CS2109s - Tutorial 3

Eric Han

Sep 13, 2023

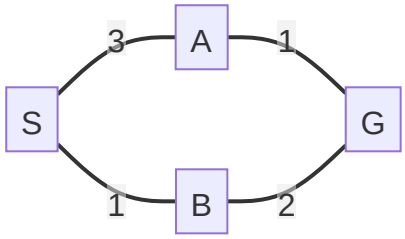
Annoucements

Important admin

No admin annoucements today.

Tidbits from tutorials

1. GBFS with a consistent heuristic is it optimal? - In this case h^* used, but still not optimal `SAG` instead of `SBG`.



$h(S) = 3, h(A) = 1, h(B) = 2, h(G) = 0$

```
S(3)
A(1) B(2)
G(0) B(2)
SAG
```

Problem Set 2

1. Question 1.2 - 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. Question 2.2 - TSP Goal node cannot be known explictly

Task 1.3 - Why heuristic is consistent and admissible [Liang Wenzhong TG4].

Let the shape of the 2D Rubik cube be `(R, C)`. Denote `M = max(R, C)`.

My heuristic is: `h(n) = sum_of_misplaced_pieces(n) / M`

Admissibility 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) <= 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.

Let `x` be the number of misplaced piece in state `n`, and `M` as described above. Then we deduce that `h(n) = x / M`

Notice that `n'` can have at least `x - M` misplaced pieces. Since every move can at most put `M` pieces into the right place.

Thus, `h(n') >= (x - M) / M = x / M - 1`

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') >= 1 + (x / M - 1) = x / M`

=> `h(n) <= c(n, a, n') + h(n')`

Thus, by the definition of consistent, it is shown that `h(n)` is consistent, and hence also 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

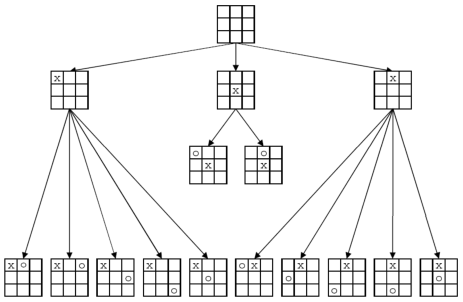


Figure 1: First move 2-ply deep search space

Answer 1a

Question 1b

Answer 1b

Question 2 [G]

Run through the α-β:

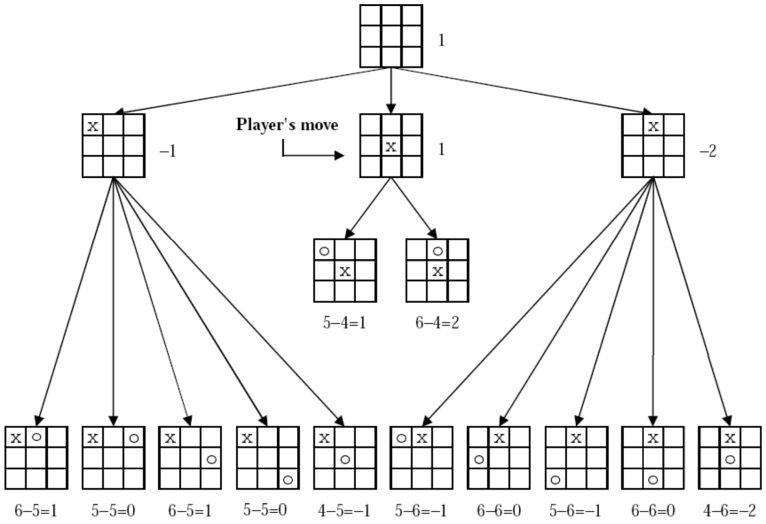


Figure 2: First move 2-ply deep search space

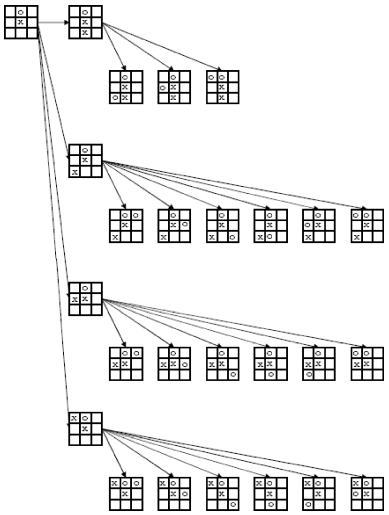


Figure 3: Second move 2-ply deep search space solution

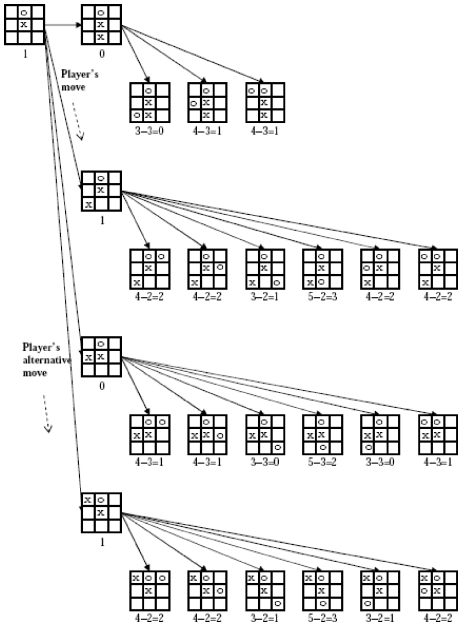


Figure 4: Second move 2-ply deep search space solution

a. Right to Left
b. Left to Right

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

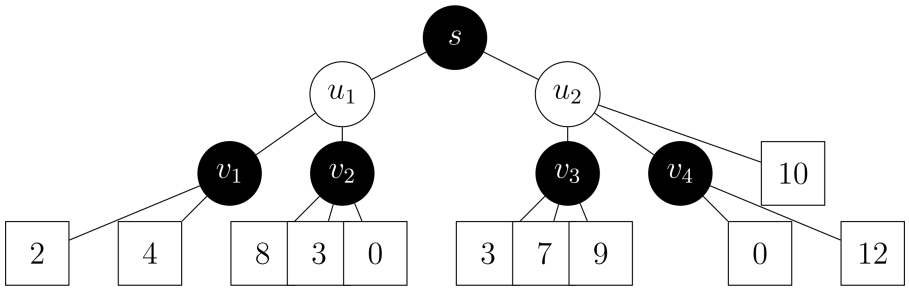


Figure 5: Alpha-Beta Tree

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.

Answer

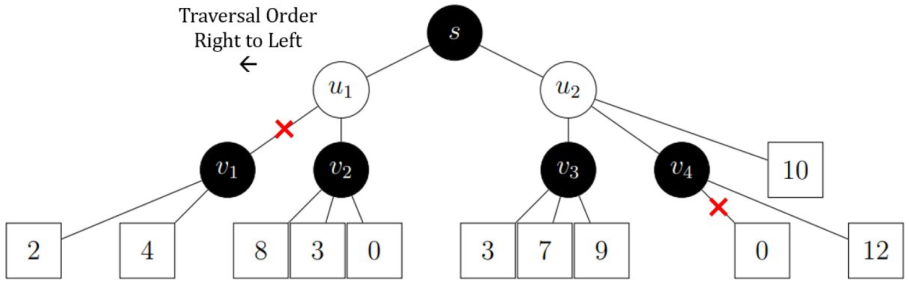


Figure 6: Right to left

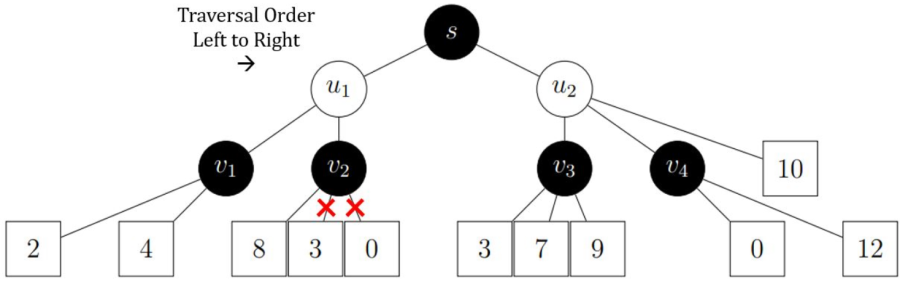


Figure 7: Left to right

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.

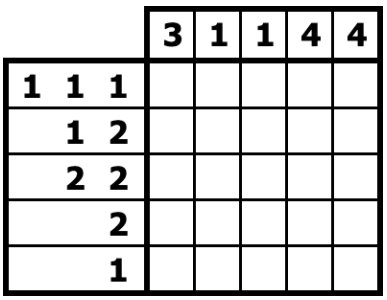


Figure 8: Inital

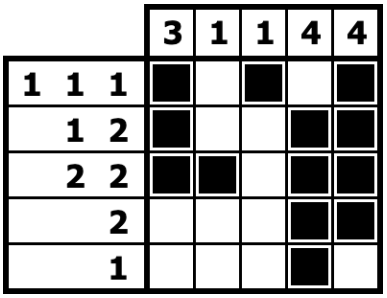


Figure 9: Solved

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.

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?

. . .

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

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

. . .

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 [G]

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

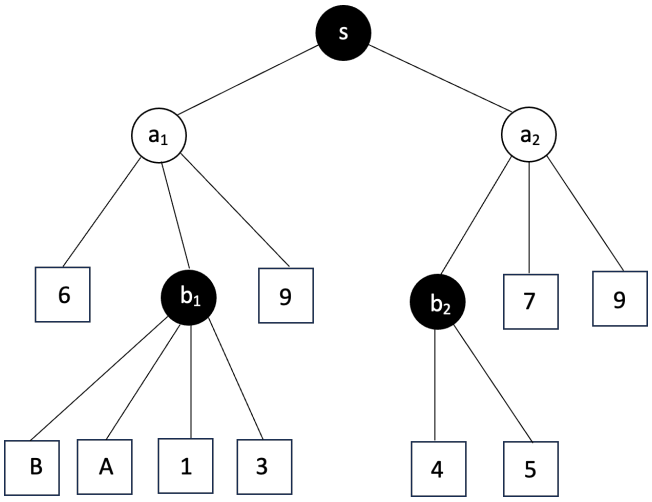


Figure 10: Find A so the B is not pruned.

```
< 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
```

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-2324s1-bonus
2. How can we benefit from α-β's efficiency?

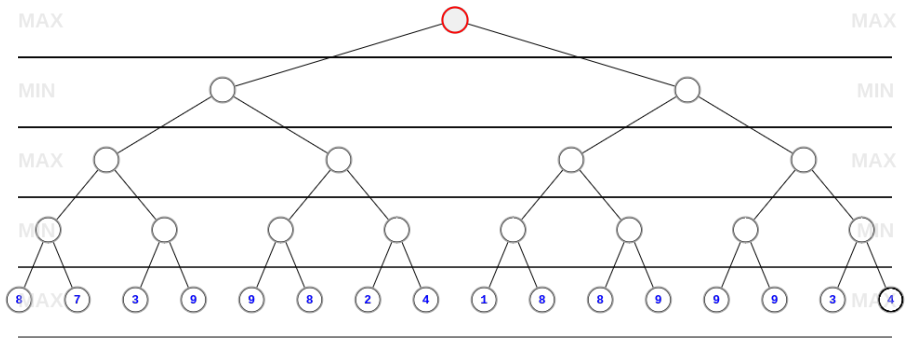


Figure 11: Alpha-Beta Example (Credit MIT)

Buddy Attendance Taking

Take Attendance for your buddy: https://forms.gle/Ckkq639TNwWEx3NT6

1. Random checks will be conducted - `python ../checks.py TG0`

Figure 12: Buddy Attendance