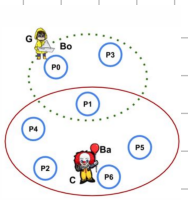
A clown C and Georgie G are tired of playing with their toys and they want to exchange them. G wants to play with the balloon Ba while, C wants to play with the boat Bo. Unfortunately, they are both very shy and they never want to be at the same place P_i. Hence, they have to find a common place (for example P_3) where to drop and collect objects (they can drop an object in a place and move into another place). G and C cannot hold two objects at the same time. The environment is depicted in the figure, G can only move in places within the dotted line; while C can only move within the continuous line. Places within the same set are all connected (i.e. $\{P_0, P_1, P_3\}$ for the dotted set and $\{P_1, P_2, P_4, P_5, P_6\}$ for the other). The figure shows the initial state where C holds Bo and he is at P_0, and C holds Ba and he is at P_5. The goal state is represented by G at P_3 holding Ba and C at P_5 holding Bo.



(a) Define the problem and the domain file in PDDL

(b) Show one possible sequence of actions to a goal state including all the states in the sequence

(c) Draw the first 3 steps of the tree generated by forward search assuming a perfect heuristic (a heuristic choosing the move in the plan given above). Show all the actions applicable at each of the traversed states, and the state reached.

```
(:requirments :adl)
(:define (domain SCAMBIO)

(:types     item
            boy-item
            toy-item
            cell )

(:predicates    (at ?I-item ?C-cell)
                (picked ?T-toy ?B-boy)
                (valid ?C-cell ?B-boy)
)

(:action go
      :parameters (?B-boy ?C-cell C1?-cell)
      :preconditions (and (valid ?c ?B) (Forall (?B1-boy) (not at ?B1 ?C)) (at ?B ?C1))
      :effects (and (not at ?B ?C1) (at ?B ?C))
)

(:action pick
   :parameters (?B-boy ?T-toy ?C-cell)
   :preconditions (and (at ?B ?C) (at ?B ?T) (not exists (?T1-toy) (picked ?T1 ?B)))
   :effects (and (not at ?T ?C) (picked ?T ?B))
)

(:action put
   :parameters (?B-boy ?T-toy ?C-cell)
   :preconditions (and (at ?B ?C) (picked ?T ?B))
   :effects (and (not picked ?T ?B) (at ?T ?C))
)

Now the problem file

(:requirments :adl)
(:define (problem 1T))
(:domain SCAMBIO)

(:objects   G C - boy
            BAL BOAT - toy
            P0 P1 P2 P3 P4 P5 P6 - cell
)

(:init  (valid P0 G) (valid P1 G) (valid P3 G) (valid P1 C) (valid P2 C) (valid P4 C) (valid P5 C) (valid P6 C)
        (at C P6) (at G P0) (picked BAL C) (picked BOAT G)
)

(:goal  (at G P3) (at C P5) (picked BAL G) (picked BOAT C))
```

one possible sequence is

(go G P1 P0) (put G BOAT P1) (go G P0 P1) (go C P1 P6) (put C BAL P1) (pick C BOAT P1)
(go C P5 P1) (go G P1 P6) (pick G BAL P1) (go G P3 P1)

the state 1 is

⟨G in P0, C in P6, BOAT in G, BAL in C⟩

we can apply:

leave BOAT in P0, leave BAL in P6, G goes to {P1, P3}, C goes to {P1, P2, P4, P5}

we go with G in P1

⟨G in P1, C in P6, BOAT in G, BAL in C⟩

we can apply:

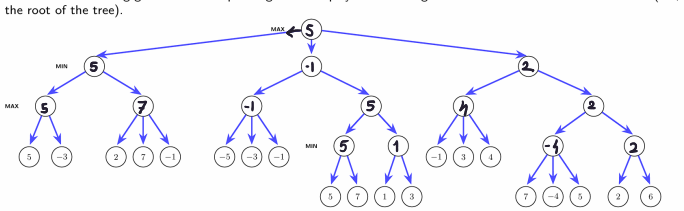
leave BOAT in P1, leave BAL in P6, G goes to {P0, P3}, C goes to {P1, P2, P4, P5}

⟨G in P1, C in P6, BOAT in P1, BAL in C⟩

pick BOAT in P1, leave BAL in P6, G goes to {P0, P3}, C goes to {P1, P2, P4, P5}

⟨G in P0, C in P6, BOAT in P1, BAL in C⟩

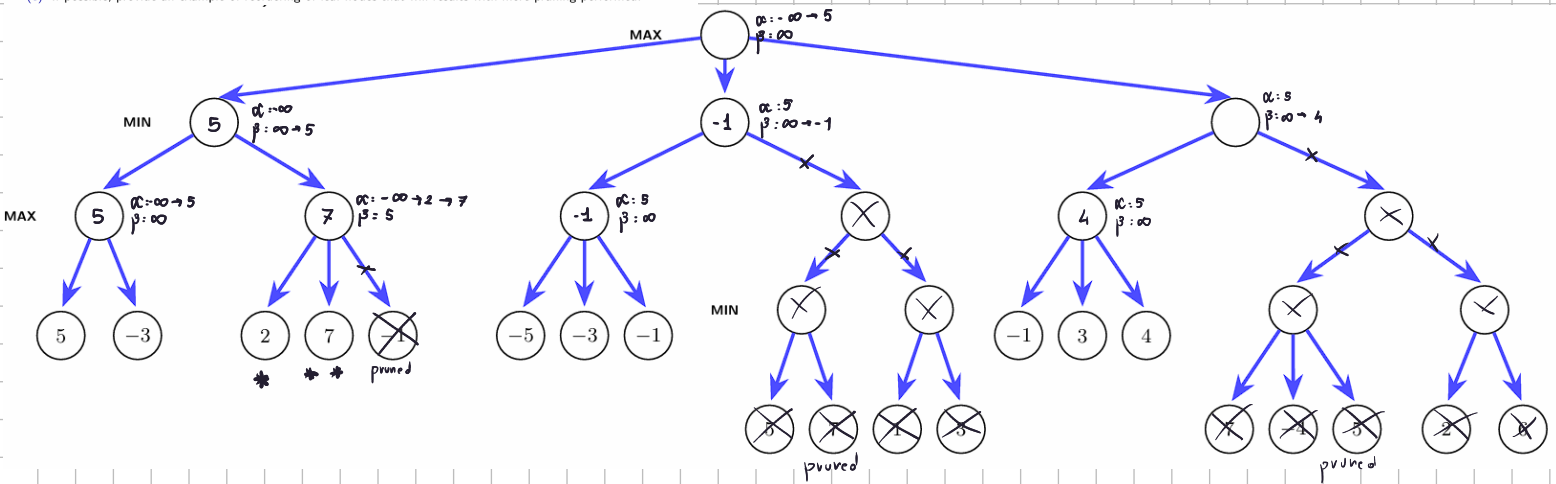
Consider the following game tree corresponding to a two-player zero-sum game. Max is to start in the initial state (i.e., the root of the tree).



(a) Perform Minimax search on the tree, i.e., annotate all internal nodes with the correct Minimax value. Which move does Max choose? left branch

(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.



10 leaf are pruned. if we swap the leaf 3 and 4 (marked * and **) we can prune also the leaf *.

(a) Transform the following formula in CNF specifying all the steps:

$$\neg\Big(\big(((A \wedge B) \to C) \wedge (A \vee B \vee C)\big) \to \big((A \leftrightarrow B) \to C\big)\Big)$$

(b) Use resolution to determine if this formula is inconsistent:

$$(\neg A \vee \neg D) \wedge (\neg A \vee D) \wedge (B \vee C \vee \neg D) \wedge (A \vee B) \wedge (\neg C \vee \neg D) \wedge (A \vee \neg B \vee \neg D) \wedge (A \vee \neg B \vee D)$$

(c) Perform DPLL on the following formula to look for a satisfiable assignment. Assume that DPLL selects variables in alphabetical order (i.e., $A, B, \dots$), and that the splitting rule first attempts the value False (F):

$$\{\{\neg B\}, \{A, B, C, D\}, \{\neg C, \neg D\}, \{C, \neg D\}, \{A, \neg B, D\}, \{A, \neg C\}\}$$

a)

$$\neg\Big(\big(((A \wedge B) \to C) \wedge (A \vee B \vee C)\big) \to \big((A \leftrightarrow B) \to C\big)\Big)$$

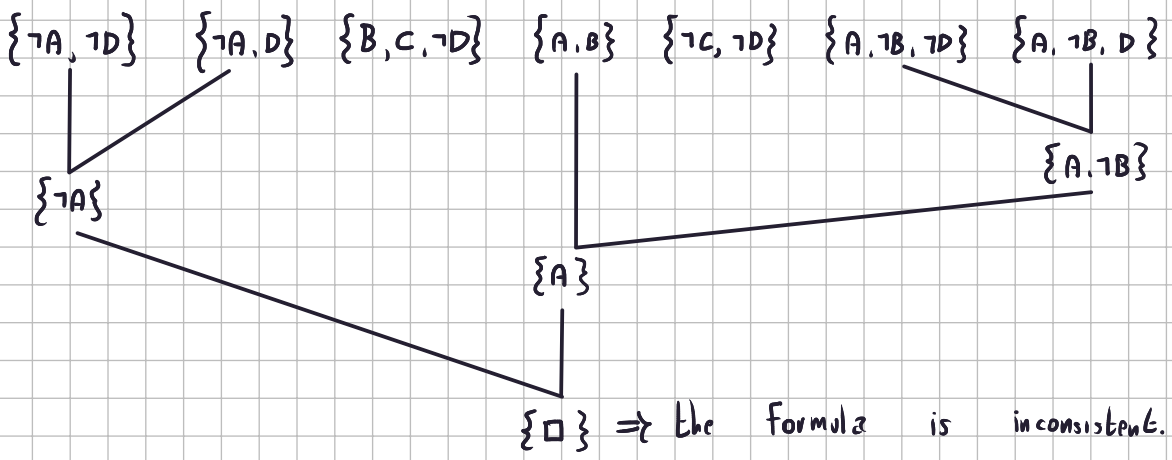
$$\neg\Big(\neg\big(((\neg A \vee \neg B \vee C) \wedge (A \vee B \vee C)) \vee ((\neg A \wedge B) \vee (A \wedge \neg B) \vee C)\big)\Big)$$

$$((\neg A \vee \neg B \vee C) \wedge (A \vee B \vee C)) \wedge \big(\neg(\neg A \wedge B) \wedge \neg(A \wedge \neg B) \wedge \neg C\big)$$

$$((\neg A \vee \neg B \vee C) \wedge (A \vee B \vee C)) \wedge \big((A \vee \neg B) \wedge (\neg A \vee B) \wedge \neg C\big) =$$

$$\Delta = \{\{\neg A, \neg B, C\}, \{A, B, C\}, \{A, \neg B\}, \{\neg A, B\}, \{\neg C\}\}$$

b)



$\{\square\} \Rightarrow$ the Formula is inconsistent.

c)

$\{\neg B\}$ $\{A, B, C, D\}$ $\{\neg C, \neg D\}$ $\{C, \neg D\}$ $\{A, \neg B, D\}$ $\{A, \neg C\}$

UP RULE: $B = 0$

// $\{A, C, D\}$ $\{\neg C, \neg D\}$ $\{C, \neg D\}$ // $\{A, \neg C\}$

CHOICE $A = 0$

$\{C, D\}$ $\{\neg C, \neg D\}$ $\{C, \neg D\}$ $\{\neg C\}$

UP : $C = 0$

$\{D\}$ // $\{\neg D\}$

CONFLICT. GO BACK TO SET $A = 1$

// $\{\neg C, \neg D\}$ $\{C, \neg D\}$ // //

CHOICE $C = 0$

$\{\neg D\}$

UP $D = 0$

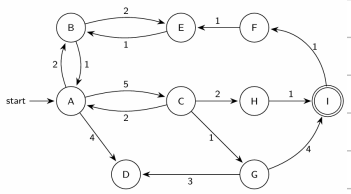
$\Rightarrow$ Final assignment $A: 1$ $B: 0$ $C: 0$ $D: 0$

Consider the state space in the figure below, where A is the initial state and I the goal state. The transitions are annotated by their costs.

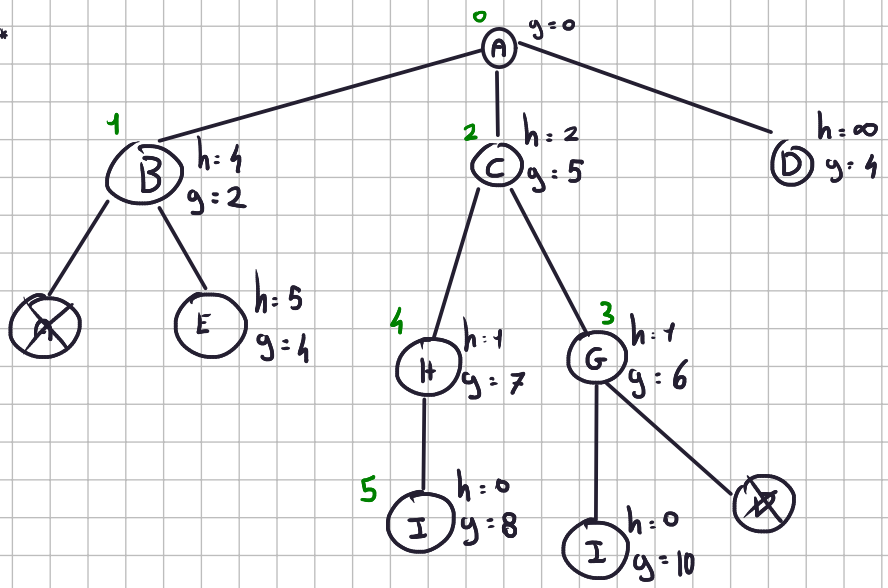
(a) Run the A^* search algorithm on this problem. As a heuristic estimate for a state s, use the minimal number of edges that are needed to reach a goal state from s (or ∞ if s is not solvable, e.g., $h(C) = 2$). Draw the search graph and annotate each node with the g and h value as well as the order 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 $h : \{A, B, \ldots, I\} \to \mathbb{N}_0^+ \cup \{\infty\}$ for the state space depicted that leads hill-climbing into a local minimum, and explain what happens. If no, please explain why.



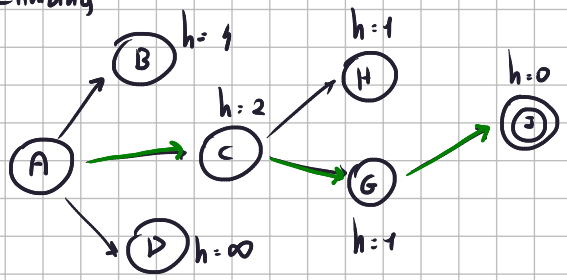
A*



path: A → C → H → I

since the heuristic is consistent and admissible, A* finds the optimal path for this problem.

Hill Climbing



the solution is not optimal since the hill climbing algorithm consider only the heuristic function and not the cost of the actual path.

Consider the heuristic $h = (x) = 100 \ \forall x$ except A, C, I Where $h(A) = h(B) = h(I) = 0$