SAPIENZA
UNIVERSITÀ DI ROMA

Fondamenti di Intelligenza Artificiale

2023/2024 Prof: Sara Bernardini

Lab 3: Search, Giochi e CSP

Francesco Argenziano
email: argenziano@diag.uniroma1.it

*The slides have been prepared using the textbook material available on the web, and the slides of the previous editions of the course by Prof. Luigia Carlucci Aiello, Prof. Daniele Nardi and Dott. Fabio Previtali.

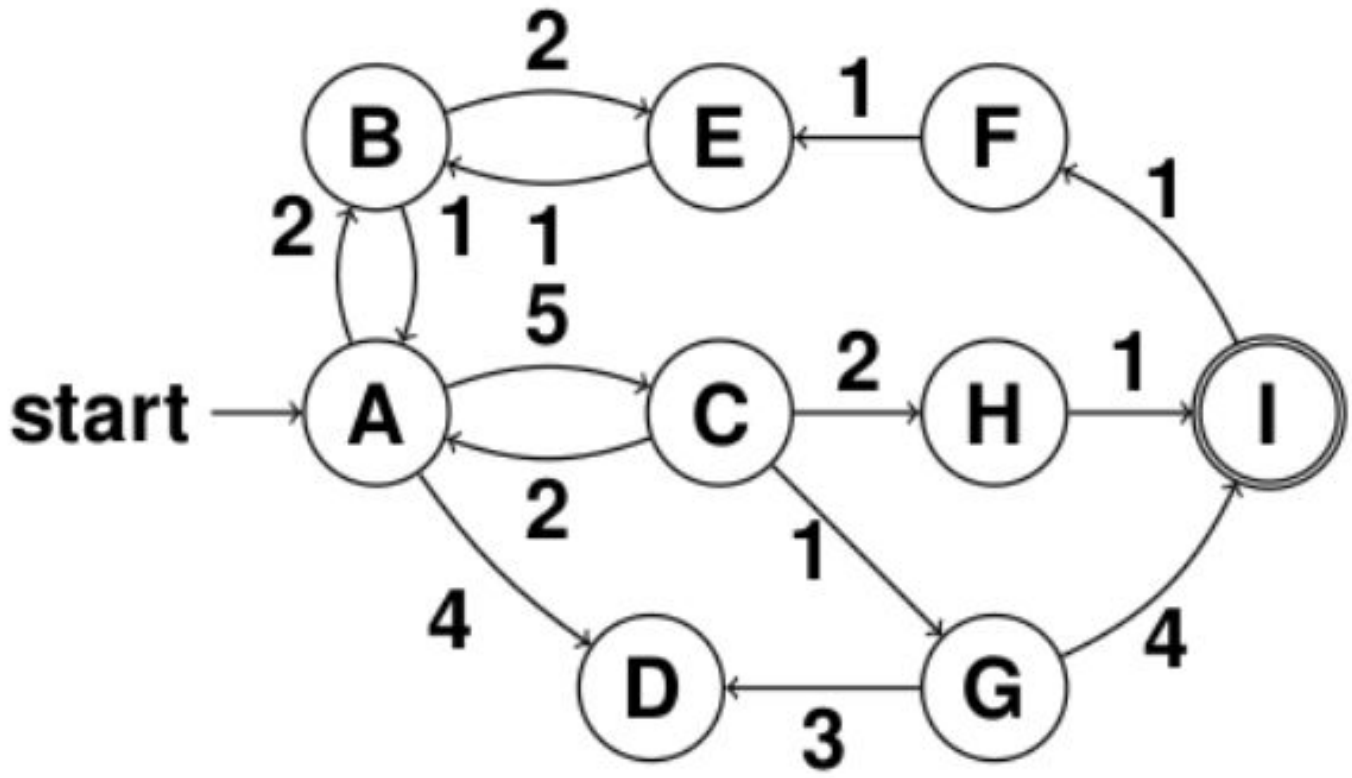


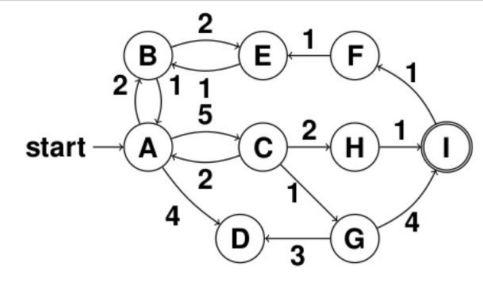
Figure 3: State space

- (a) Run the A^* search algorithm on this problem. As a heuristic estimate for a state s, use the minimal number of edges that are needed to reach a goal state from s (or ∞ if s is not solvable, e.g., h(C)=2). Draw the search graph and annotate each node with the g and h value as well as the order of expansion. Draw duplicate nodes as well, and mark them accordingly by crossing them out. If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first. Give the solution found by A^* search. Is this solution optimal? Justify your answer.

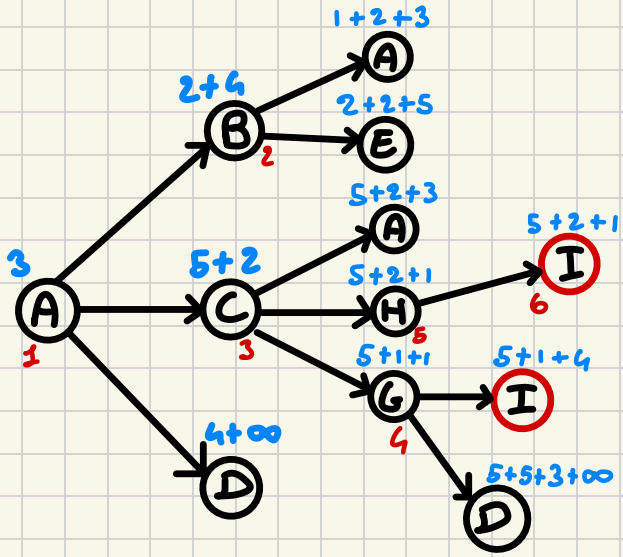
- (b) Run the hill climbing algorithm on this problem. Use the heuristic function from part (a). For each state, provide all applicable actions and the states reachable using these actions. Annotate states with their heuristic value. Specify which node is expanded in each iteration of the algorithm. If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first. Does the algorithm find a solution? If yes, what is it and is it optimal?

- (c) Could the hill-climbing algorithm stop in a local minimum without finding a solution? If yes, give an example heuristic h : $\{A, B, ..., I\} \to \mathcal{N}_0^+ \cup \{\infty\}$ for the state space depicted that leads hill-climbing into a local minimum, and explain what happens. If no, please explain why.

a)



S	h(s)
A	3
B	4
C	2
D	∞
E	5
F	6
G	1
H	1



SOLUZIONE OTTIMA: A-C-H-I

A* TROVA SEMPRE PERCORSI OTTIMALI PERCHÈ CONSIDERA I POSSIBILI PERCORSI MIGLIORI

b) 1) A → B ; h(B) = 4
A → C ; h(C) = 2
A → D ; h(D) = ∞

2) C → A ; h(A) = 3
C → H ; h(H) = 1
C → G ; h(G) = 1

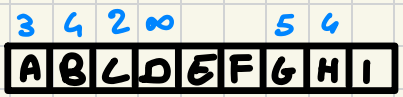
3) G → D ; h(D) = ∞
G → I ; h(I) = 0

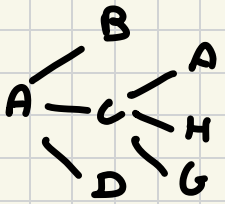
4) I → F ; h(F) = 6

MI FERMO PERCHÈ h(F) > h(I)

TROVA UNA SOLUZIONE (MINIMO LOCALE) MA NON È OTTIMA

c)



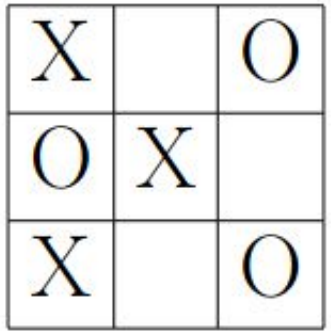


C TROVA DELLE h PIÙ GRANDI E SI FERMA SU C CHE NON È SOLUZIONE

Example: Tic-Tac-Toe

Tic-Tac-Toe is played on square grid of size 3 × 3 At each turn, players select an empty cell and place there its own symbol (i.e., O or X) A player wins when he places three of its own symbols in a line (vertical, horizontal or diagonal)

If the grid is filled without any player being able to place three symbols in a line, the game ends in a draw.

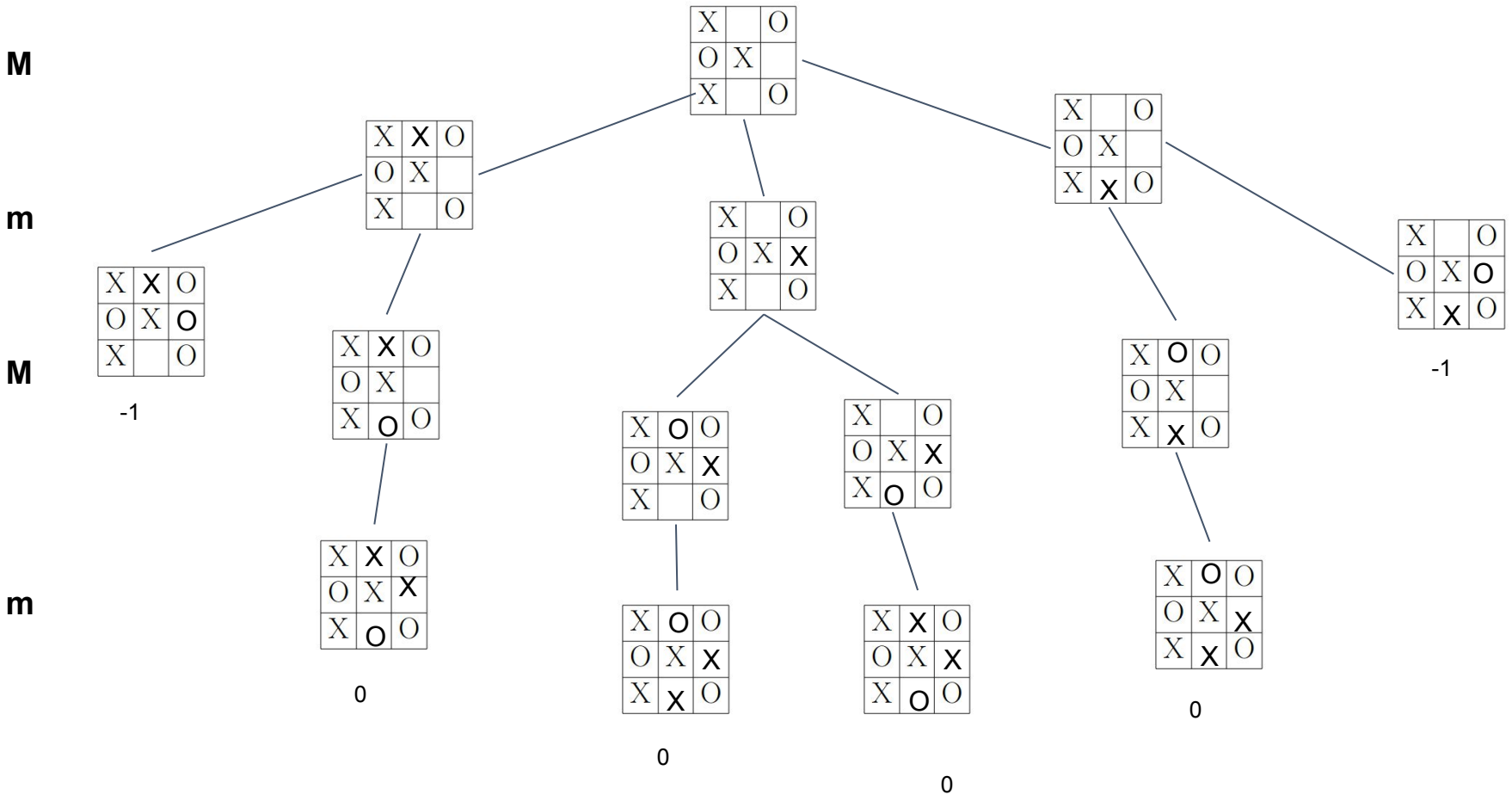


1. Suppose that the game is in the state described by the previous picture and that X must move next. Draw the game tree of the rest of the match.

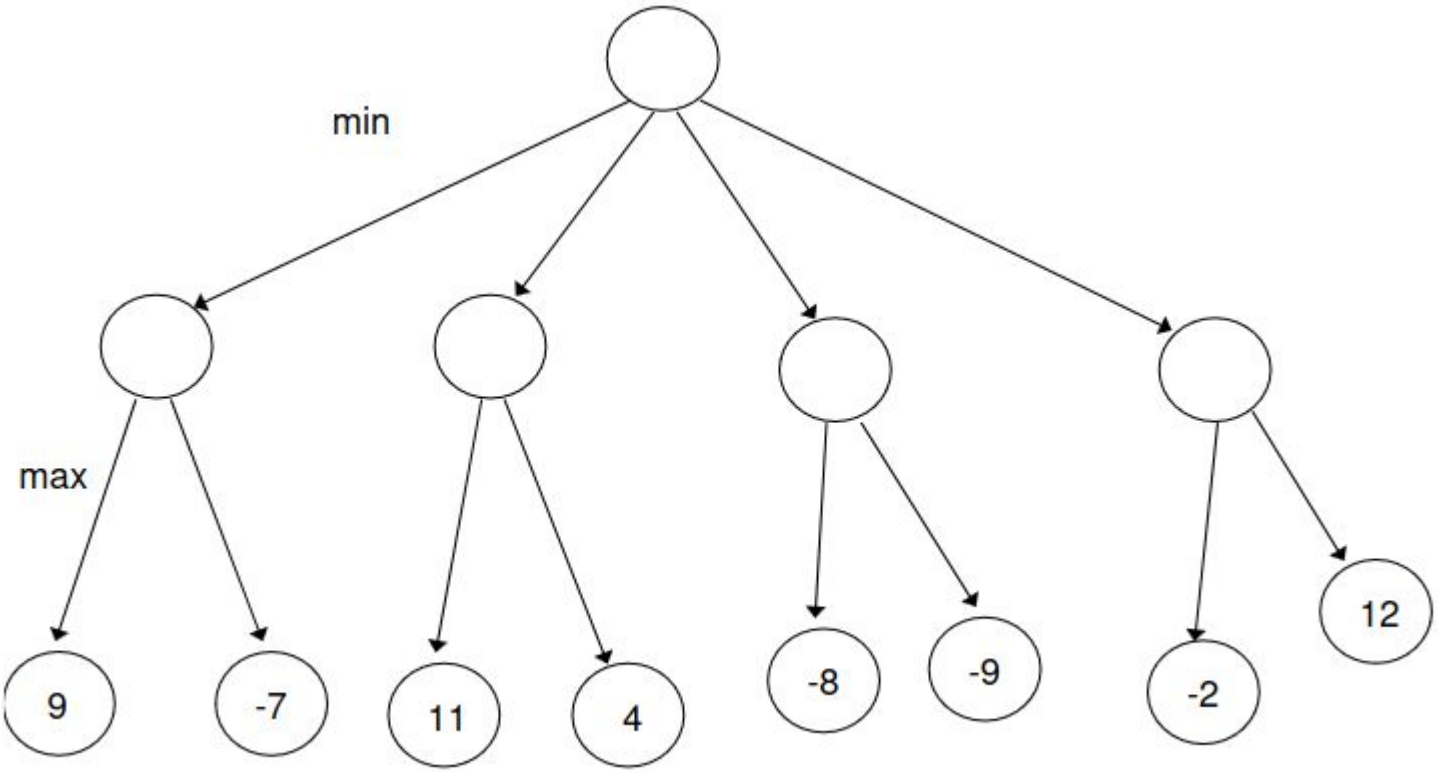
2. Show the solution of the game using minimax, knowing that payoffs, are for each player, -1, 0, +1 depending on if he loses, draws or wins, respectively.

Example: Tic-Tac-Toe

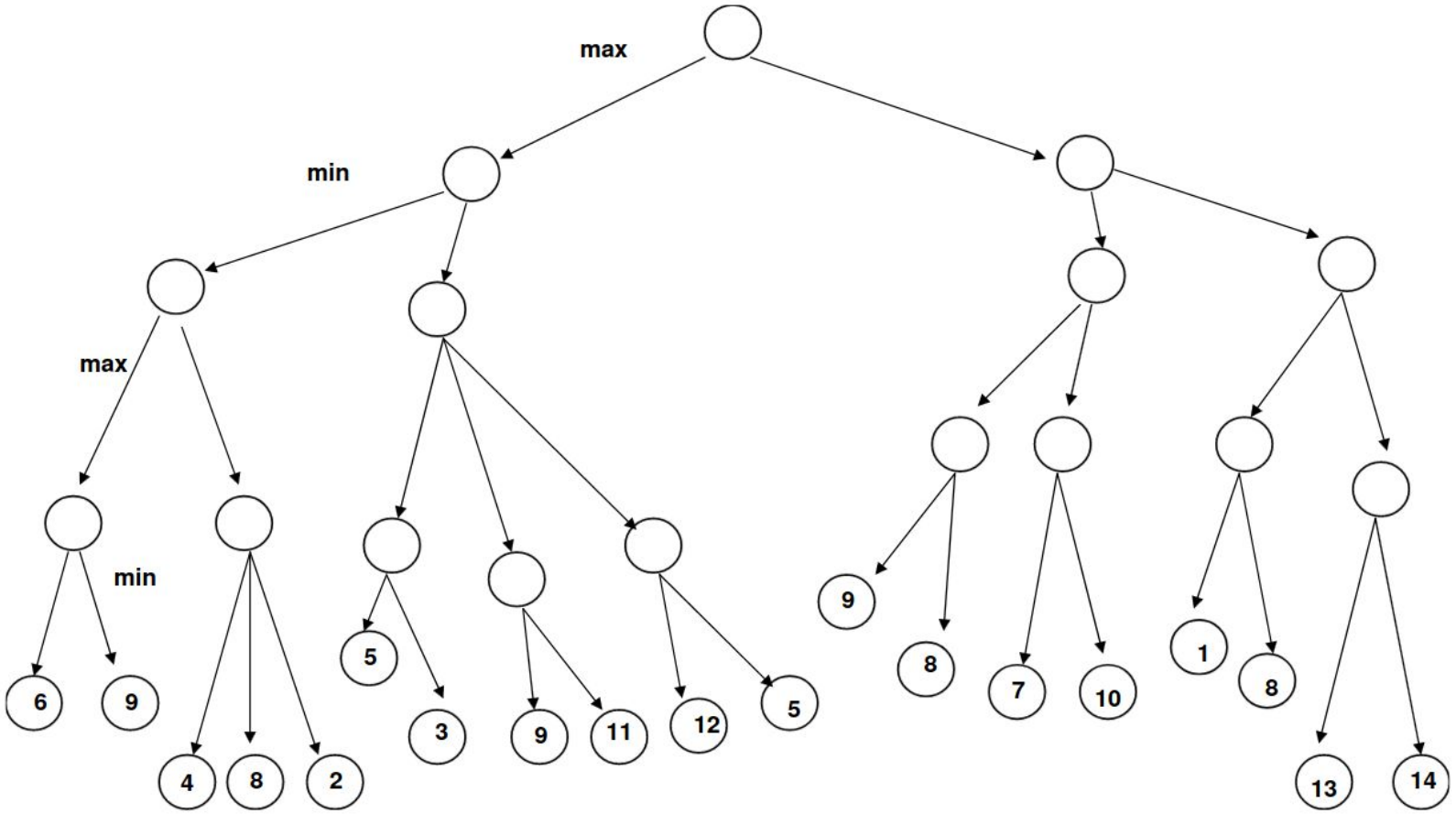
Max -> X
Min -> O



Example: Minimax 1



Example: Minimax 2



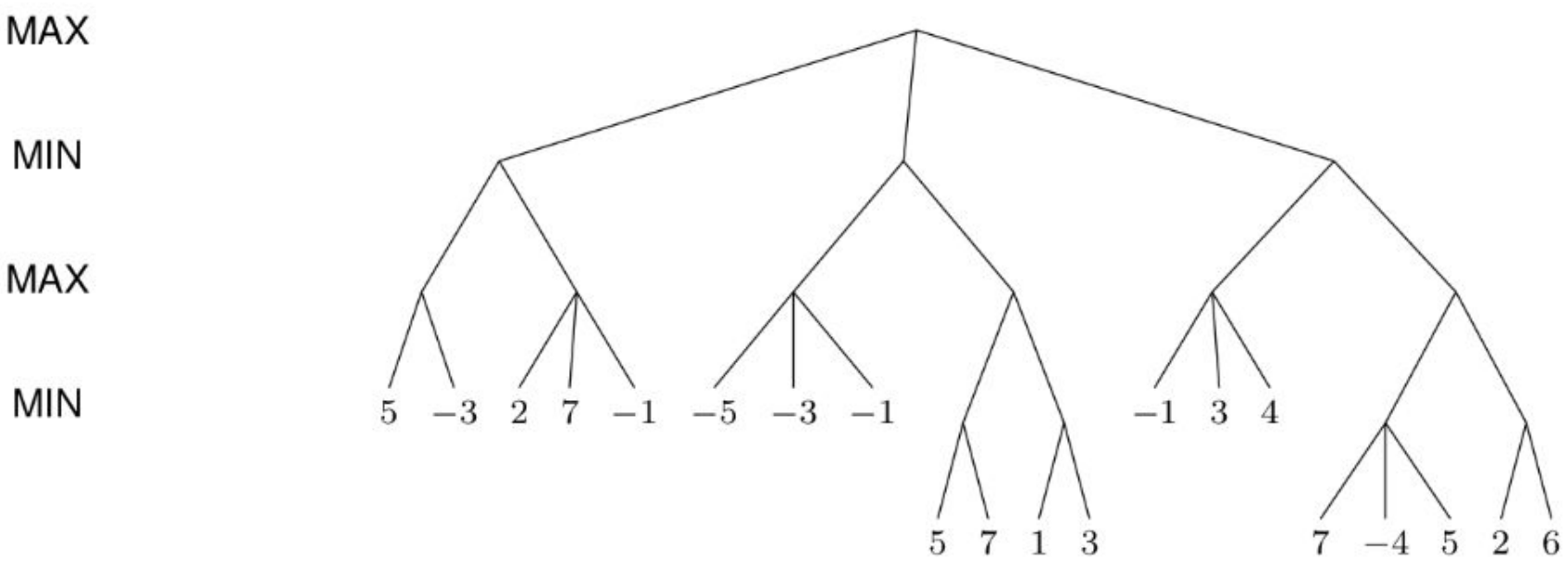
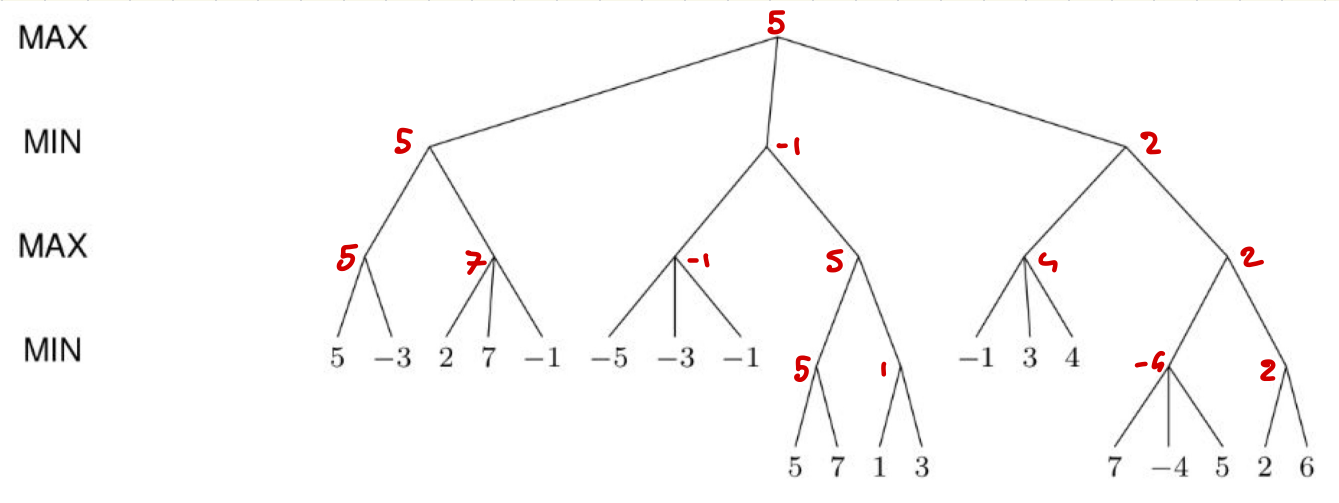


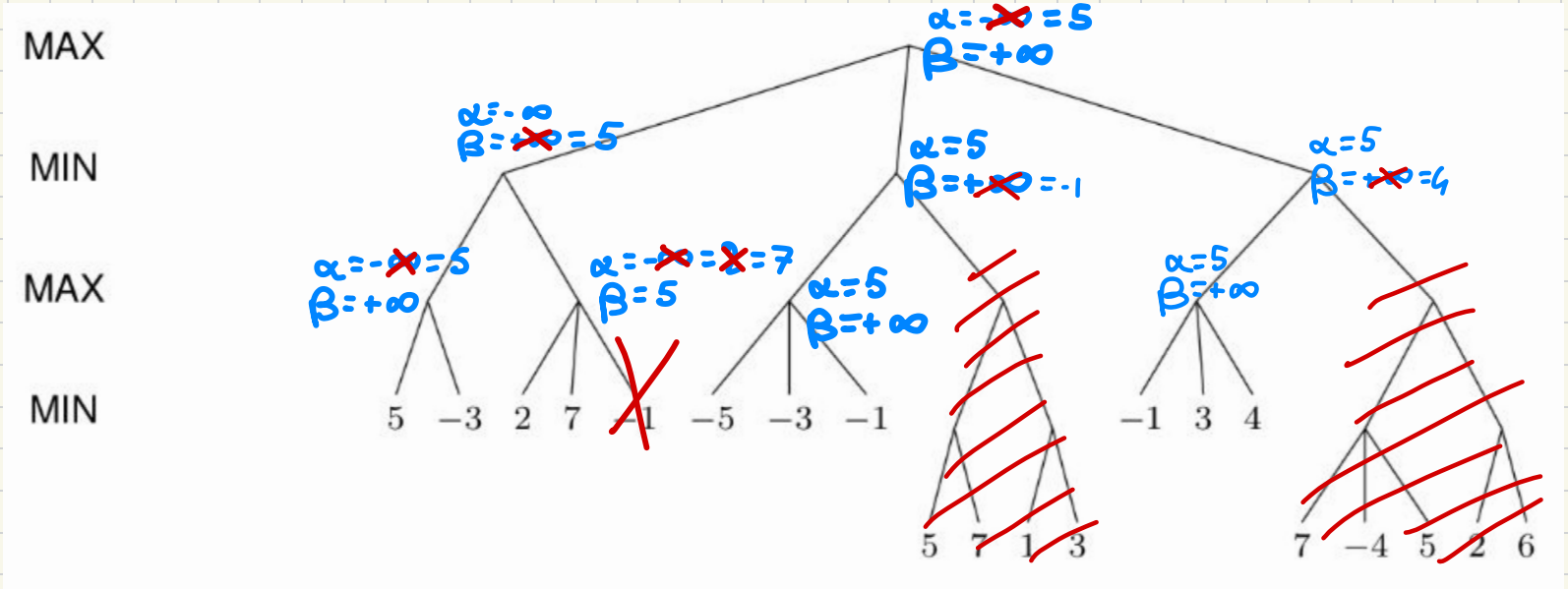
Figure 2: Game tree

- (a) Perform Minimax search on the tree, i.e., annotate all internal nodes with the correct Minimax value. Which move does Max choose?
- (b) Perform Alpha-Beta search on the tree. Annotate all internal nodes (that are not pruned) with the value that will be propagated to the parent node as well as the final $[\alpha, \beta]$ window before propagating the value to the parent. Mark which edges will be pruned. How many leaf nodes are pruned?
- (c) If possible, provide an example of reordering of leaf nodes that will results with more pruning performed.

a)


MAX
MIN
MAX
MIN
5
5
-1
2
5
7
-1
5
4
2
5 −3 2 7 −1 −5 −3 −1
5
1
−1 3 4
-4
2
5 7 1 3
7 −4 5 2 6

b)


MAX
MIN
MAX
MIN
α=-∞=5
β=+∞
α=-∞
β=+∞=5
α=5
β=+∞=-1
α=5
β=+∞=4
α=-∞=5
β=+∞
α=-∞=2=7
β=5
α=5
β=+∞
α=5
β=+∞
5 −3 2 7 −1 −5 −3 −1
−1 3 4
5 7 1 3
7 −4 5 2 6

Example: Scheduling Tasks

Consider the following constraint network $\gamma = (V, D, C)$. Lisa is moving out of Egham soon, and has many tasks to finish in only 5 days. These tasks are:

(a) arrange a farewell party for her friends,

(b) book a bus ticket,

(c) clean her apartment,

(d) de-register from the council,

(e) end all her contracts,

(f) find a subtenant for her apartment,

(g) get packing cases.

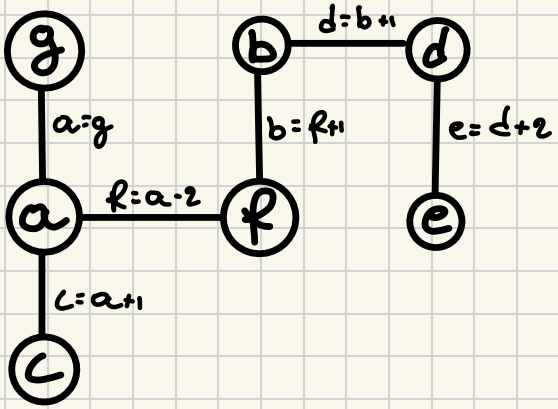
The time slots in which she can schedule these tasks are Monday (M), Tuesday (Tu), Wednesday (W), Thursday (Th) and Friday (F). To define the constraints we will use the $+$ operation on days of the weeks. To do that, we interpret every day as a number from Monday (1) to Friday (5). For example, $F = Th + 1$, and $Th = M + 3$ but $M \neq F + 1$. Formally, the network is defined as follows:

- Variables: $V = \{\text{a, b, c, d, e, f, g}\}$.
- Domains: For all $v \in V$: $D_v = \{\text{M, Tu, W, Th, F}\}$.

Example: Scheduling Tasks

- Constraints :
  - Since Lisa will need to go to the city to get party supplies and packing cases, she wants to do both on the same day. (a=g)
  - She cannot end her contracts without de-registering first, so she needs to end her contracts two days after de-registering. (e=d+2)
  - She needs to find a subtenant before the apartment gets filled with the party stuff, so she wants to do it two days before the party. (f=a-2)
  - She can only book her bus ticket after finding a subtenant. (b=f+1)
  - She cannot clean the apartment before the party, since the apartment will get dirty afterwards. So she wants to clean the apartment a day after the party. (c=a+1)
  - She will only de-register from the city once she knows when her bus is. (d=b+1)

Run the AC-3 algorithm. For each iteration of the while-loop, give the content of M at the start of the iteration, give the pair (u, v) removed from M, give the domain D_u of u after the call to Revise(γ, u, v), and give the pairs (w, u) added into M.
Note: Initialize M as a lexicographically ordered list (i.e., (a, b) would be before (a, c), both before (b, a) etc., if any of those exist). After initialization, use M as a FIFO queue, i.e., always remove the element at the front and add new elements to the back.



$M = \{(a,c),(a,f),(a,g),(b,d),(b,f),(c,a),(d,b),(d,e),(e,d),(f,a),(f,b),(g,a),(b,d),(f,b),(a,f),(c,a),(g,a)\}$

$D_v: \{1,2,3,4,5\}$

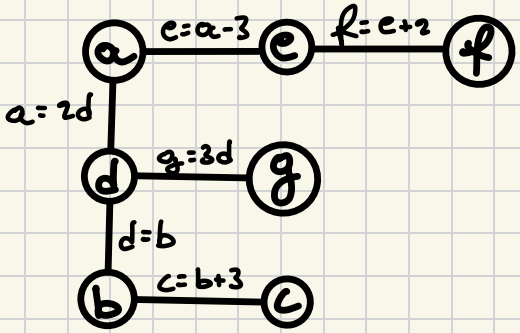
1) $a = c-1$ $\quad D_a = \{1,2,3,4\}$
2) $a = f+2$ $\quad D_a = \{3,4\}$
3) $a = g$ $\quad D_a = \{3,4\}$
4) $b = d-1$ $\quad D_b = \{1,2,3,4\}$
5) $b = f+1$ $\quad D_b = \{2,3,4\}$
6) $c = a+1$ $\quad D_c = \{4,5\}$
7) $d = b+1$ $\quad D_d = \{3,4,5\}$
8) $d = e-2$ $\quad D_d = \{3\}$
9) $e = d+2$ $\quad D_e = \{5\}$
10) $f = a-2$ $\quad D_f = \{1,2\}$
11) $f = b-1$ $\quad D_f = \{1,2\}$
12) $g = a$ $\quad D_g = \{3,4\}$
13) $b = d-1$ $\quad D_b = \{2\}$
14) $f = b-1$ $\quad D_b = \{1\}$
15) $a = f+2$ $\quad D_a = \{3\}$
16) $c = a+1$ $\quad D_c = \{4\}$
17) $g = a$ $\quad D = \{3\}$

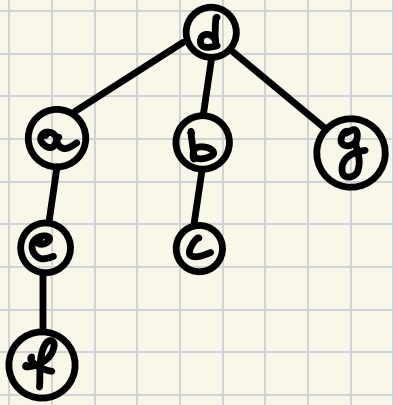
Example: Constraint Network

Consider the following constraint network: $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d, e, f, g\}$
- Domains: for all $u \in V, D_u = \{1, 2, 3, 4, 5, 6\}$
- Constraints: $a = 2d,\ g = 3d,\ d = b,\ e = a - 3,\ c = b + 3,\ f = e + 2$

Run the AcyclicCG algorithm from the lecture slides Draw the constraint graph of γ. Pick d as the root and draw the directed tree obtained by performing step 1. Give the resulting variable ordering obtained by performing step 2. If the ordering of the variables is not unique, break ties using alphabetical order. List the calls to $Revise(\gamma, v_{parent(i)}, v_i)$ in the order executed by running step 3, and, for each of them, give the resulting domain of $v_{parent(i)}$. For each recursive call to BacktrackingWithInference in step 4, give the domain D'_{v_i} of the selected variable v_i after Forward Checking and the value $d \in D'_{v_i}$ assigned to v_i.





ORDER: d, a, b, c, e, f, g

Revise(x, P(V_i), V_i) $D_V = \{1, 2, 3, 4, 5, 6\}$

i=7	Revise(d, g)	$D_d = \{1, 2\}$
i=6	Revise(e, f)	$D_e = \{1, 2, 3, 4\}$
i=5	Revise(a, e)	$D_a = \{4, 5, 6\}$
i=4	Revise(b, c)	$D_b = \{1, 2, 3\}$
i=3	Revise(d, b)	$D_d = \{1, 2\}$
i=2	Revise(d, a)	$D_d = \{2\}$

a=4
b=2
c=5
e=1
f=3
g=6

TUTTI VALORI PRESENTI NEI DOMINI