ISL logo coloring problem:

$$D— \quad —S$$
$$I— \quad —L$$

→
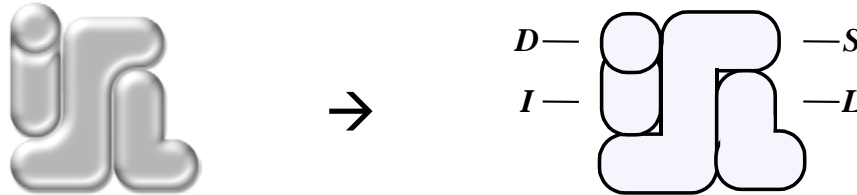
Depth-First Search (no forward checking):
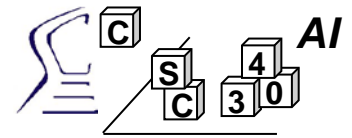
search space:

```
        D
        I
        L
        S    R G B
```

(tree with uniform depth of 4, branching factor 3)

R G B

. . .

uniform depth of 4

branching factor = 3

number of nodes: exactly

$1 + 3 + 3^2 + 3^3 + 3^4 = \underline{121}$
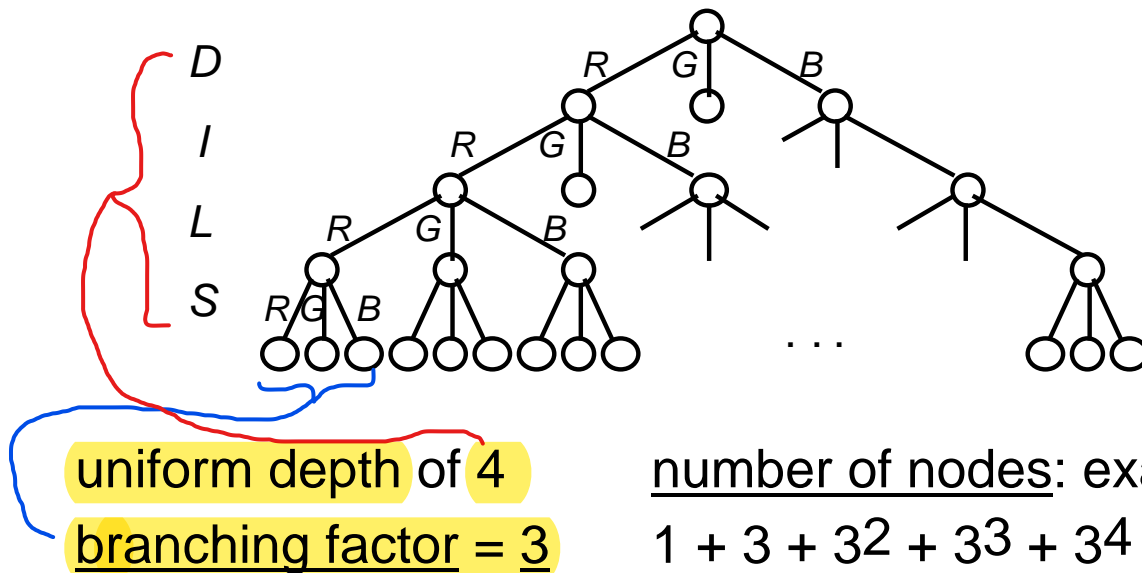
DFS a good choice?

yes → depth-limited search space, all solutions at maximum depth, many possible solutions

no → need to generate all solutions (not just one), blind search generates invalid colorings, backtracks unnecessarily
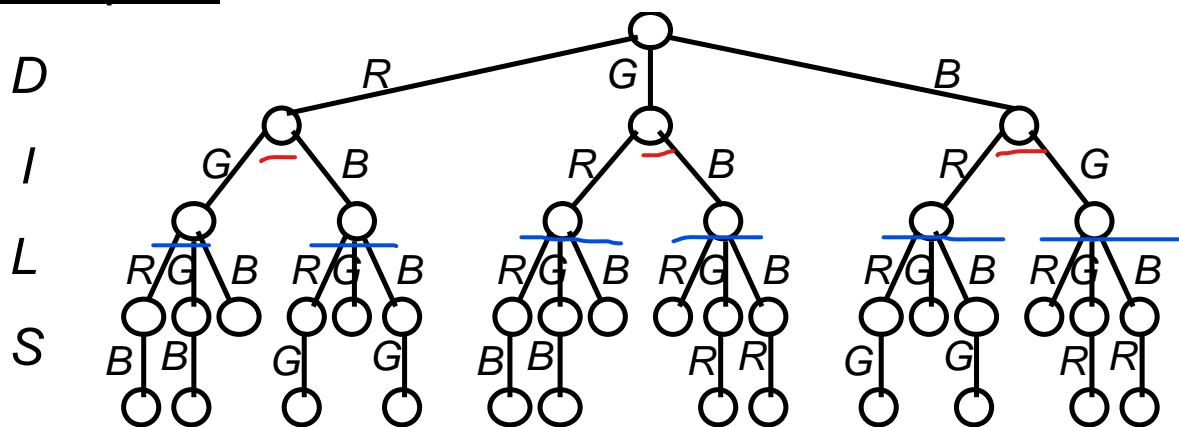
additional slide -- for reference only

DFS with *forward checking, arbitrary ordering:*

logo colouring = Constraint Satisfaction Problem

forward checking: *take constraints into account to significantly prune the search space*

search space:

number of nodes: 1 + 3 + 3∗2 + 6∗3 + 6∗2 = 40

avg branching factor: (40 − 1) / (1 + 3 + 6 + 6∗2) = 1.77

non root nodes / non leaf nodes

efficiency: significant improvement i.e.,
        search space reduced from 121 to 40 nodes,
        branching factor decreased from 3 to 1.77,
        no invalid colorings, little backtracking

however elements are assigned a color in arbitrary (alphabetical) order → *not* optimal

did not start with S involving the largest number of constraints
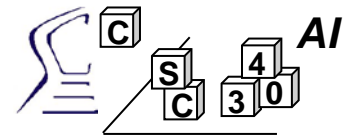
General CSP heuristics, and their usefulness:

Most Constraining Variable (Degree)

select among yet unassigned variables the one involved in the largest number of constraints

i.e., logo element with the largest nb of neighbors

→ useful to optimize the order of variable assigments

Most Constrained Variable(Minimum Remaining Value)
select variable with the fewest possible values

ie. logo element with the fewest colors available

-> useful to complement the above in case of a tie
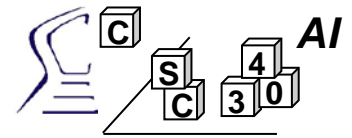
Least Constraining Value

select a value that leaves the largest choice of values for other constraint-related variables

i.e., color that less constrains other logo elements

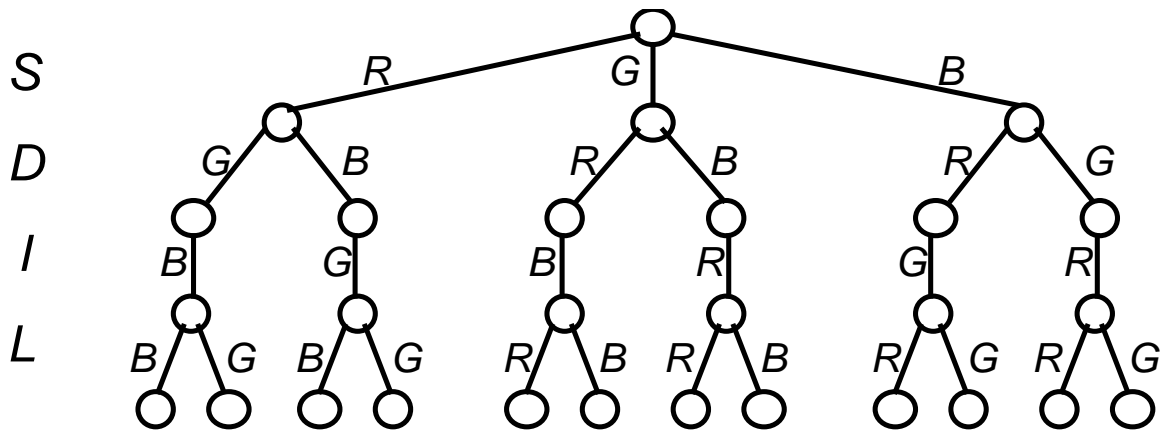→ useful to prevent deadlocks, reduce backtracking

best heuristics: *Most Constraining Variable*

DFS with forward checking, heuristic ordering:

most constraining logo element: S with 3 neighbors,
followed by D and I with 2 neighbors each,
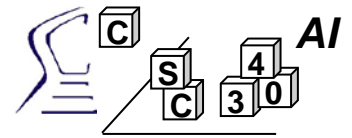then L with only 1 neighbor

search space:

S D I L

number of nodes: $1 + 3 + 3*2 + 6*1 + 6*2 = \underline{28}$
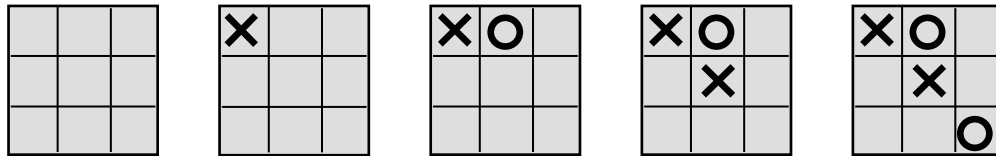
avg branching factor: $(28 - 1) / (1 + 3 + 6 + 6*1) = \underline{1.69}$

efficiency: further good improvement i.e.,
search space reduced from 40 to 28 nodes,
branching factor decreased from 1.77 to
1.69, no invalid colorings, no backtracking
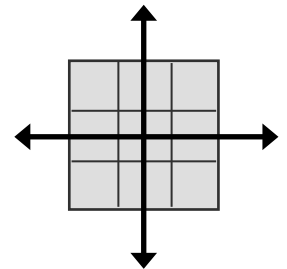
elements are assigned a color in *optimal* order
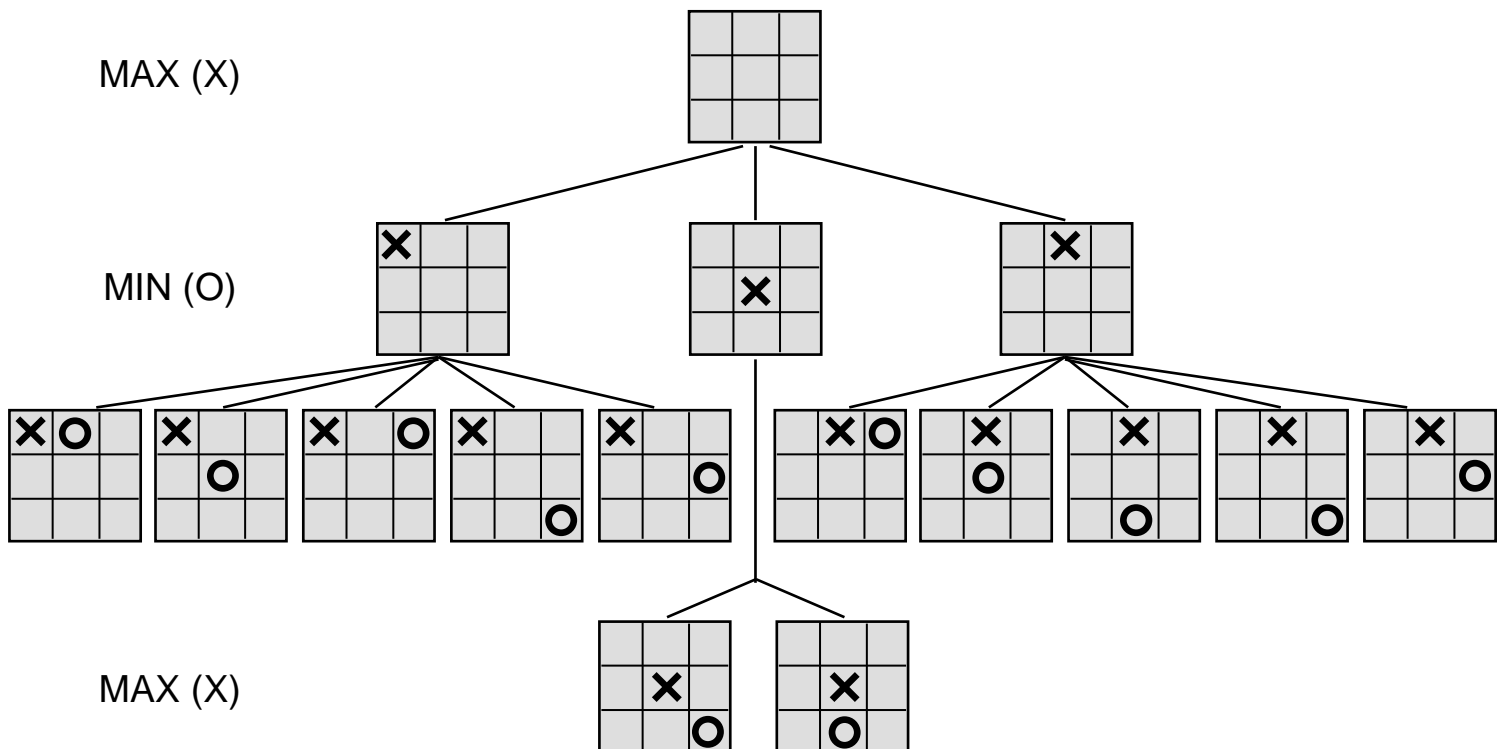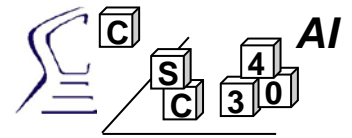
Depth-2 game tree for Tic-Tac-Toe:

horizontal and vertical symmetry
> thus only 3 moves at depth 1 (vs. 9)
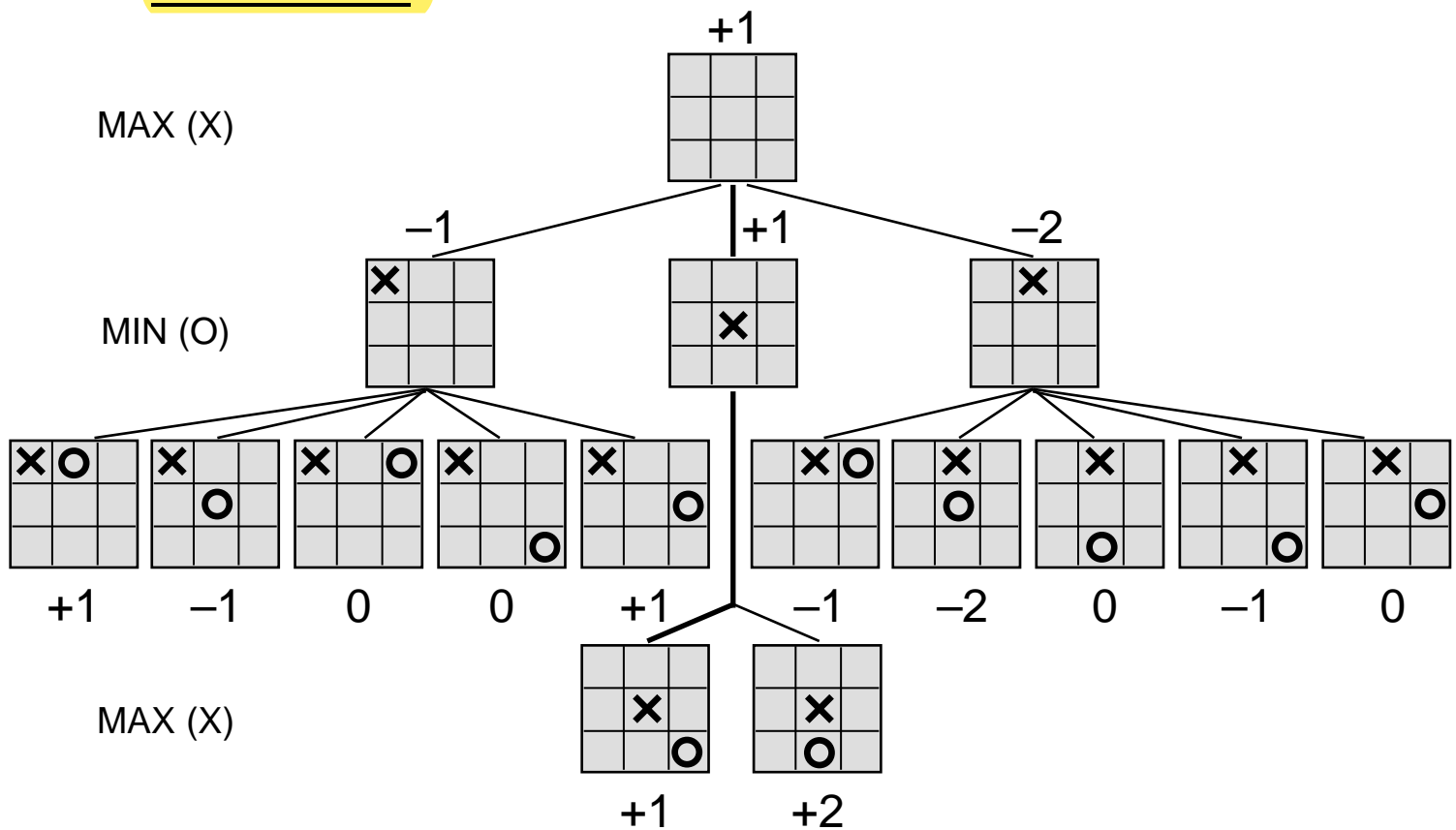> and 12 moves at depth 2 (vs. 72)

search tree:

MAX (X)

MIN (O)

MAX (X)

Heuristics for Tic-Tac-Toe:

value of non–terminal nodes: $3X_2 + X_1 - 3O_2 - O_1$

at <u>depth 2</u>: $X_2 = O_2 = 0$ → $h = X_1 - O_1$

<u>search tree</u>:

<u>best move</u> for MAX: *play in the centre!*

fair game? no, whoever plays first has a strong advantage