CS2109s - Tutorial 3

Eric Han (TG12-TG15)

Feb 15, 2024

Annoucements

Important admin

1. TG15, PS4 - PS8 will be marked by **Guo Mingqi** due to my high teaching workload [By Teaching Committee/Rizki].

Problem Set 2

1. Q1.2 - A*Star algorithm
    1. Check if node is visited
    2. PQ not updated correctly when encountering a path to a child node with a lower cost.
2. Q2.2 - TSP Goal state cannot be known explictly - state explicitly that goal state cannot be known, if not i do not know if you are trying to define it or not.
3. Q2.6 - Abuse of `random.sample` to create the inital state, it will give the desired effect but this is abuse!
4. Q1.3 - Most explanations are sloppy but since its 1 marks as long as you explain in line of the max rows/cols, I will award you; Learn to explain properly.

Task 1.3 - Why heuristic is consistent and admissible [Wenzhong 2324S1/TG4].

Let the shape of the 2D Rubik cube be (R, C). Let x be the number of misplaced piece in state n, and $M = \max(R, C)$.

My heuristic is: $h(n) = \frac{x}{M}$

Admissibility (but not necessary) Heuristic is admissible since in the original puzzle, each move will correct at most M number of pieces, and the heuristic is saying each move is the best move possible (i.e. correct M number of pieces), hence this heuristic can never be larger than the actual number of moves. $=> h(n) \leq h^*(n)$

Consistency Let n be a state in the puzzle, and n' be the direct successor of n (i.e. the children of n) Denote G to be the goal state.

1. From my heuristic $h(n) = \frac{x}{M}$
2. Since every move can at most put M pieces into the right place, n' can have at least $x - M$ misplaced pieces
3. Thus, $h(n') \geq \frac{x-M}{M} = \frac{x}{M} - 1$
4. Also, for this puzzle $c(n, a, n') = 1$ given that a is a valid move that move n to n', as each move of the Rubik cube have uniform cost.
    - $c(n, a, n') + h(n') \geq 1 + \frac{x}{M} - 1 = \frac{x}{M} = h(n)$
    - $h(n) \leq c(n, a, n') + h(n')$
5. Therefore by definition, $h(n)$ is consistent.
6. Since consistent imples admissible and $h(g) = 0$, so $h(n)$ is admissible.

Question 1

Tic-Tac-Toe - Use the minimax to determine the first move of the player.

$$Eval(n) = P(n) - O(n), \quad \text{where } P(n), O(n) \text{ are the no. of winning lines}$$

Recap

1. What is the `MINIMAX` algorithm? Why is it used?
2. What are the ingredients needed to setup a minimax problem?
3. What is the impact of choosing min/max in our computation?
4. [@] When was `MINIMAX` famously used in AI?

. . .

- Actors: Min/Max, Leaf Cost
- IBM Deep Blue versus Garry Kasparov in Chess.

Question 1a

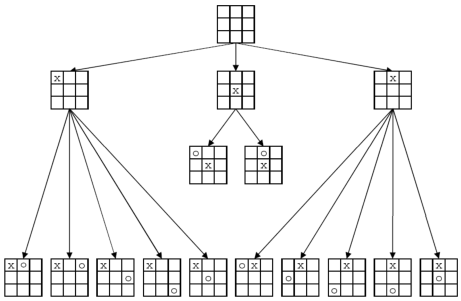

Figure 1: First move 2-ply deep search space

Answer 1a

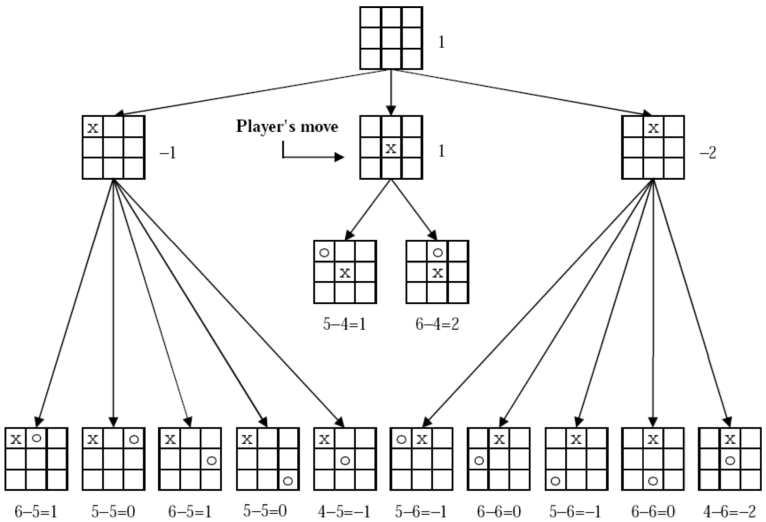

Figure 2: First move 2-ply deep search space

Question 1b

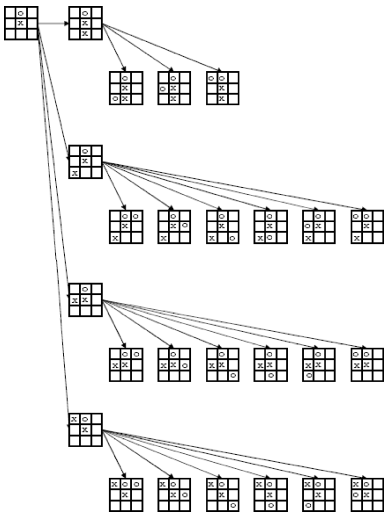

Figure 3: Second move 2-ply deep search space solution

Answer 1b

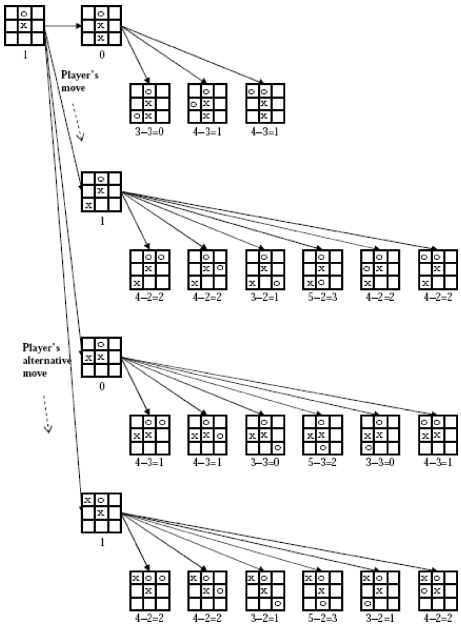

Figure 4: Second move 2-ply deep search space solution

Question 2 [G]

Run through the α-β:

a. Right to Left
b. Left to Right

Then determine if the effectiveness of pruning depends on iteration order.

Recap

1. What does α-β do?
2. What kind of efficiency do you gain?
3. What is deep cutoff?

. . .

Save on static evaluation and move generation.

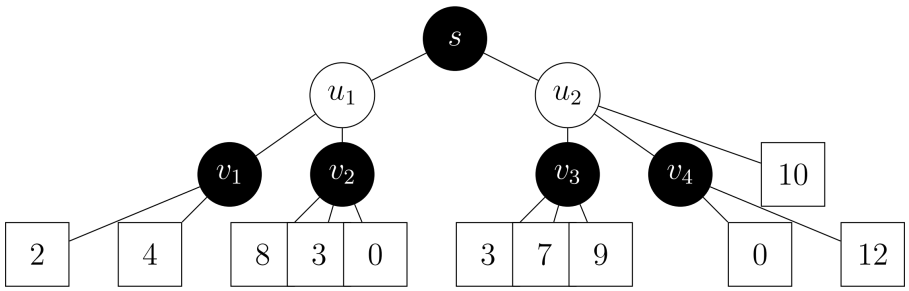

Figure 5: Alpha-Beta Tree

Answer

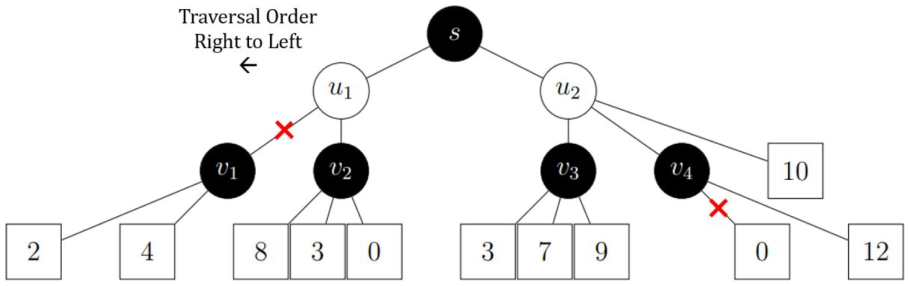

Figure 6: Right to left

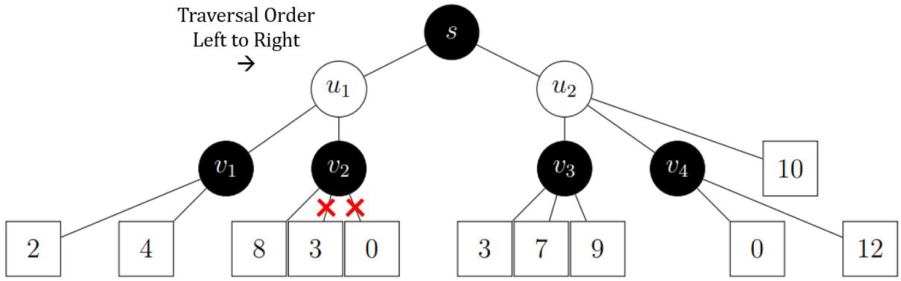

Figure 7: Left to right

Question 3

Nonogram, aka Paint by Numbers, is a puzzle where cells are colored or left blank according to the numbers at the side of the grid.

Recap

1. What are the ingredients needed for informed search?
2. What are the ingredients needed for local search?
3. What are the objectives for informed/local search?

. . .

Un/Informed Search (Path): State space, Initial, Final, Action, Transition

- Uninformed: BFS, UCS, DFS
- Informed: GBFS, A*

Local Search (Goal): Inital state, Transition, Heuristic/Stopping criteria

- Hill Climbing, Sim. Annealing, Beam, Genetic…

			3	1	1	4	4
1	1	1					
	1	2					
	2	2					
		2					
		1					

Figure 8: Inital

			3	1	1	4	4
1	1	1	■		■		■
	1	2	■			■	■
	2	2	■	■		■	■
		2				■	■
		1				■	

Figure 9: Solved

Adversarial Search: Actors, Actions, Leaf Costs

- Minimax, Alpha-Beta

Question 3a [G]

Having learnt both informed search and local search, you think that local search is more suitable for this problem. Give 2 possible reasons why informed search might be a bad idea.

. . .

Answer 3a

- We are only interested in the final solution.
- Search space is large $O(2^{n \times n})$ for a $n \times n$ grid.
- May not be solvable? In that case we can get a config that minimize violations.

Question 3b / 3c / 3d / 3e [G]

Find a formulation for **Local Search**.

. . .

Answer 3b / 3c / 3d / 3e

$n \times n$ boolean matrix, where each element is either true (if the corresponding cell is colored) or false (if the corresponding cell is not colored).

- **Inital state** is an $n \times n$ boolean matrix with every row having random permutations of boolean vector satisfying row constraints, while the rest of the entries are set to false.
- **Transition**: we can pick a random row and generate the list of neighbours with the corresponding row permuted satisfying row constraints.
- **Heuristic/Stopping criteria**: number of instances where the constraints on the column configurations are violated.

Question 3f [G]

Local search is susceptible to local minimas. Describe how you can modify your solution to combat this.

. . .

Answer 3f

- Introduce random restarts by repeating local search from a random initial state
- Simulated annealing search to accept a possibly `bad` state with a probability that decays over time
- beam search to perform k hill-climbing searches in parallel.

Question 4 [@]

In order for node B to NOT be pruned, what values can node A take on?

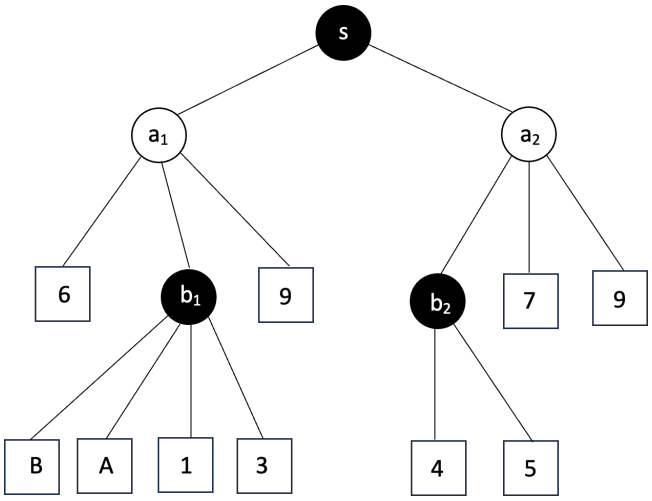

Figure 10: Find A so the B is not pruned.

```
< S -inf inf
        < a2 -inf inf
        > a2 -inf 9
        < a2 -inf 9
        > a2 -inf 7
        < a2 -inf 7
                < b2 -inf 7
                > b2 5 7
                < b2 5 7
                > b2 5 7
        > a2 -inf 5
> S 5 inf
```

```
< S 5 inf
        < a1 5 inf
        > a1 5 9
        < a1 5 9
                < b1 5 9
                > b1 5 9
                < b1 5 9
                > b1 5 9
                < b1 5 9
                > b1 Pruned val >= beta: 9 >= 9
        > a1 5 9
```

```
        < a1 5 9
        > a1 5 6
> S 6 inf
```

Pruned when $A \geq 9$, Not pruned when $A \leq 8$

Bonus Qn

To help you further your understanding, not compulsory; Work for Snack/EXP!

Tasks

1. Trace Manually/Use code Figure 11 to see the full capability.
    1. Some code implemented in https://github.com/eric-vader/CS2109s-2324s2-bonus
2. How can we benefit from α-β's efficiency?

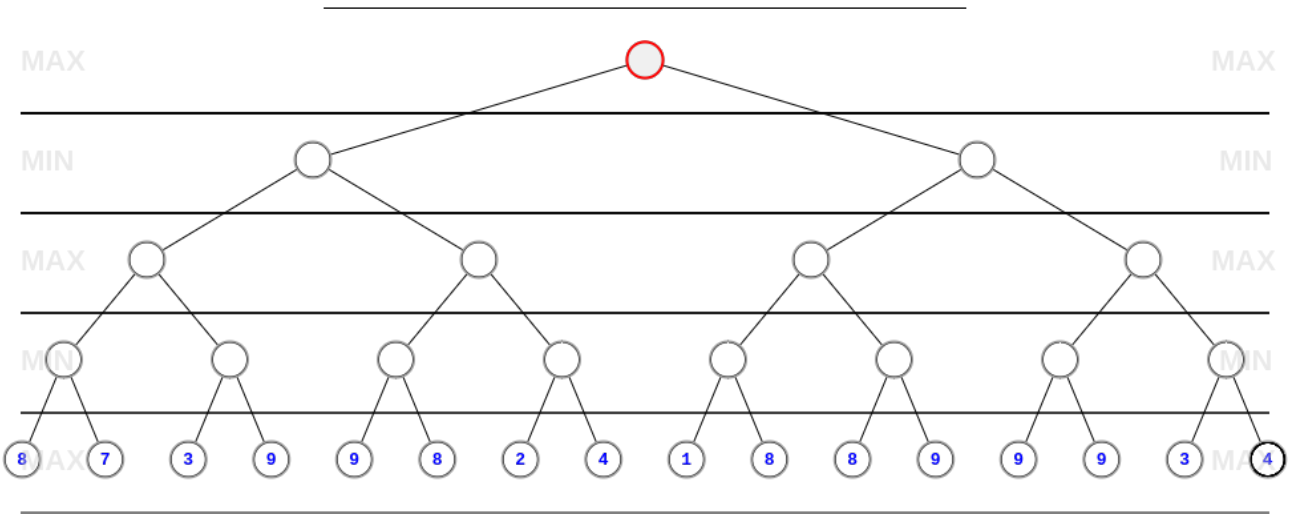

Figure 11: Alpha-Beta Example (Credit MIT)

Study More

1. MIT Lecture - https://youtu.be/STjW3eH0Cik?si=YcnrXUJko5jjLzB0
2. IBM Deep Blue - https://www.sciencedirect.com/science/article/pii/S0004370201001291
3. Game Theory Concepts Within AlphaGo - https://towardsdatascience.com/game-theory-concepts-within-alphago-2443bbca36e0
4. What Game Theory Reveals About Life, The Universe, and Everything - https://youtu.be/mScpHTIi-kM?si=CLagrjz3WVi-EkXG

Buddy Attendance Taking

1. [@] and Bonus declaration is to be done here; You should show bonus to Eric.
2. Attempted tutorial should come with proof (sketches, workings etc…)
3. Random checks will be conducted - `python ../checks.py T13`

Figure 12: Buddy Attendance: https://forms.gle/jsGfFyfo9PTgWxib6