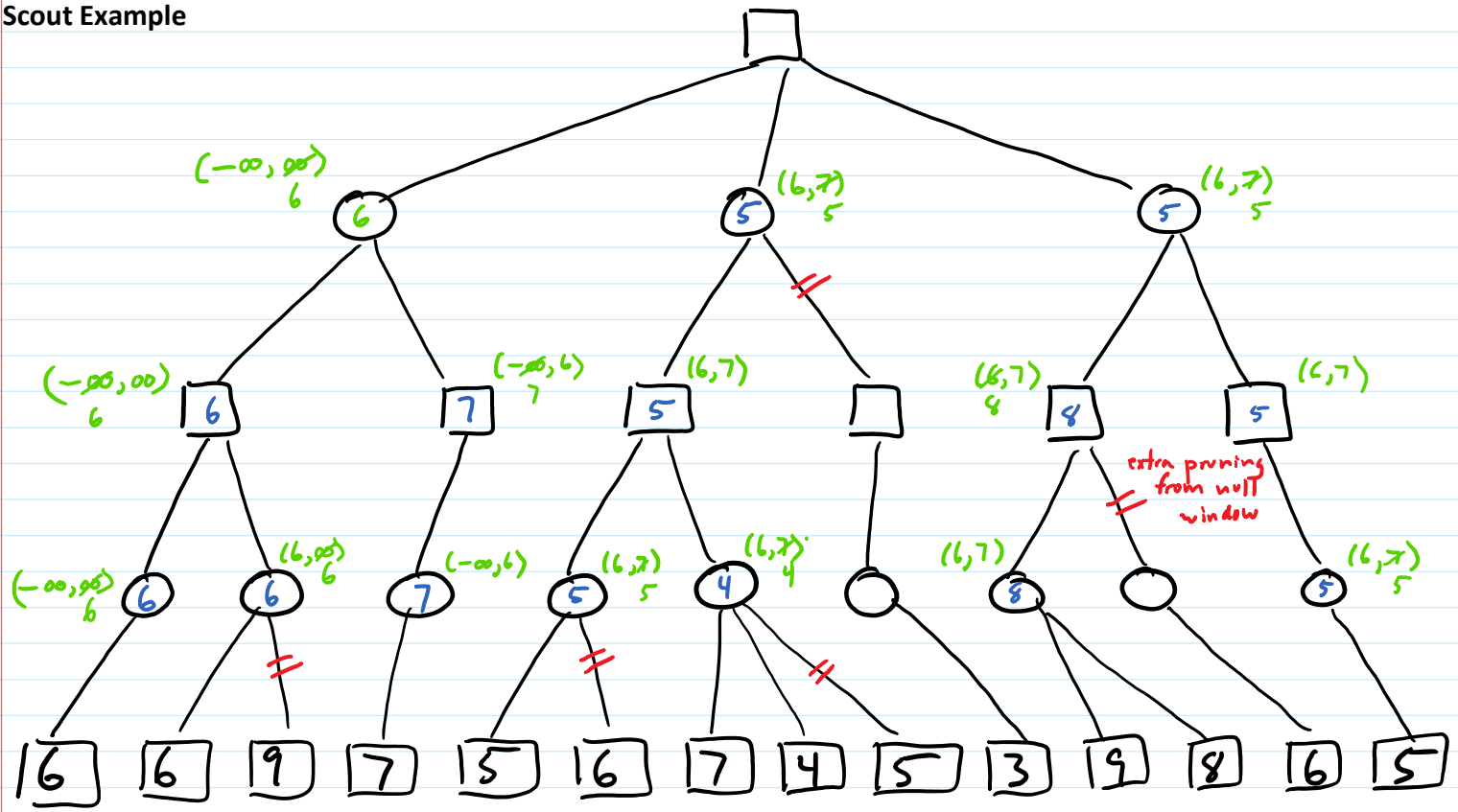
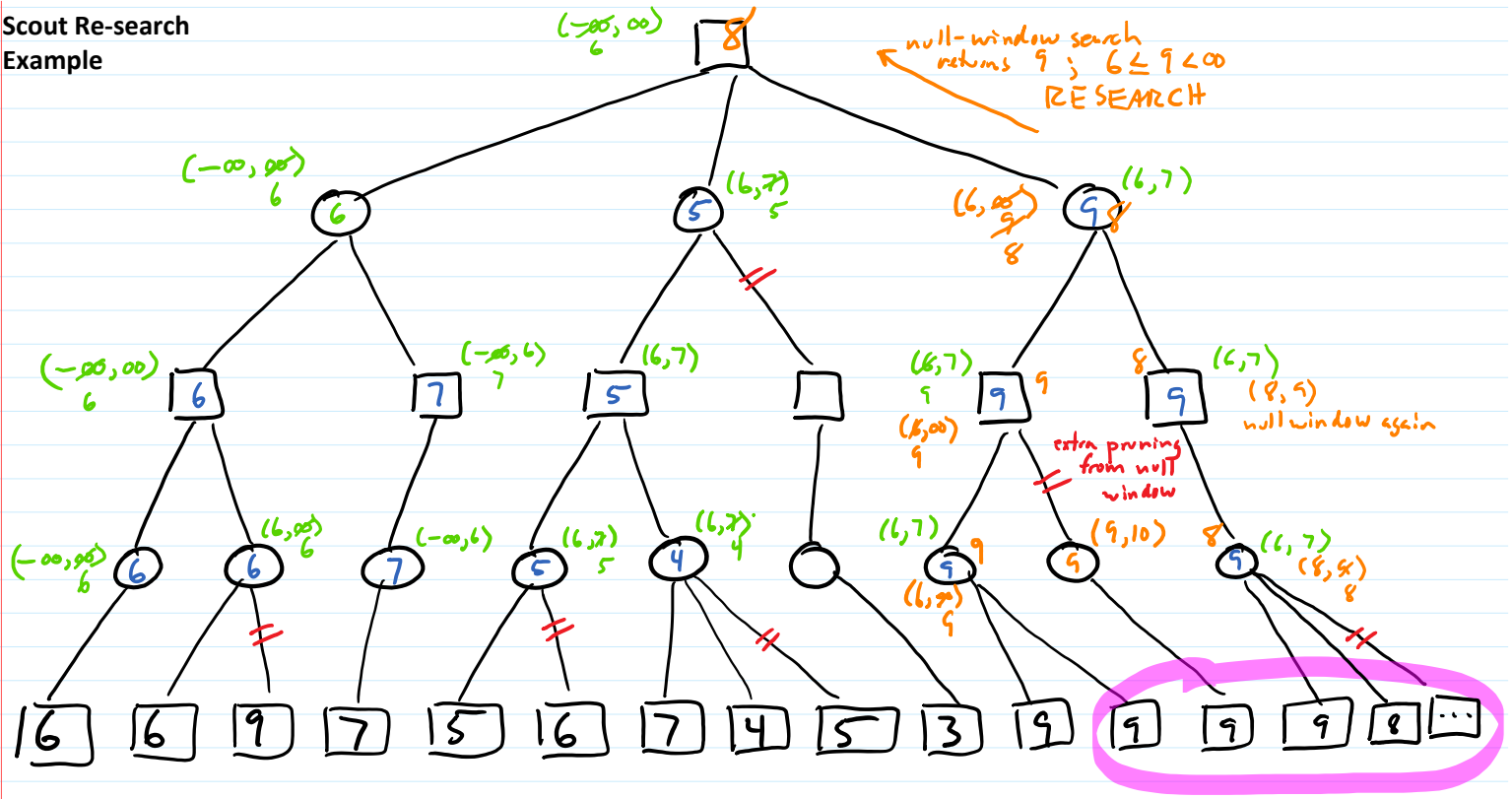
```
Scout (p, α, β, depth, h)
    if p is terminal then return value(p)
    if depth = 0 then return heuristic(p)
    if p is a max position
        best ← −∞
        b ← β

        for each reachable position p' and while α < β
            curr ← Alpha-Beta(p', α, b, depth−1, h)
                   (already null window; don't need Scout)
                                          (null window (except 1st iteration))
                                          is p' > best child so far?
            if b ≤ curr < β                (false for 1st iteration (b = β))
                                          yes; find value of p'
                curr ← Scout(p', b, β, depth−1, h)
                       (already know value is ≥ b)
            best ← max(best, curr)
            α ← max(α, best)              α = best unless best is bad (< original α)
            b ← max(α+1, best+1)          narrow window for next time around

        return best

    else
        ⋮ min position; symmetric
        ⋮
```

Scout Example

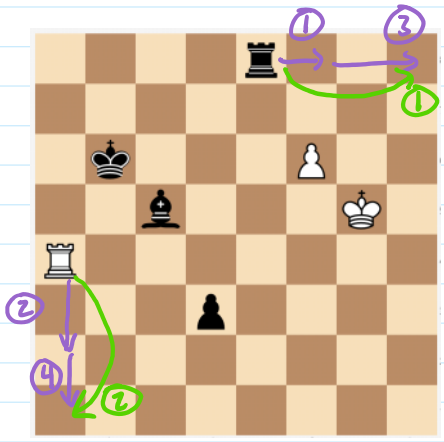


Scout Re-search Example



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 lowerband and value ≥ β return val

Save returned values in table

fixed size – replacement policy
- deepest
- largest
- newest
- two-level

Alpha-Beta Pruning — updated for transposition table

```
Alpha-Beta(p, α, β, depth, h, tt)   (returns bounds on value of p, given h and depth)
                                     ↪ (lowerBound, upperBound)
    if p in tt
        (low, up, d) ← tt.get(p)
        if d ≥ depth                 don't use results of shallow searches
                                     had exact value (given h and depth)   may have been result of deeper search
            if low = up   return (low, up)
            if low ≥ β    return (low, up)
            if up ≤ α     return (low, up)

    if p is terminal then return value(p)        assuming this is fast so needn't be stored

    if depth = 0 then
            value ← h(p)
            tt.put(p, (value, value), depth)     if heuristic is expensive it makes sense to save it
            return value                         exact value for given h

    if p is a max position

            bound ← (-∞, -∞)                     bounds on value of best child so far

            for each position p' reachable in one move from p and while α<β      cut off if know value of p > β
                bound ← max(bound, Alpha-Beta(p', α, β, depth-1, h, tt))
                α ← max(α, bound.low)
            if not all p' examined then bound.up = ∞
            tt.put(p, bound, depth)              record bounds on value of p
            return bound
        else (p is a min position)

            ;   symmetric
```

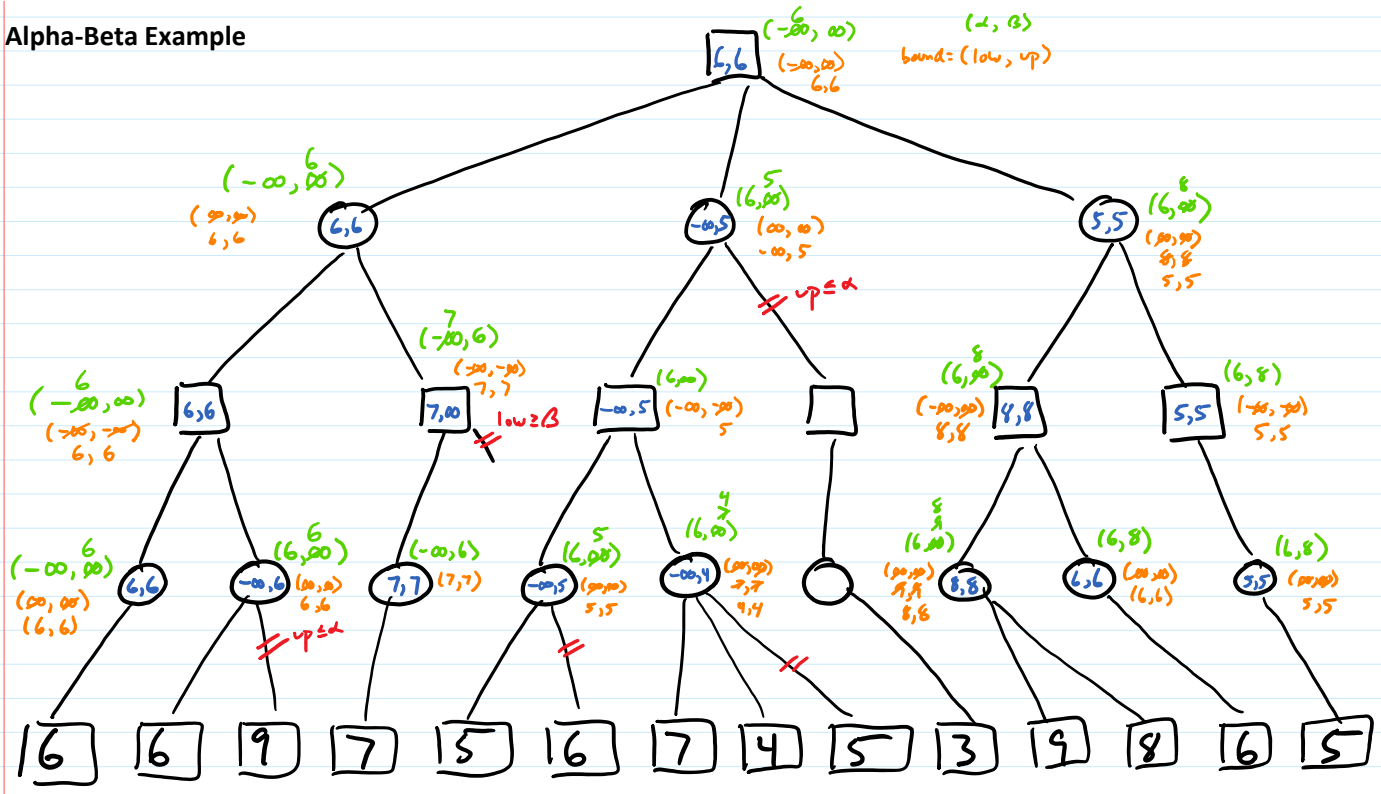
(max lower bounds, max upper bounds)

bound = ( )

A-B(p') = ( )

new best so far = ( )

Alpha-Beta Example



Modified example from http://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning

MTD-f

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
```