AlphaBeta wants to know is next child better than best so far?

possible answers: 1) no value $\le \alpha$

(for max node) 2) yes, and the value is exactly — $\alpha <$ value $< \beta$

3) yes, and so good you don't even want to know value $\ge \beta$

Scout wants to know is next child better than best so far?

possible answers: 1) no value $\le \alpha$

(for max node) 2) yes value $> \alpha$ ← if yes, need to distinguish between

Scout $(p, \alpha, \beta, depth, h)$

if p is terminal then return value(p)

if depth $= 0$ then return heuristic(p)

if p is a max position

best $\leftarrow -\infty$

for each reachable position p' in order of ↓ goodness and while $\alpha < \beta$

if p' is first pos

score ← Scout$(p', \alpha, \beta, depth-1, h)$

else

null window (assuming integers)

score ← AB$(p', \alpha, \alpha+1, depth-1, h)$

already passing null window, so no reason for Scout

if $\alpha <$ score $< \beta$

if score $\ge \beta$ then have lower bound on $p' \ge \beta$ so value of $p \ge \beta$ cut off

score ← Scout$(p', score, \beta, depth-1, h)$

already know value(pos) $\ge$ score

best ← max(best, score)

α ← max(best, α)

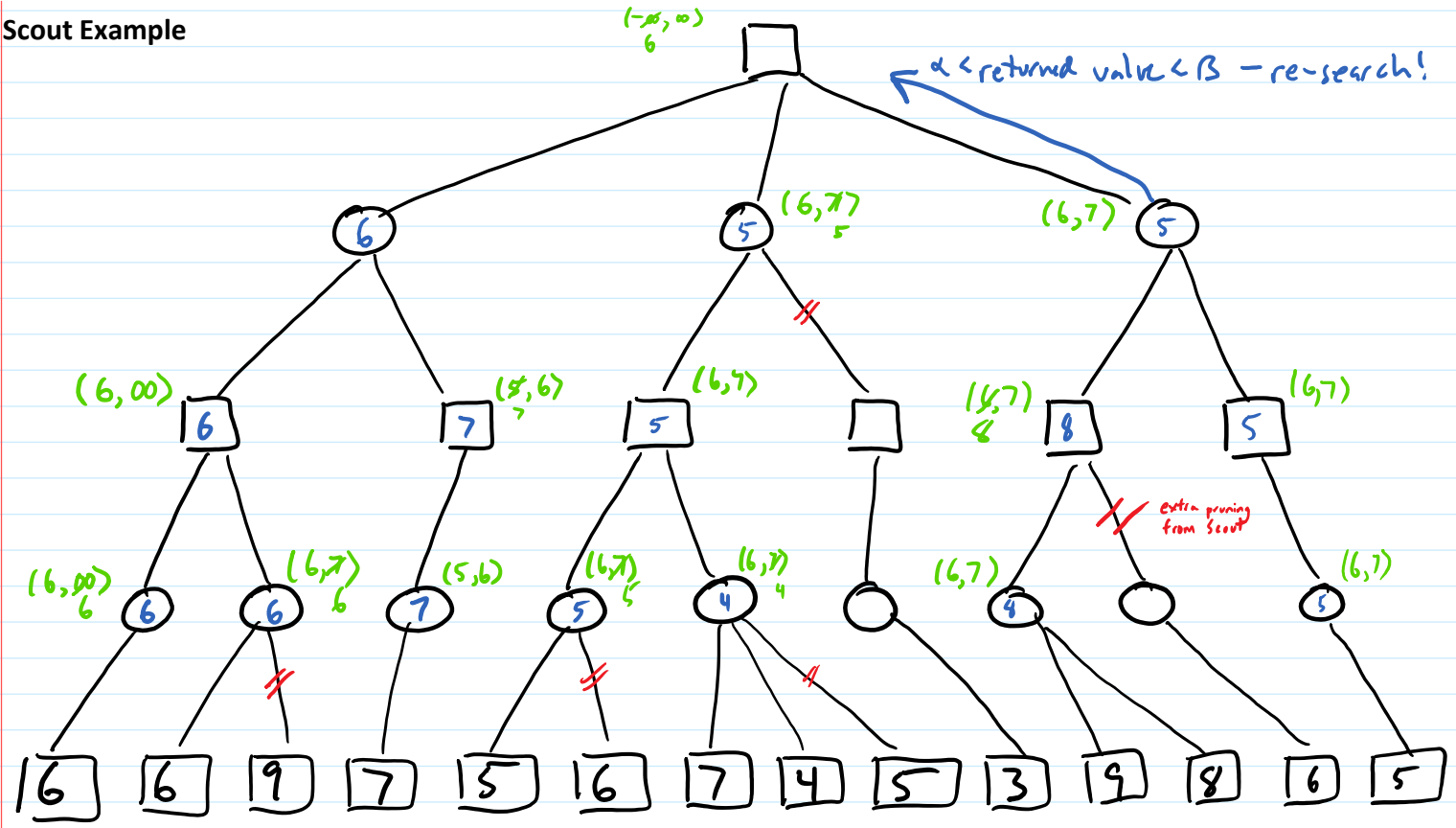
return best (or fail-hard version)

else

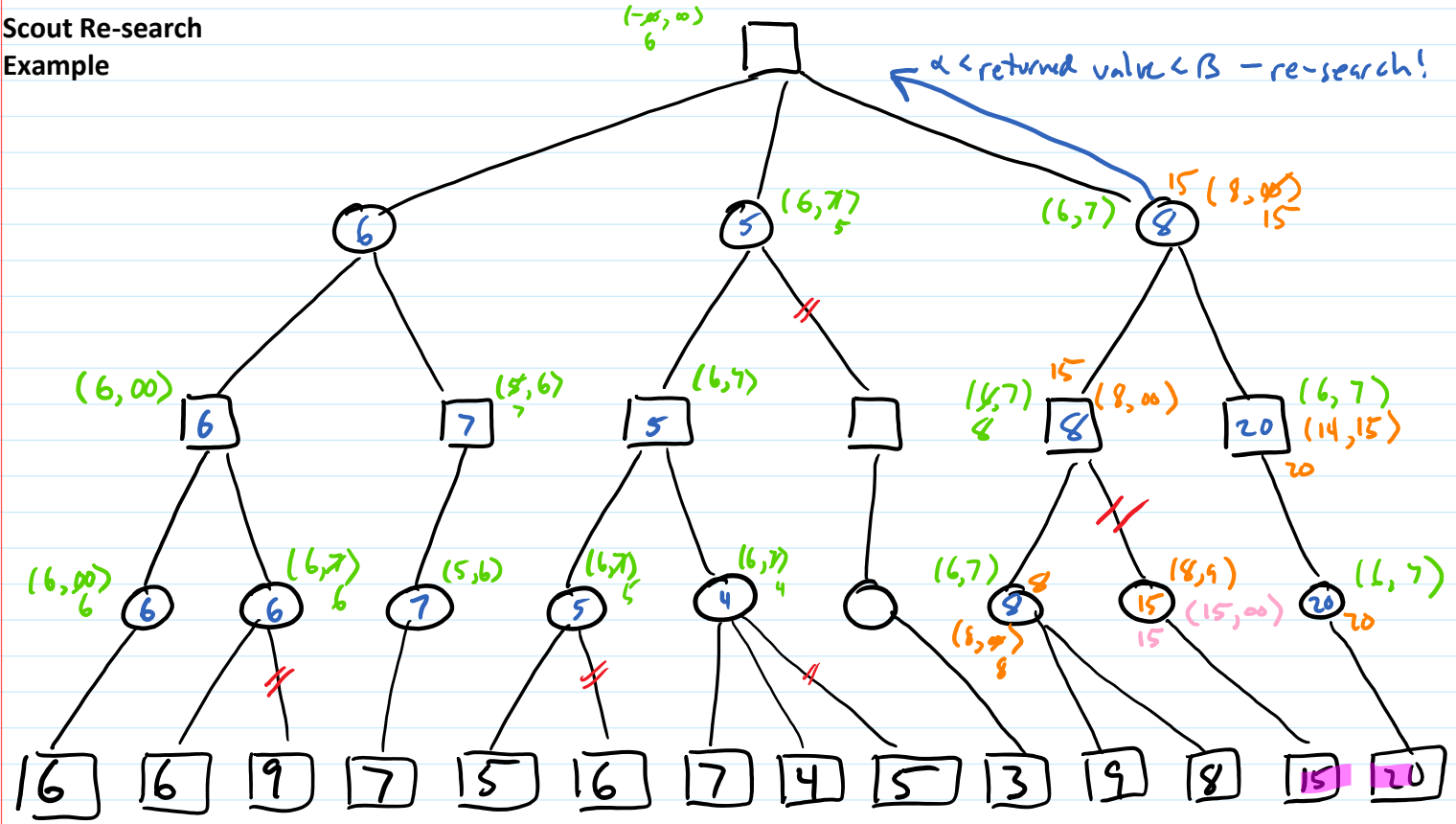
⋮ min position; symmetric

⋮

Scout Example

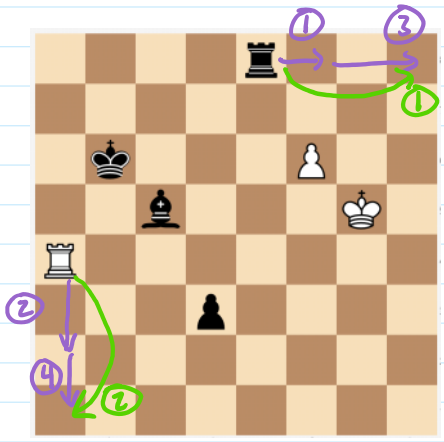


Scout Re-search Example



Transposition Table

Positions may be reachable by multiple sequences of moves



Keep table of values for all positions examined in tree

Keys: positions

Values: (value/bound, move, depth)

lower, upper bounds = if exact (given h, depth)

if value/bound from shallower search, ignore

Add check at start of A-B

if pos present and searched depth ≥ desired depth

if value is exact, return value

else if upper bound < α return val

else if lower bound and value ≥ β return val

Save returned values in table

fixed size – replacement policy

deepest

largest

newest

two-level

MTD-f

assuming integer values

pos to eval first guess

```
MTD-f (n, f, d)

    lowerBound ← -∞
    upperBound ← ∞
    g ← f

    while lowerBound < upperBound

        β ← max(lowerBound+1, g)
                with TT
        g ← A-B(n, β-1, β, d)
                                g-1, g most of the time
        if g < β    upperBound ← g
        else        lowerBound ← g

    return g
```