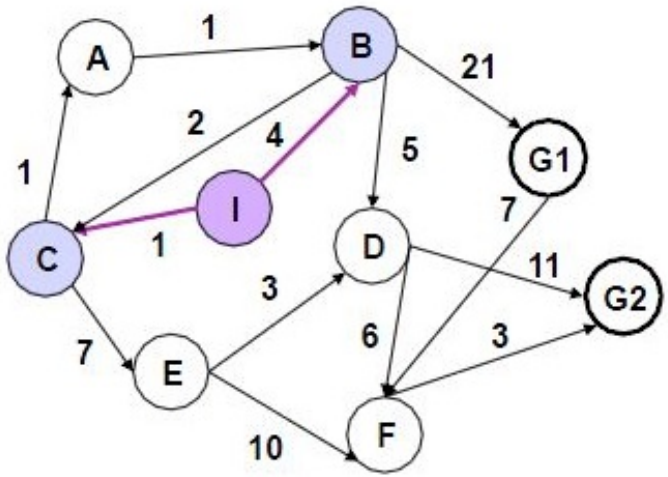
Search Space

Not complete
Not optimal

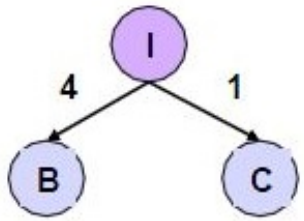
Open nodes list : I

Example Depth First Search II

State Space

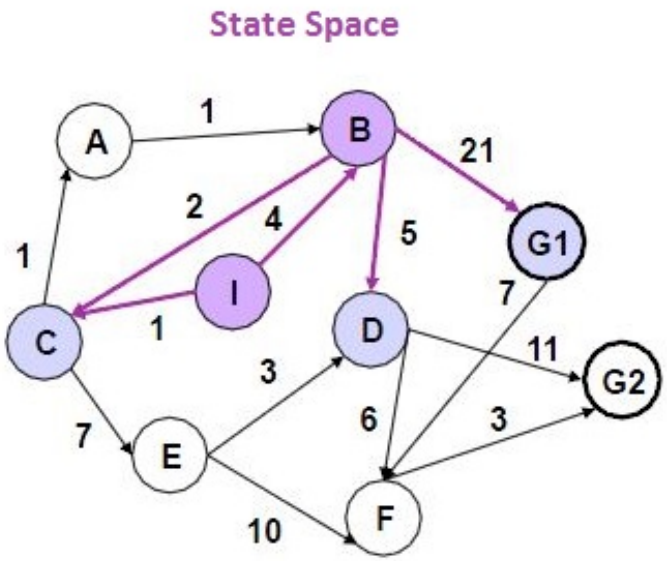


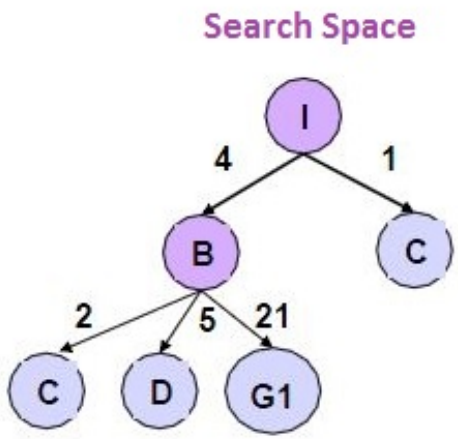
Search Space



Open nodes list	: B C

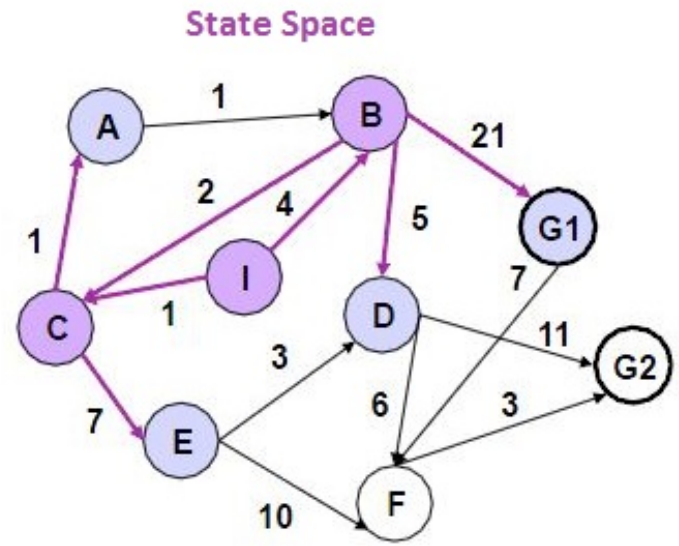
Example Depth First Search III

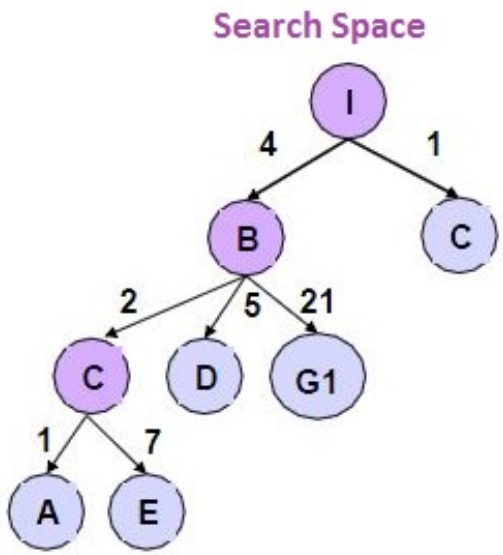




Open nodes list : C D G1 C

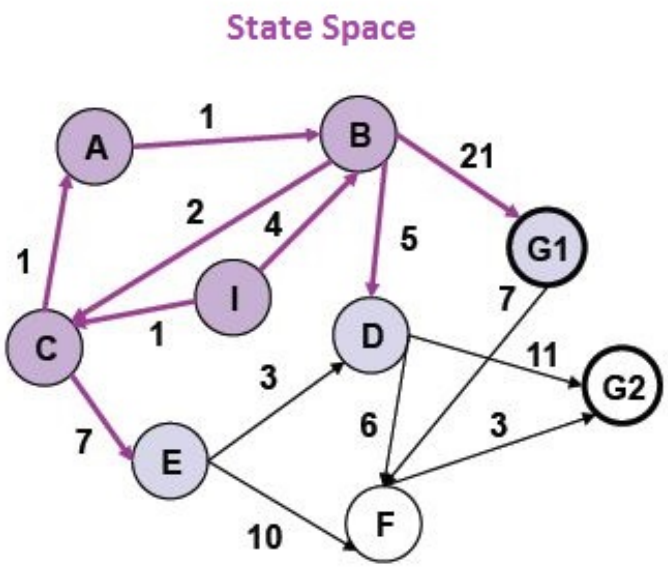
Example Depth First Search IV

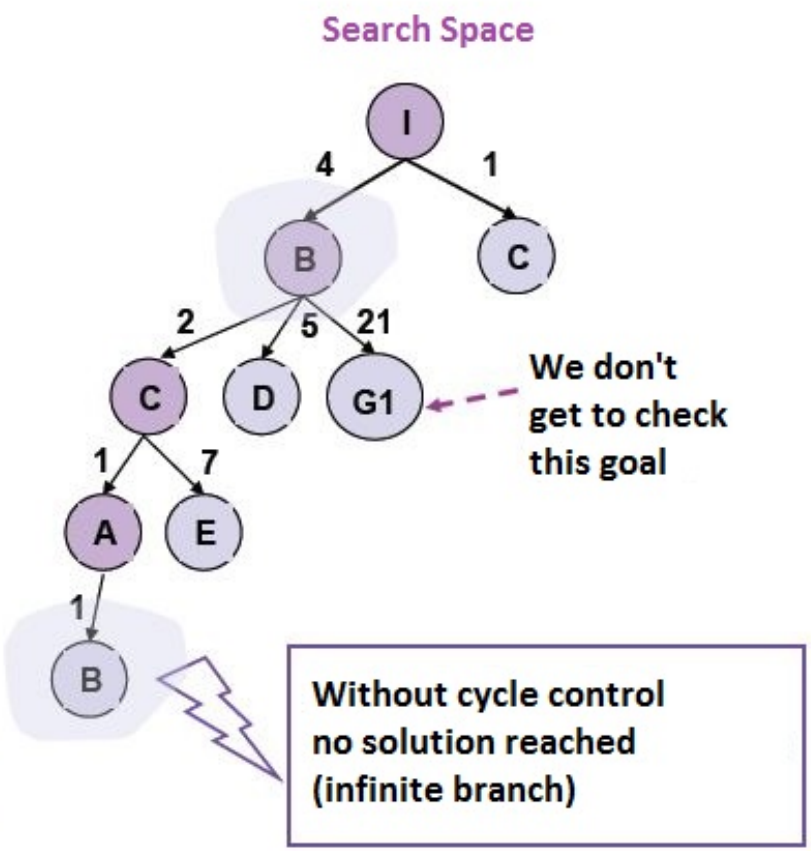




Open nodes list : A E D G1 C

Example Depth First Search V

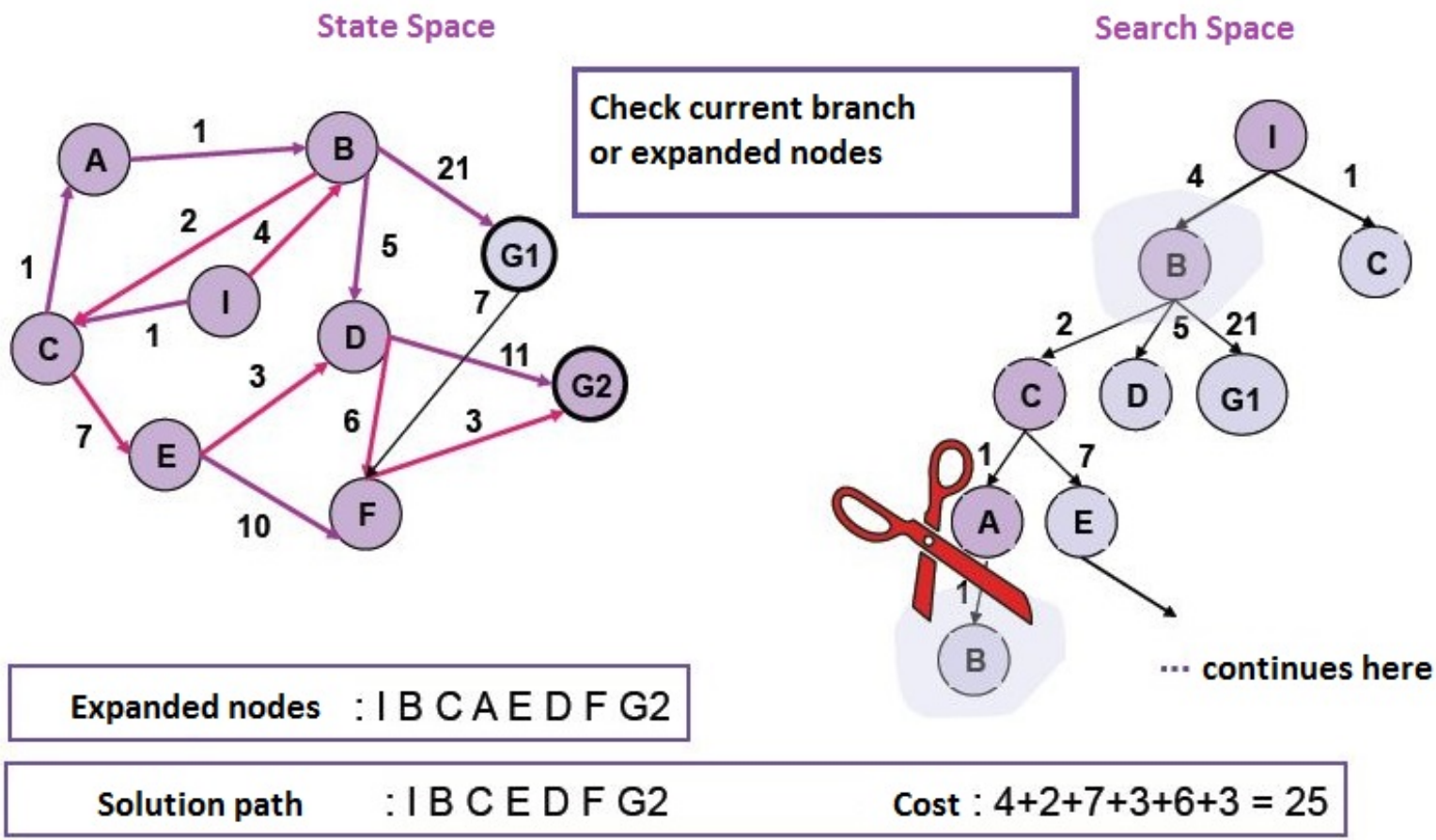




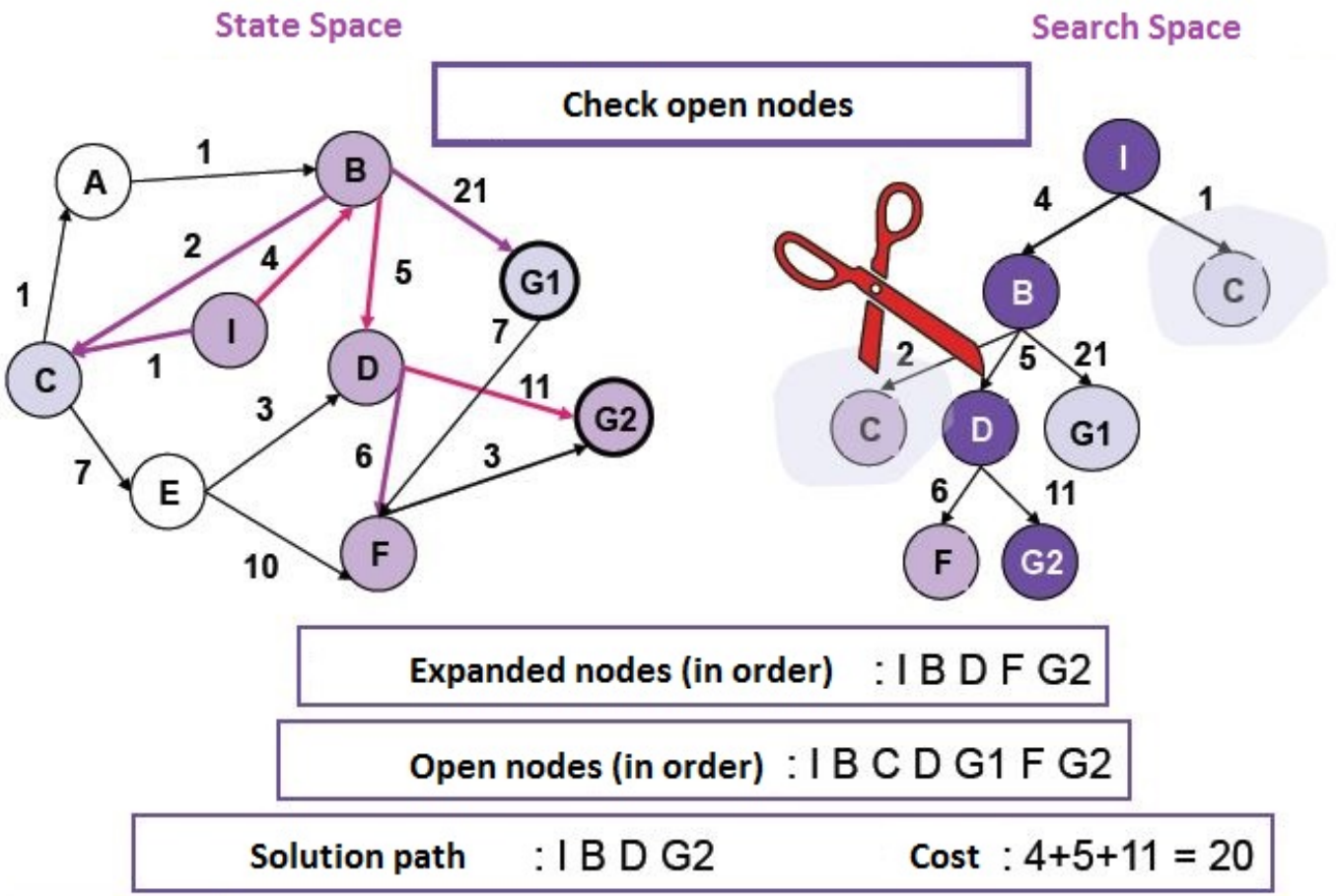
Open nodes list : B E D G1 C

Closed expanded nodes : I B C A

Example Depth First + cycle control Search I

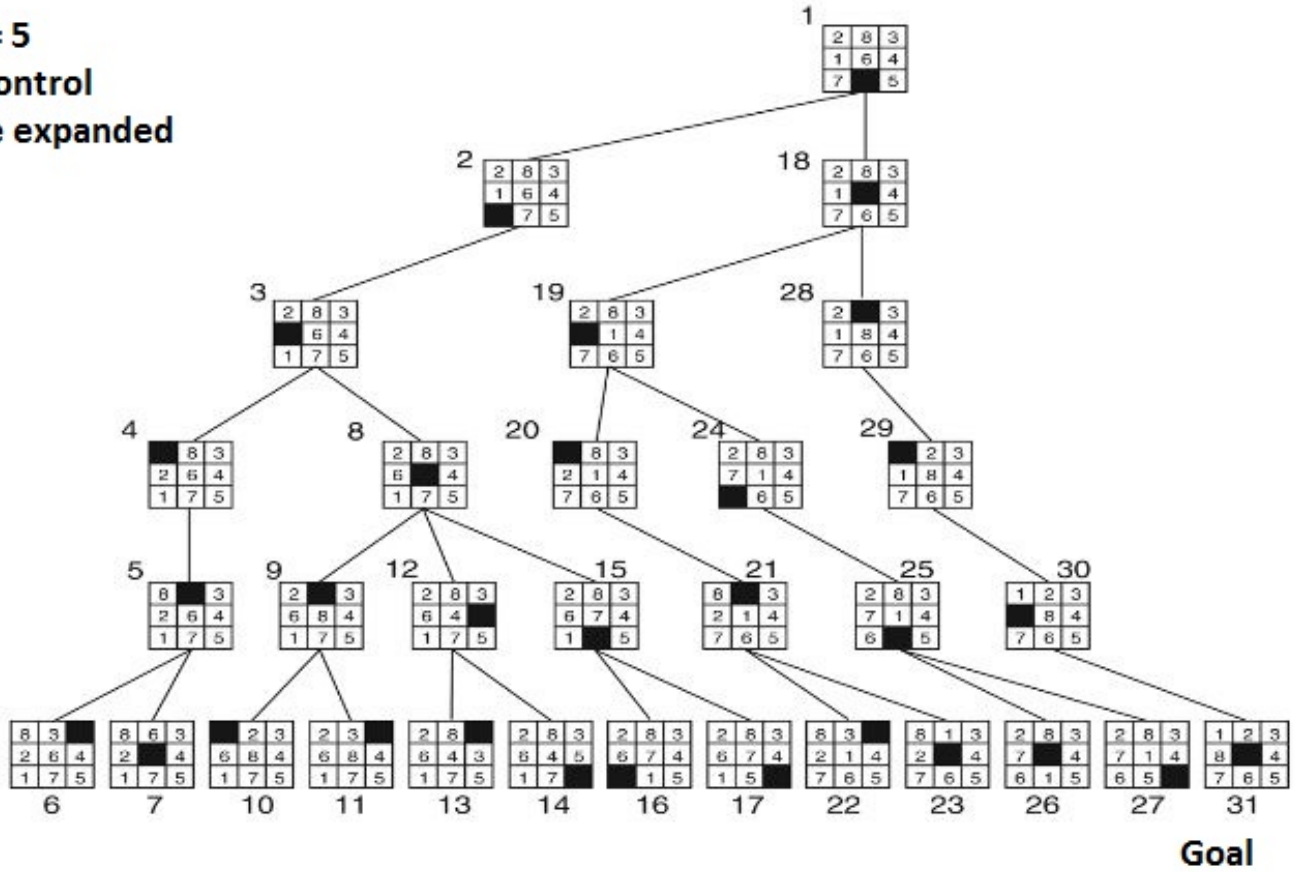


Example Depth First + cycle control Search II

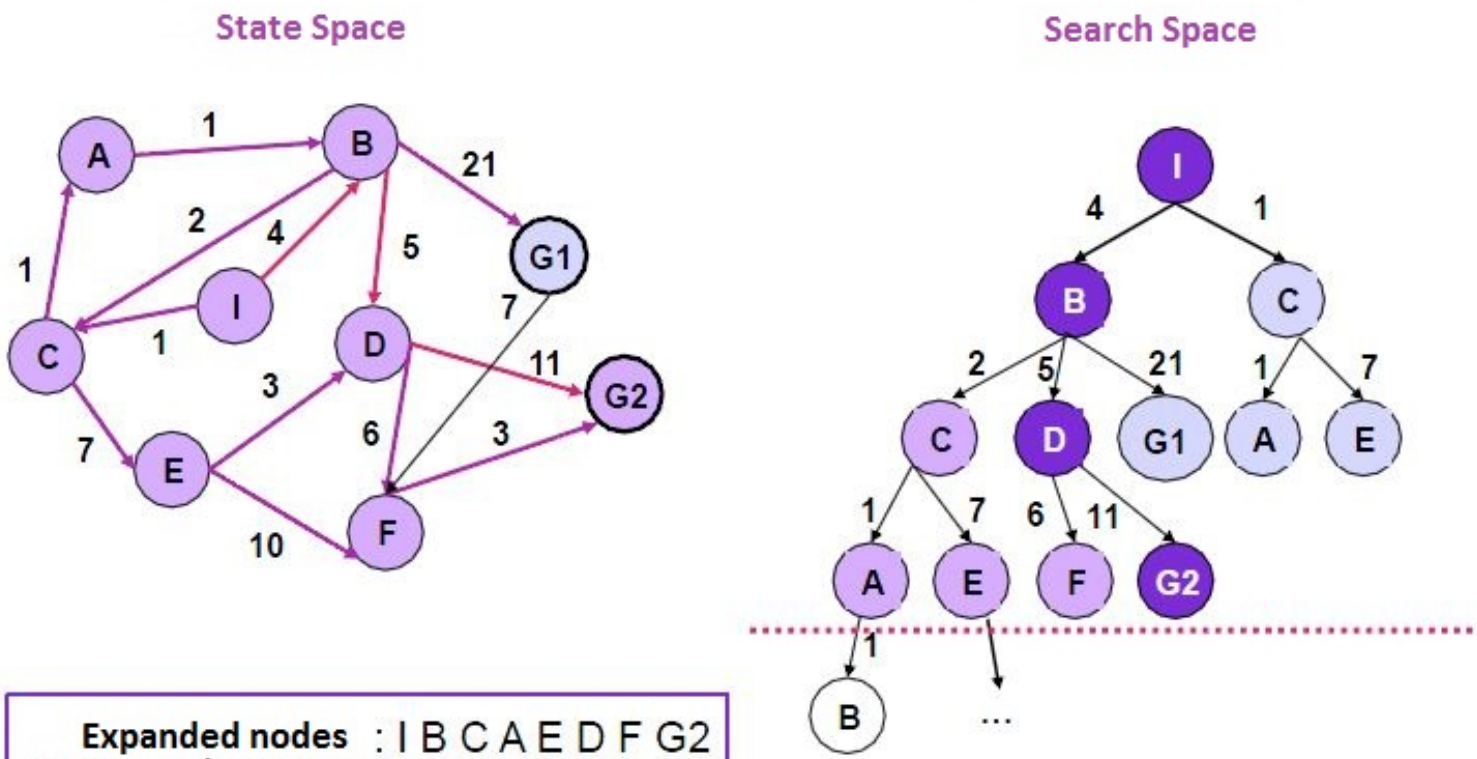


Example Depth First Search + depth limit

Limit L= 5
Cycle Control
Just the expanded



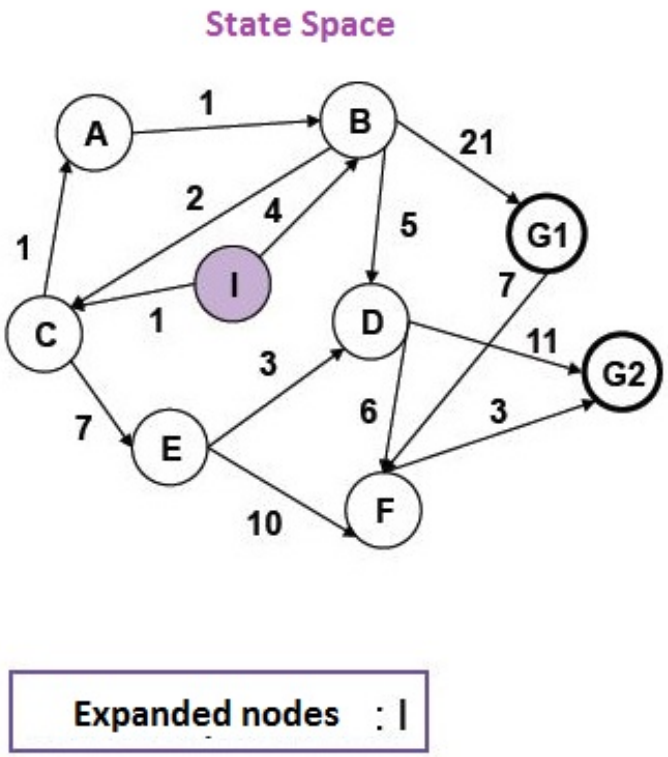
Example Depth First Search + depth limit=3

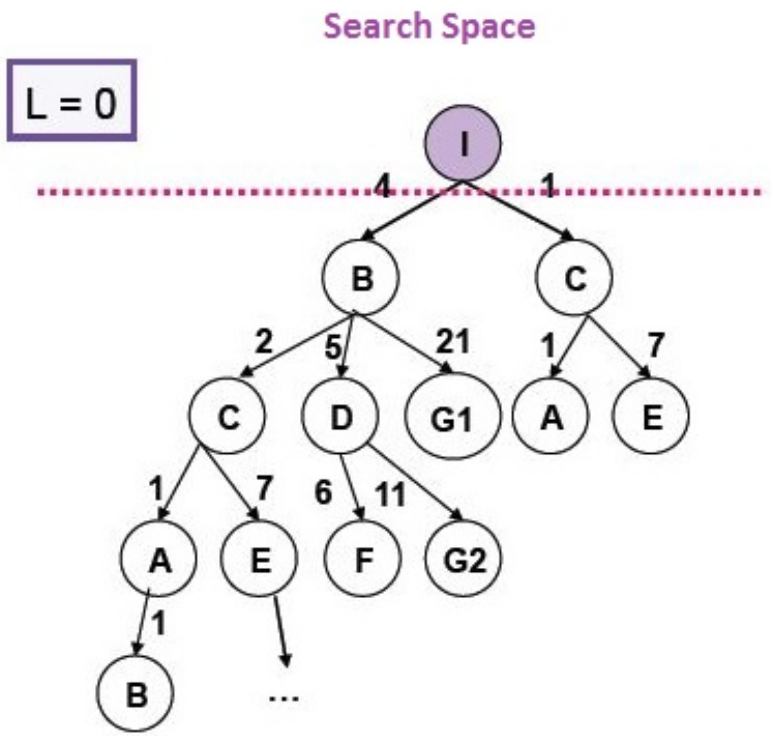


Expanded nodes : I B C A E D F G2

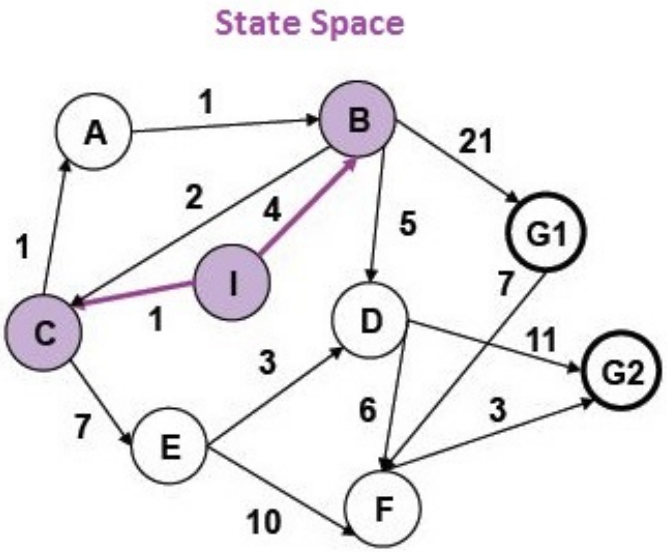
Solution path : I B D G2 **Cost** : 4+5+11 = 20

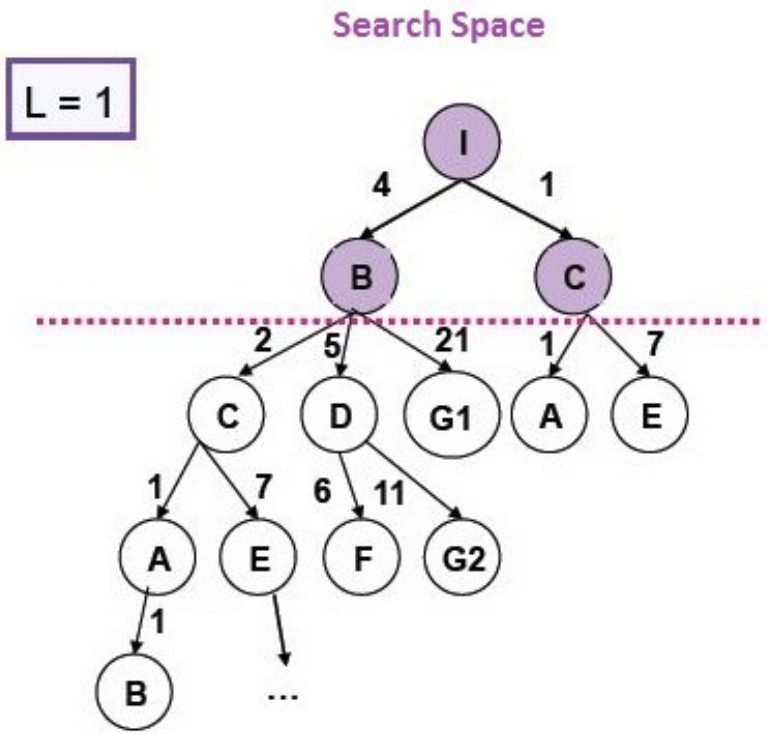
Example Iterative Search I





Example Iterative Search II



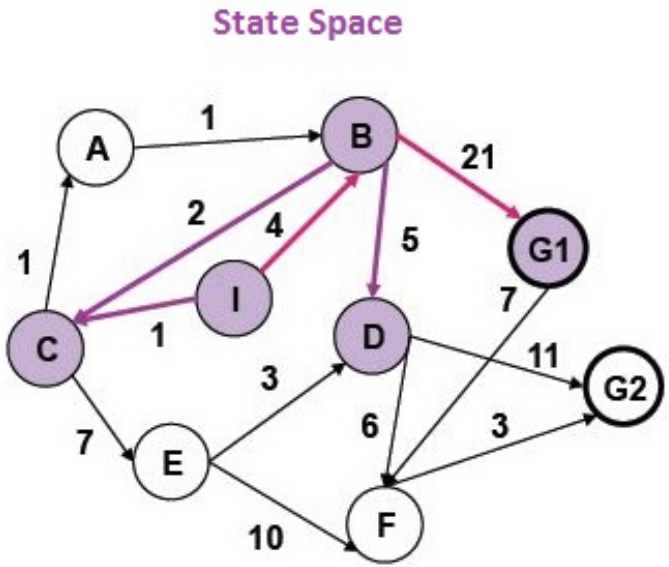


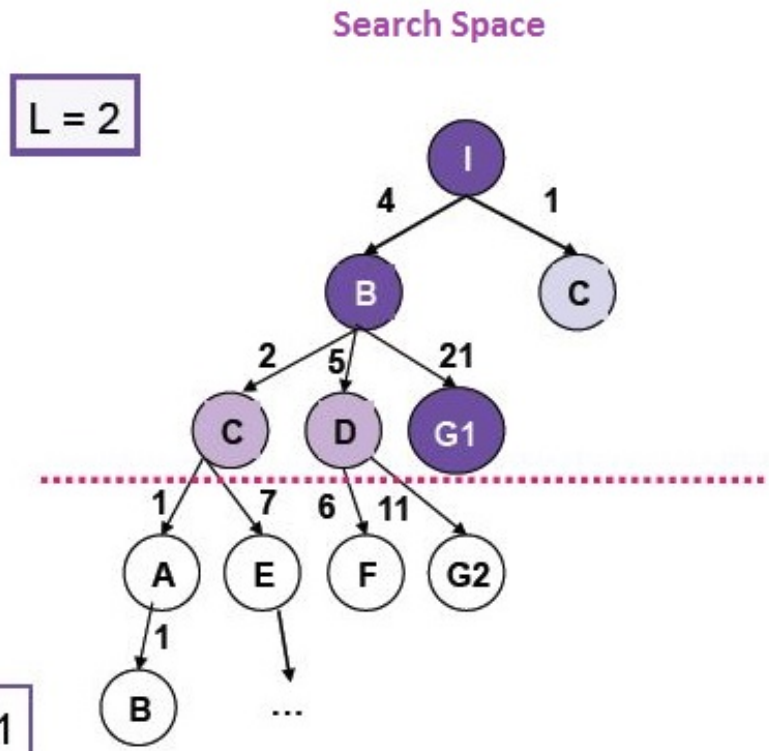
Optimal and Complete

Expanded nodes : I I B C

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

Example Iterative Search III



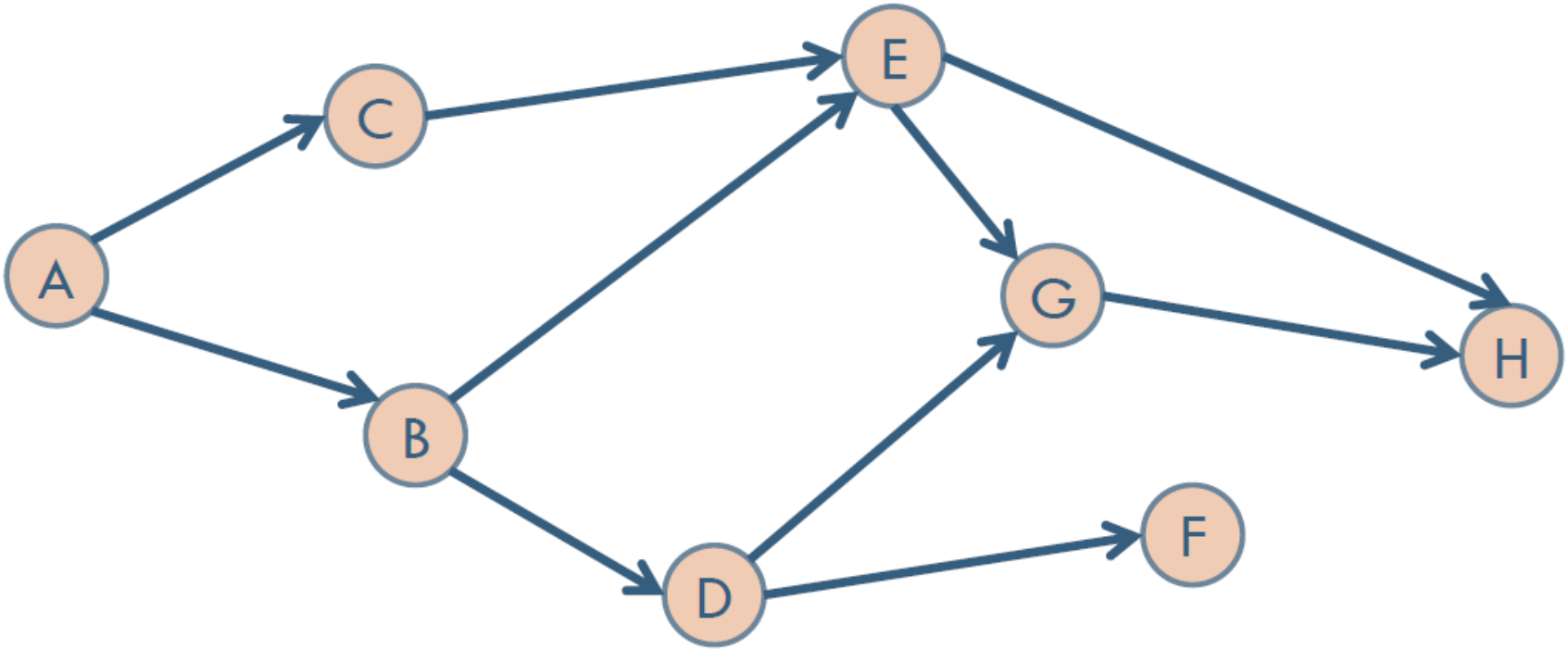


Expanded nodes : I I B C I B C D G1

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

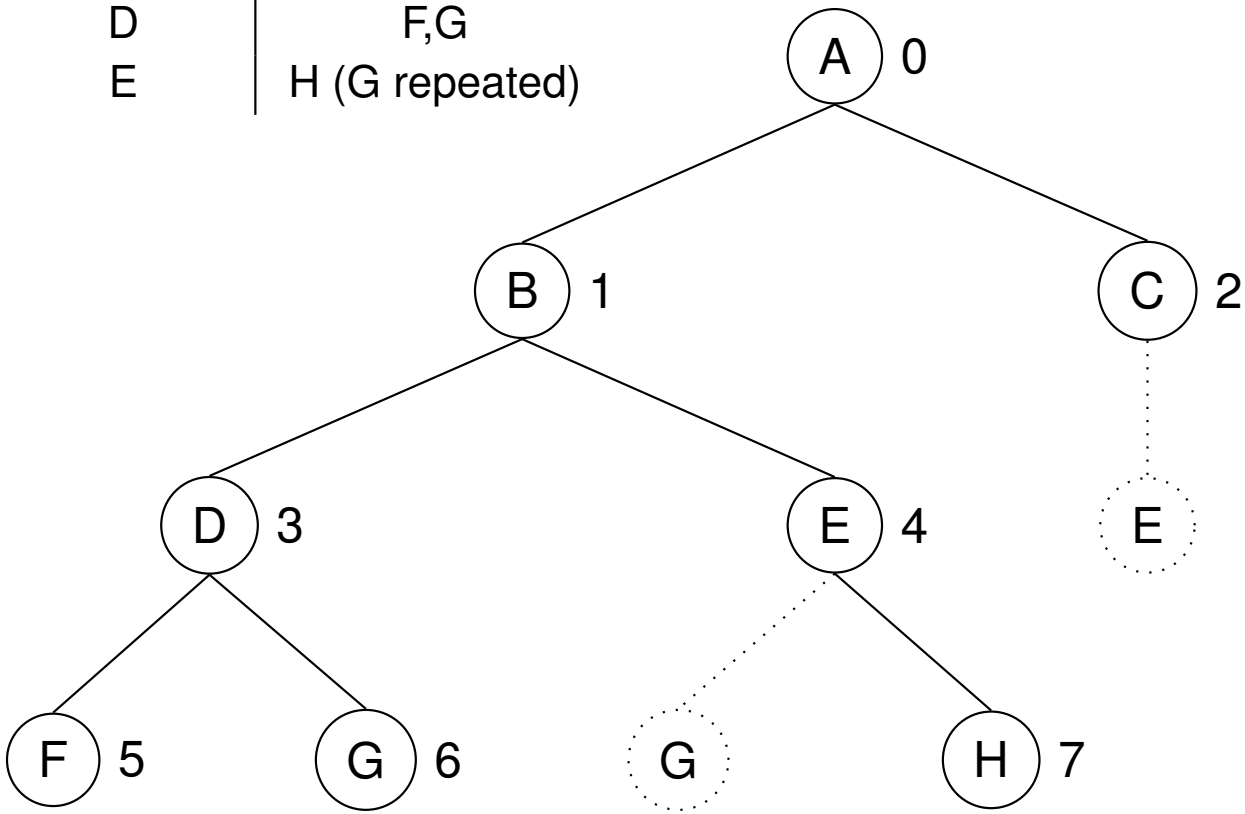
Exercise 1: Breadth First Search and Depth First Search

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.



Solution Exercise 1 – BFS

Open	Expanded	Generated
A	A	B,C
B,C	B	D,E
C, D, E	C	$\emptyset$ (E repeated)
D,E	D	F,G
E, F, G	E	H (G repeated)

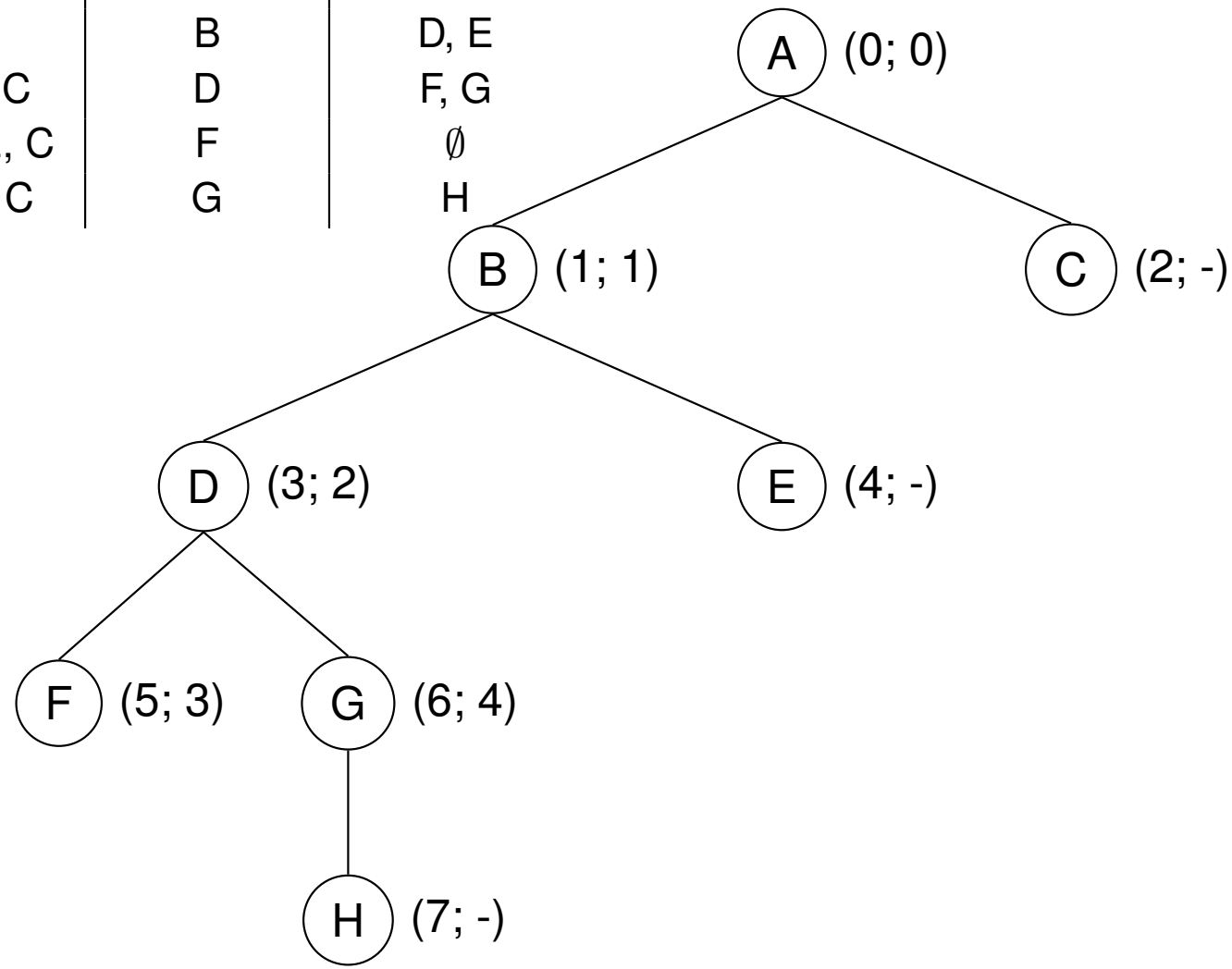


Numbers near nodes represent the generation order. Generation and expansion orders are the same in BFS

Solution (optimal): path A-B-E-H

Solution Exercise 1 – DFS

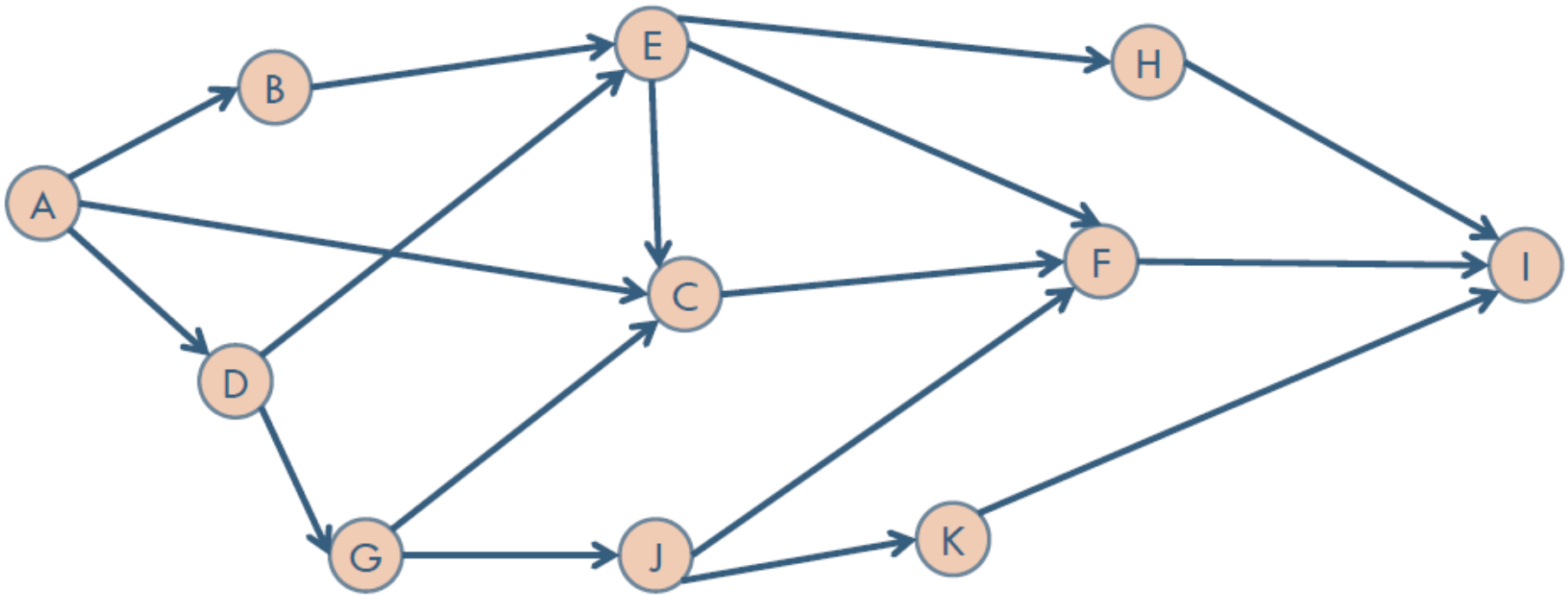
Open	Expanded	Generated
A	A	B, C
B,C	B	D, E
D, E, C	D	F, G
F, G, E, C	F	∅
G, E, C	G	H



Numbers near nodes represent the (generation; expansion) orders.

Exercise 2: Shortest path

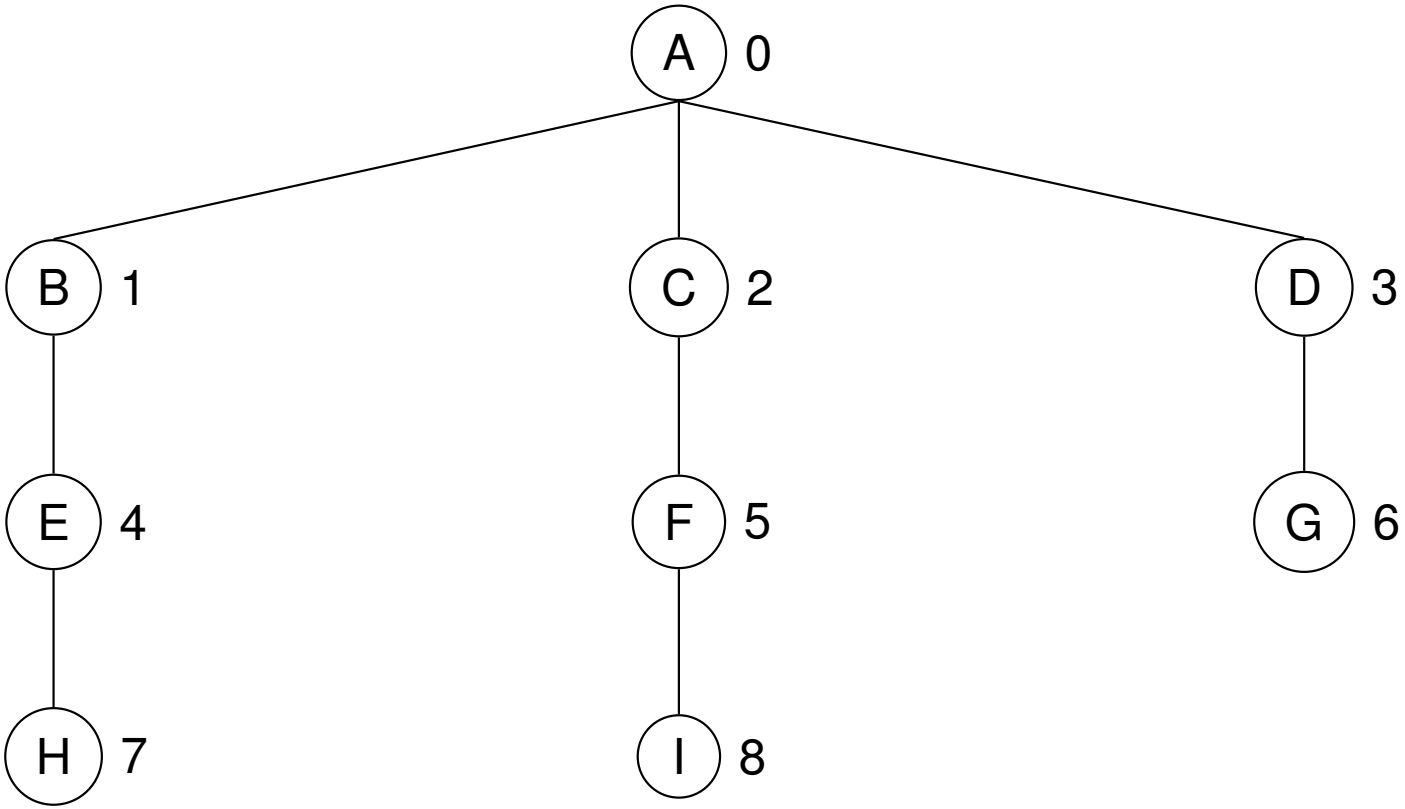
Find the shortest path (less edges) between A and I



Solution Exercise 2

To ensure shortest path $\rightarrow$ BFS

(Assumption: repeated states are pruned)

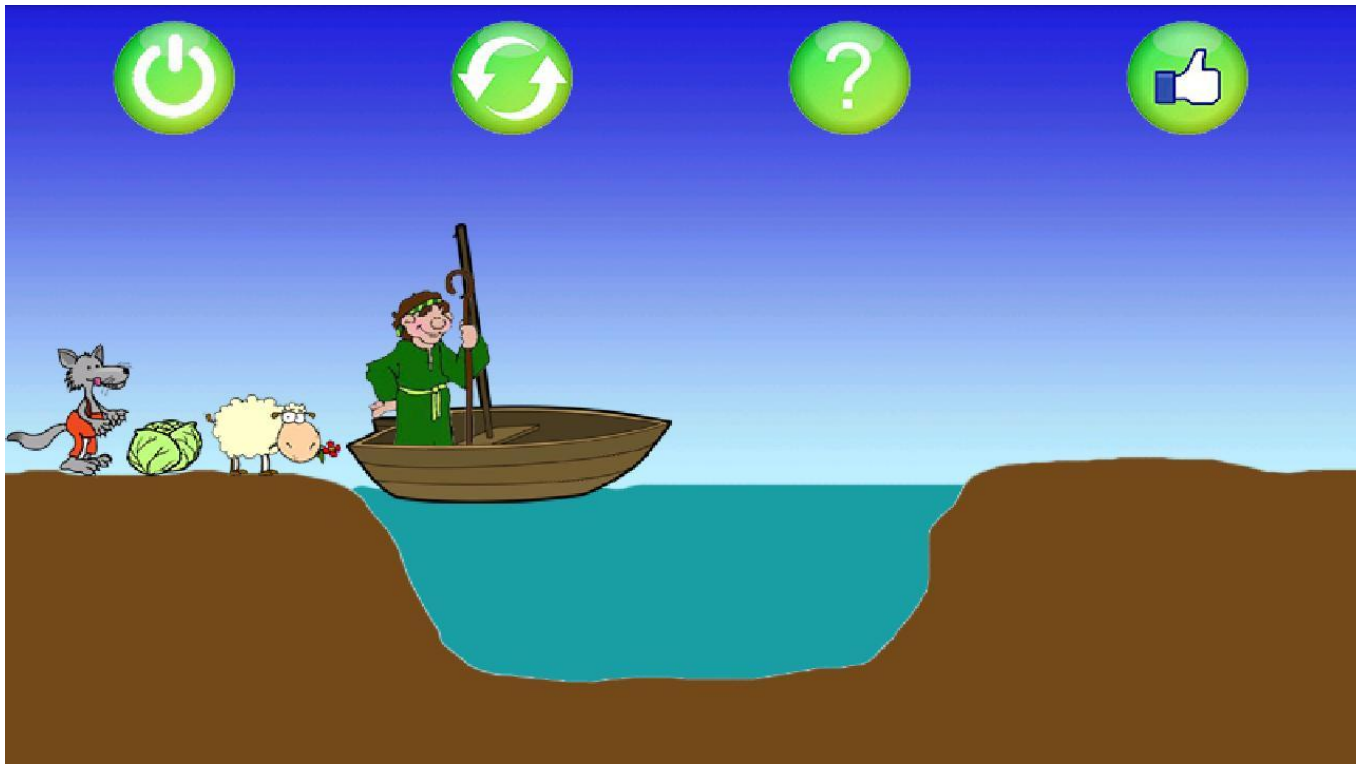


Solution (optimal): path A-C-F-I

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).

Solution Exercise 3 – Representation of states

Different possibilities

- Represent what is on each bank:
  - Initial state: ($\{$PWSC$\}$,$\emptyset$)
  - Goal state: ($\emptyset$,$\{$PWSC$\}$)
- Represent the bank of each character (P,W,S,C)
  - Initial state (0,0,0,0)
  - Goal state (1,1,1,1)

Solution Exercise 3 – Operators

We will assume the first representation of states

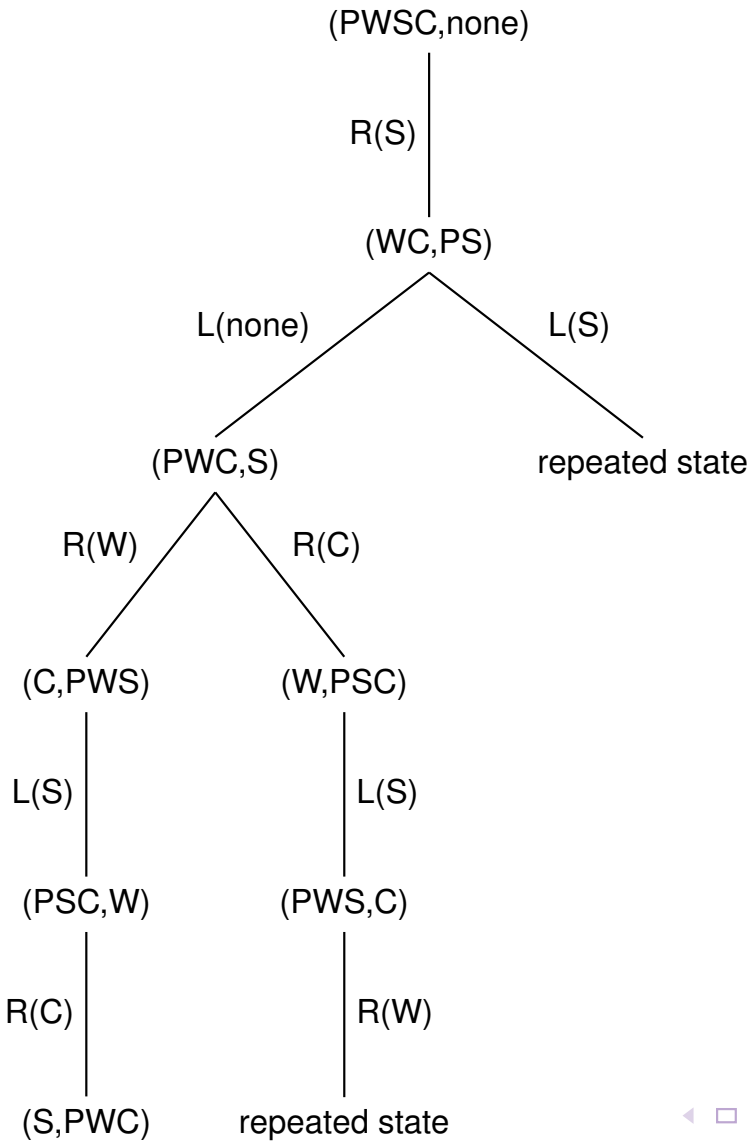
- R(X): the man takes X from the right bank to the left bank (X can be Wolf, Sheep, Cabbage or none)
- L(X): the man takes X from the left bank to the right bank (X can be Wolf, Sheep, Cabbage or none)

The application of operators should generate only valid states!

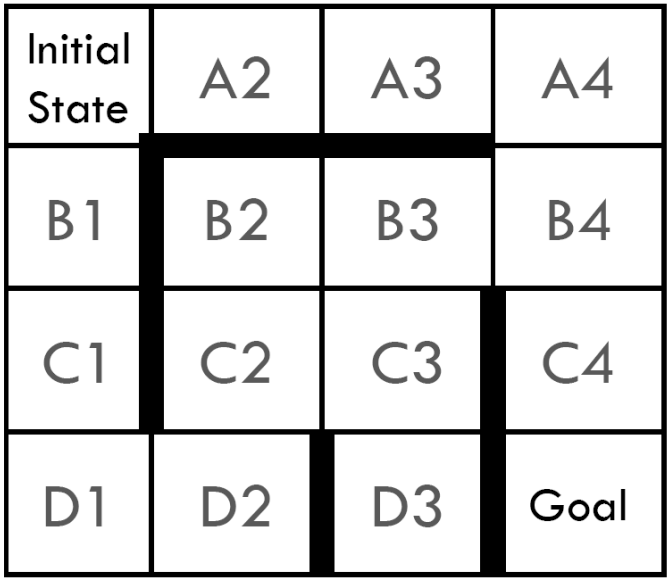
- Operator R(X) only applicable in the following states
  - If X=W: applicable in all states with PW in the right bank where S and C are not together in the right bank
  - If X=S: applicable in all states such that PS are on the right bank
  - If X=C: applicable in all states with PC in the right bank where W and S are not together in the right bank
  - If X = none: applicable in states with P in the right bank such that WS and SC are not together in the right bank
- Operator L(X): similar analysis for the left bank

Solution Exercise 3 – search

Breadth First Search (pruning repeated states)

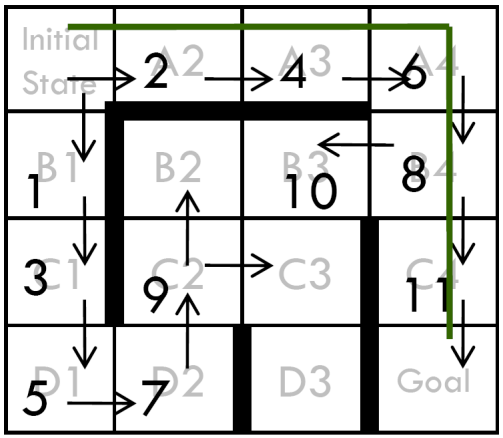


Exercise 4: maze

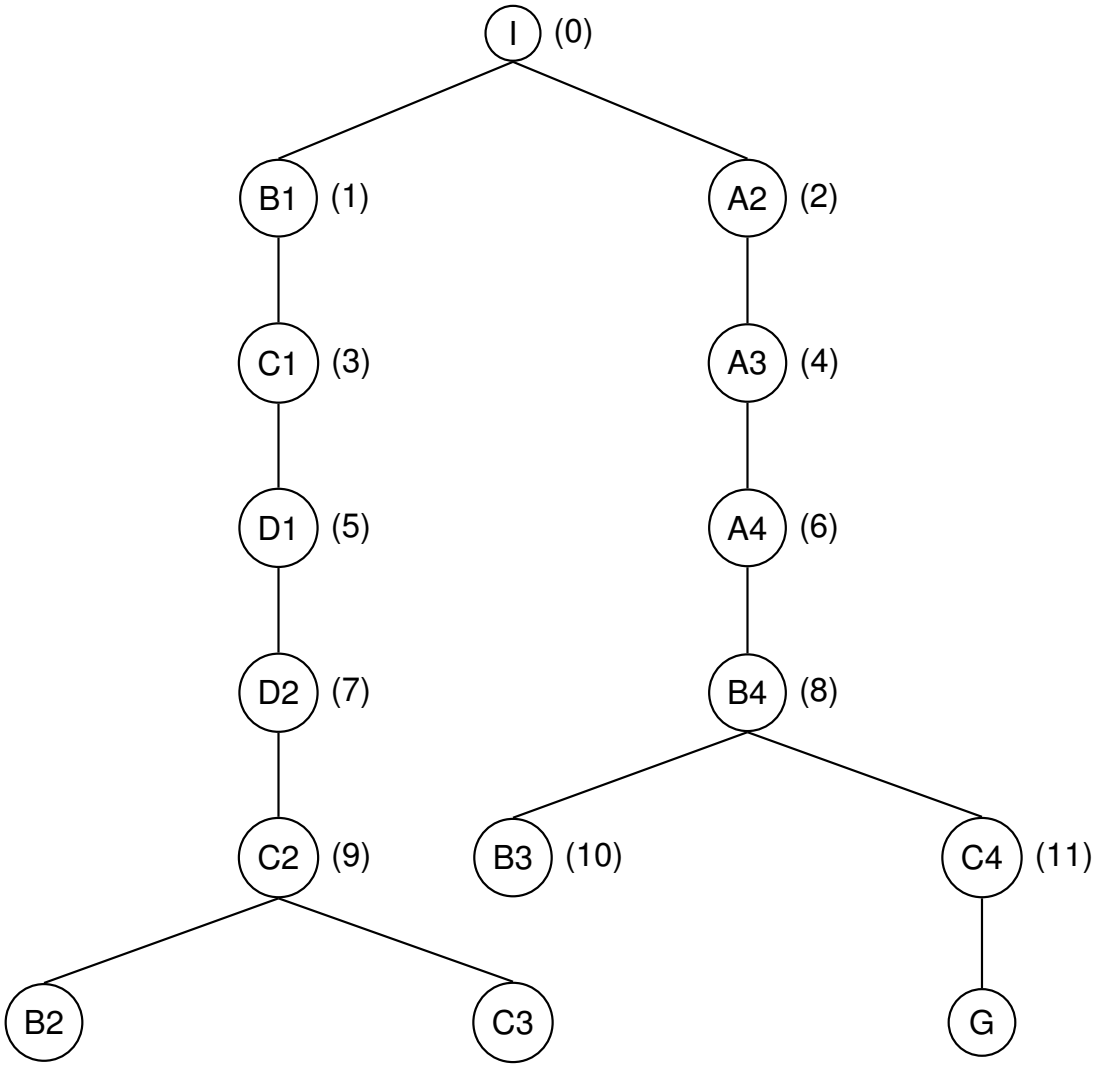


- The aim is to find a path to the goal
- Operators: Move up, move left, move down, move right
- Assuming the order given for the operators, use breadth first search to solve the maze problem. Mark every cell with the number of expanded node
- Apply depth first search for the same problem
- Change the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

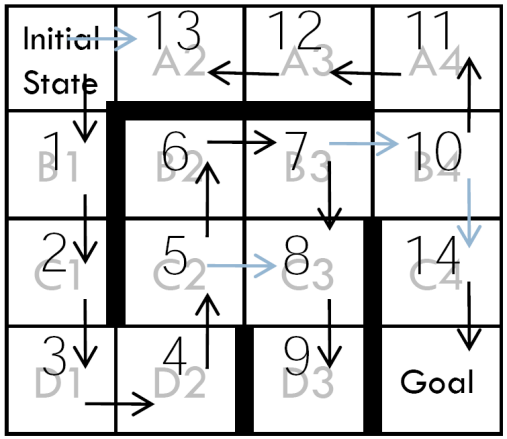
Solution Exercise 4: maze – BFS



Repeated states are pruned
Numbers represent the expansion order

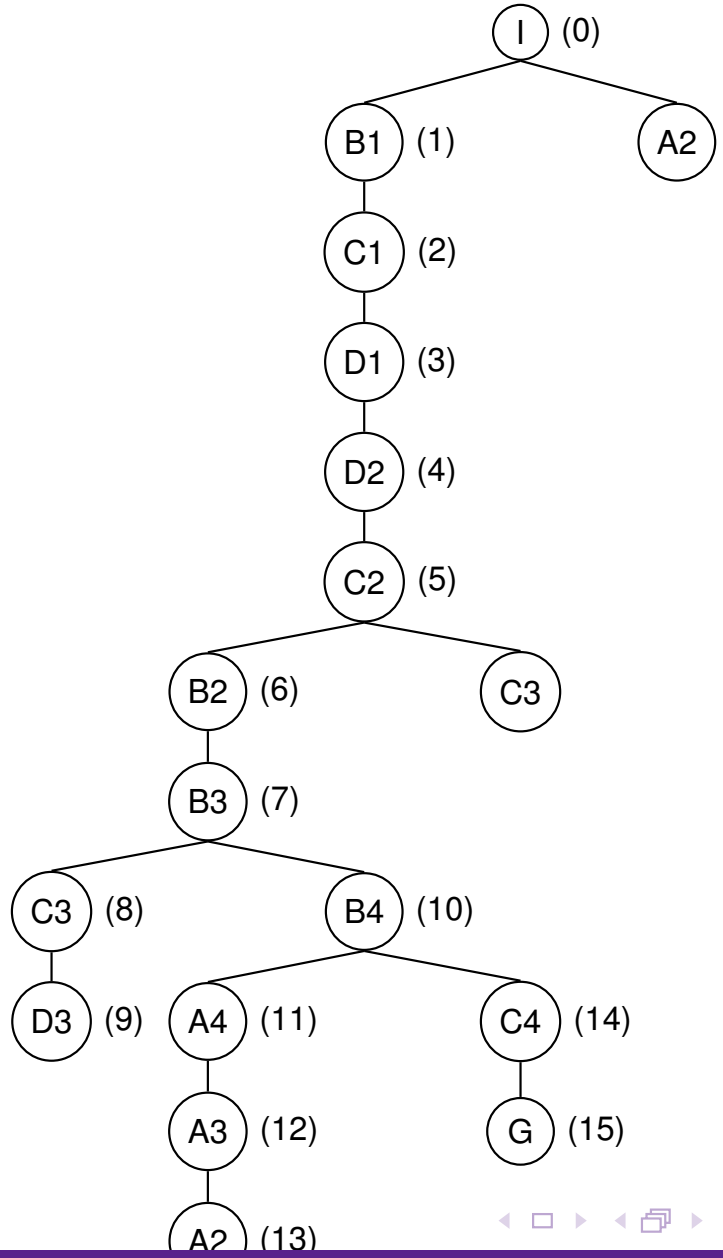


Solution Exercise 4: maze – DFS

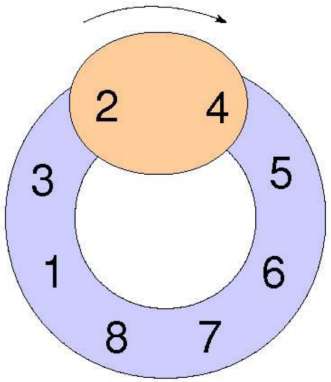


Repeated states on the current path from the initial state are pruned

Numbers represent the expansion order

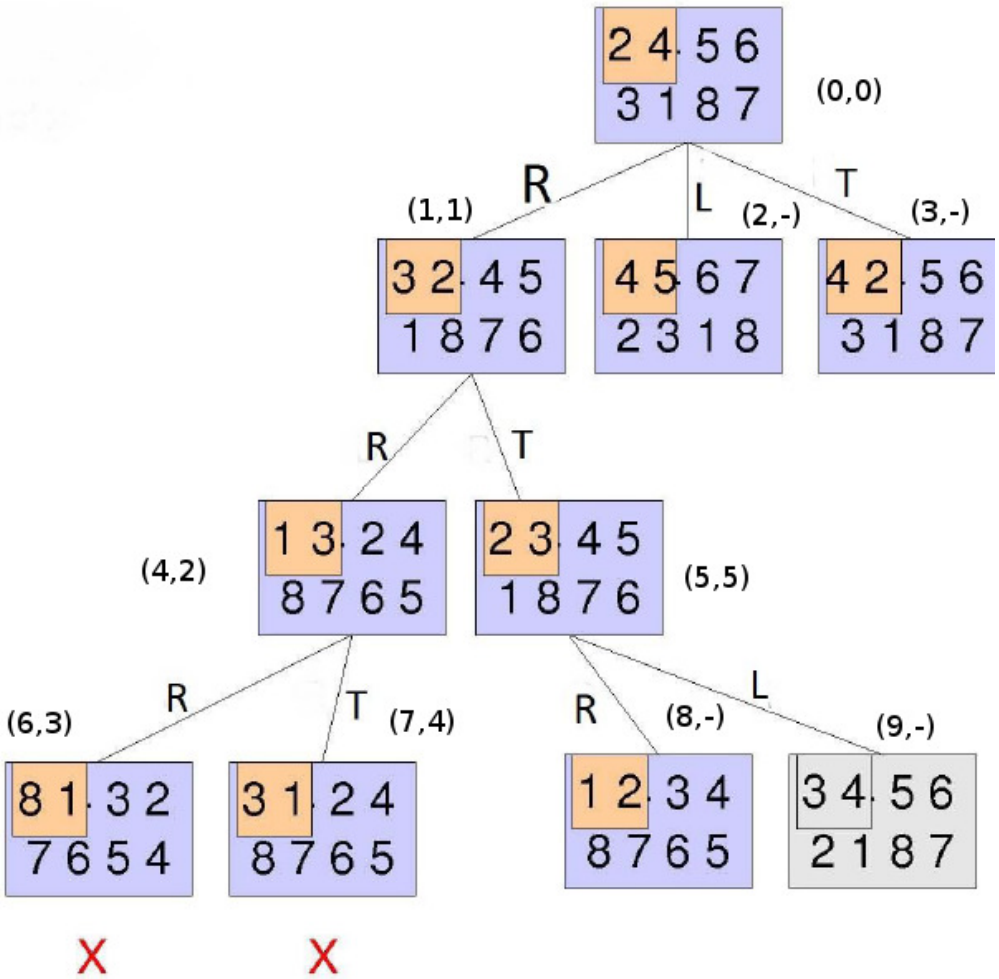


Exercise 5: top spin

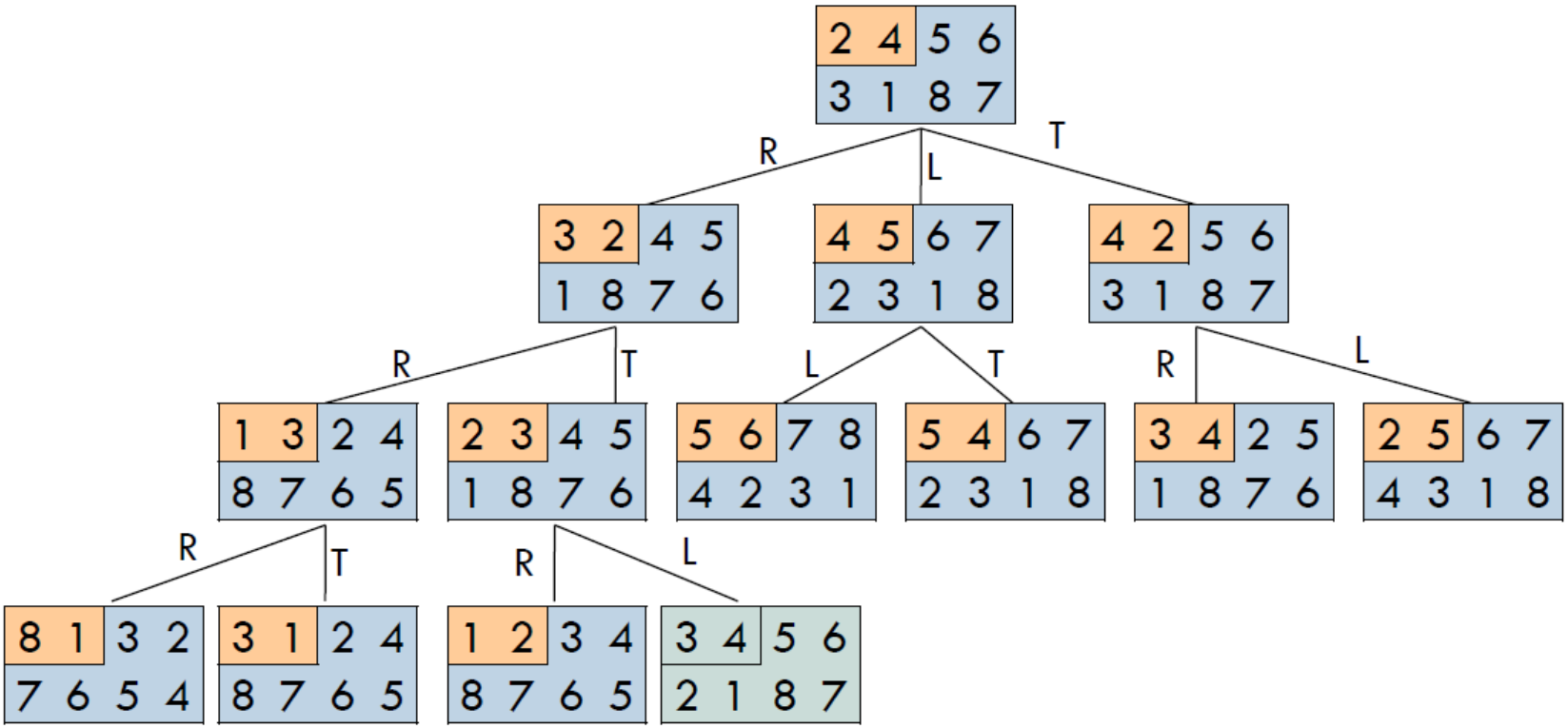


- Permutation game in which we rotate the base to exchange the order of numbers
- Goal state: numbers are ordered from 1 to 8 with the pair (1 2) in the base
- Operators: move numbers to the right, move numbers to the left, rotate base to switch order of the numbers in it
- Apply DFS to solve the top-spin 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?

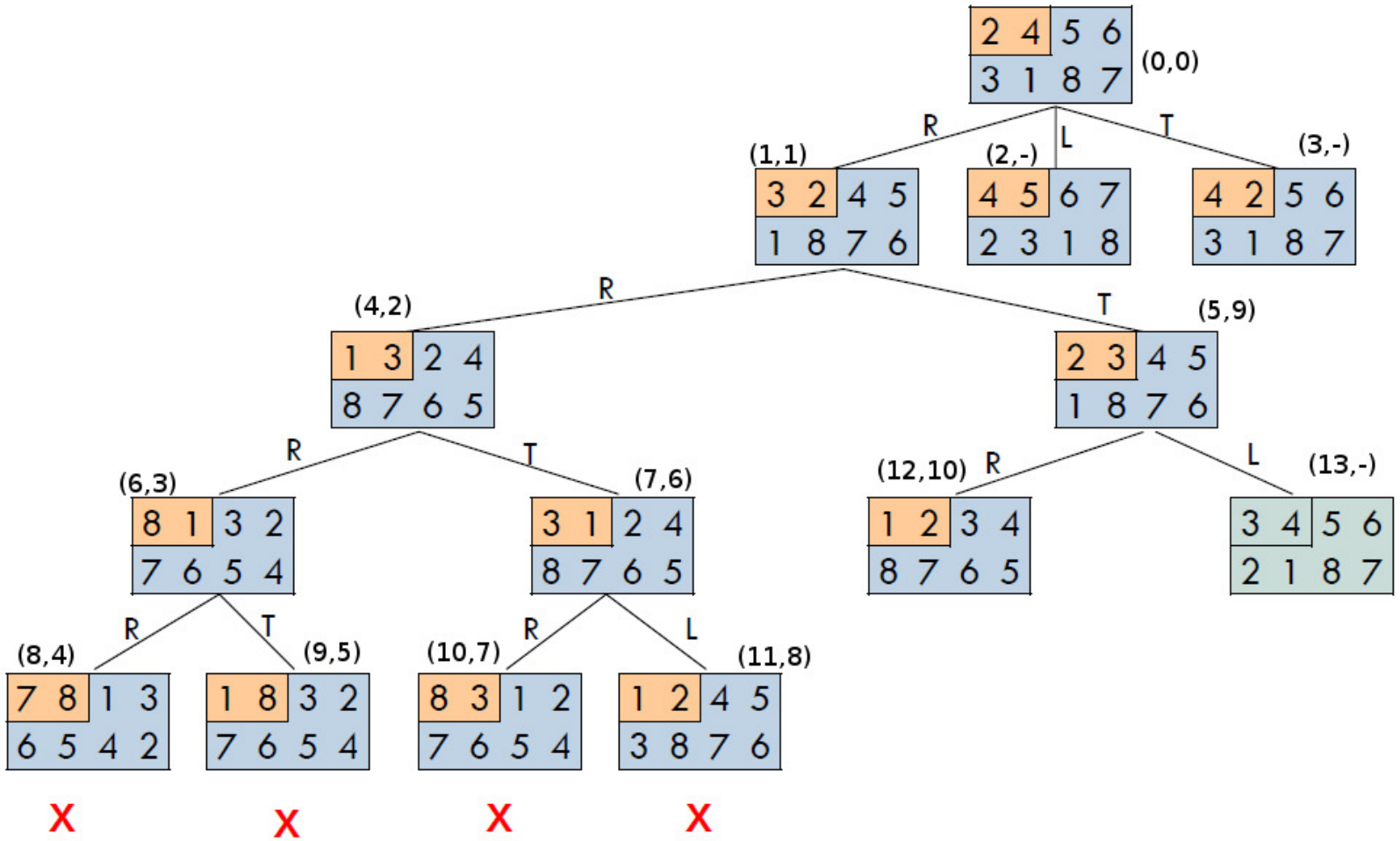
Solution Exercise 5: top spin – DFS (d=3)



Solution Exercise 5: top spin – BFS



Solution Exercise 5: top spin – DFS (d=4)



Exercise 6: Space-State Representation problem

- Description: Missionary and Cannibal
  - 3 missionaries and 3 cannibals are by the edge of a rive near a boat
  - the goal is for them to cross the river
  - Restrictions:
    - In the boat only 1 or 2 people can cross
    - In the edges of the river there cannot be more cannibals than missionaries
- Represent the problem as a space-state representation problem, draw the space-state

Solution Exercise 6

- Representation: Abstract as much as you can
- You could use a concept for each character: M1, M2, M3, C1, C2, C3
- And have states of the kind (M1, M2, M3, C1, C2, C3,Boat)
- too much, try to simplify

Solution Exercise 6

- State= (Number M, Number C, Position boat)
- cost = number of time you cross the river
- initial state (3,3,1)
- final state (0,0,0)
- allowed states (2,2,0), (3,2,1)

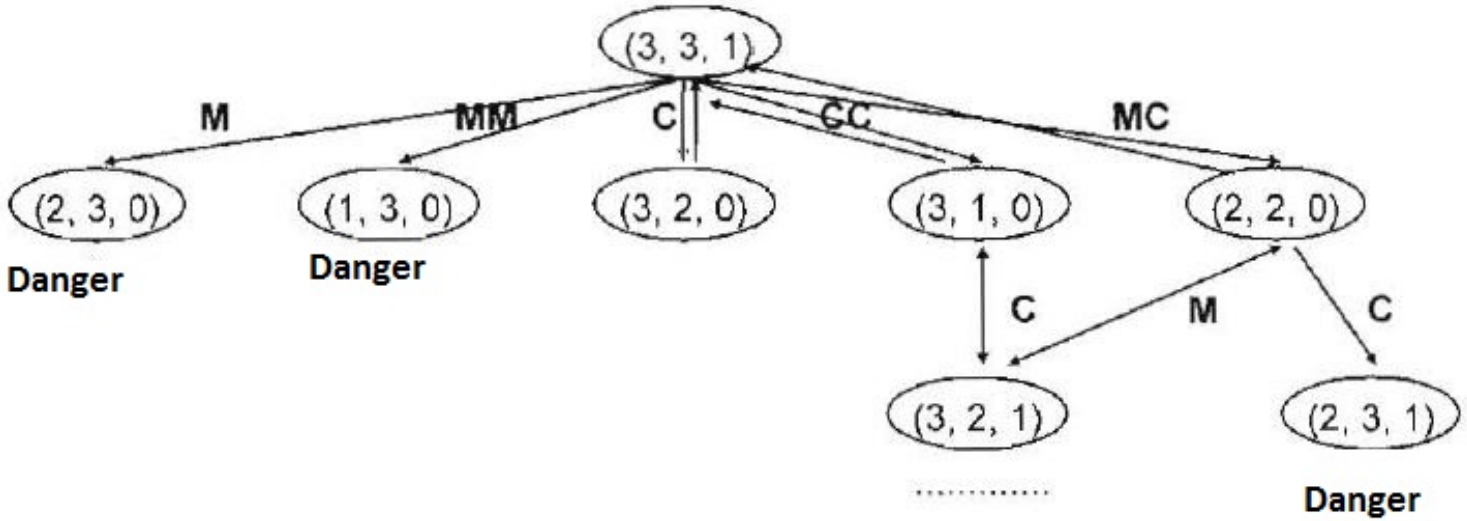
Solution Exercise 6: Operators

- M
- MM
- C
- CC
- MC
- note that the position of the boat is key
- CrossM (NM,NC,B):
  - Precondition: $(NM > 0 \wedge B = 1) \vee (NM < 3 \wedge B = 0)$
  - Action: $IfB = 1 then NM = NM - 1 \wedge B = 0 = (NM - 1, NC, 0)$
  - Action: $IfB = 0 then NM = NM + 1 \wedge B = 1 = (NM + 1, NC, 1)$

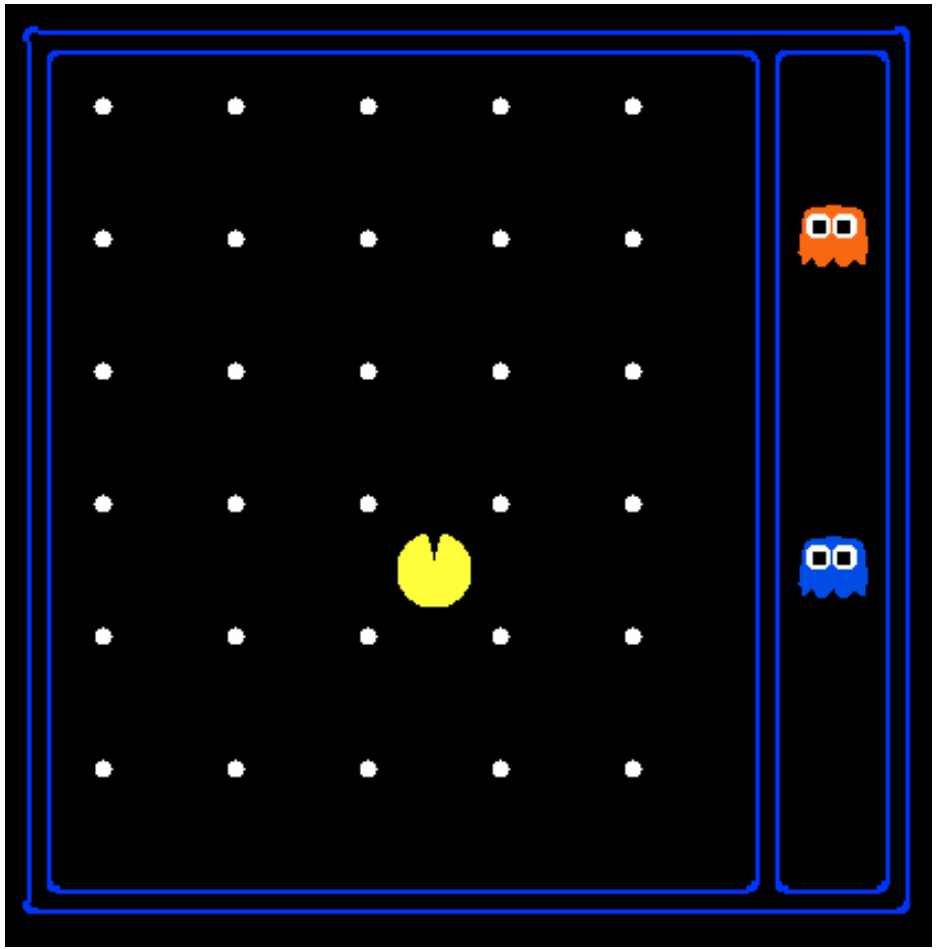
Solution Exercise 6: Danger

- (3,3,1) and (2,2,0) are not dangerous
- (1,2,0) and (2,3,0) are dangerous
- is it enough with $NM < NC$. No (0,2,1)
- then: $(NM < NC \wedge NM \neq 0) \vee (NM > NC \wedge NM \neq 3)$

Solution Exercise 6: Danger Space-State



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?

Solution Exercise 0 - Part 1

- How is the domain state characterized? Pacman's position, if there is/isn't food in a position where there was initially food, the current position
- How many possibilities do we have? World state:
  - Agent positions: 120
  - Food count: 30
  - Ghost positions: 12
  - Agent facing= where does Pacman look?: NSEW = 4
- Then we have $120 \times (2^{30}) \times (12^2) \times 4$ states

Solution Exercise 0 - Part 2

- In states for the search problem where Pacman only has to reach a final position given and initial position we only have to represent Pacman's current position
- Then we have 120 posible states. That is, the size of the space state is 120
- The initial state is defined by the initial position (xi, yi). The final state is defined by the final position (xf, yf)
- Actions: N, S, E, W. Actions move Pacman one position to the direction the action indicates.

Solution Exercise 0 - Part 3

- In states for the search problem where Pacman has to reach eat all the food balls we have to represent Pacman's position and if there is/isn't a ball in each of the positions where there could be one. This can be represented with booleans (yes,no) for each of these positions.
- Then we have 120×2^{30} possible states. This is the size of the space state.
- The initial state is defined by the initial position (xi, yi) and all the booleans of the balls are true. The final states are those where all the booleans representing balls are false
- Actions: N, S, E, W. Actions move Pacman one position to the direction the action indicates and they make disappear the balls in the case a new position matches one of them.