Complexity

[Game complexity - Wikipedia]

State Space – # of arrangements of pieces

checkers $\leq 2 \cdot 5^{32} \approx 10^{20}$

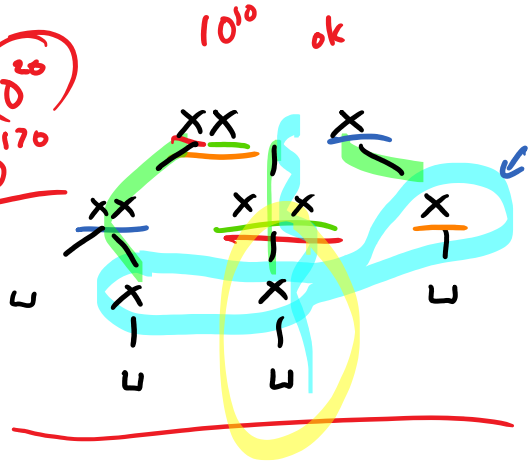
go $2 \cdot 3^{361} \approx 10^{170}$

Game tree size – # of nodes in game tree
sequences of moves

$\approx b^d$

Branching factor
b = # of moves available

Depth
d = length of game



Backtracking

for finite puzzles (so no cycles)

```
find_solution(s)
  if s is solved position return []

  for every possible move m
    s' <- result of move m at state s
    solution =  find_solution(s')

    if solution is not NIL
      return [m] + solution
    else
      undo move m on s
  return NIL
```

make move m on s

A*

```
g[s], f[s] <- infinity for all states s
g[initial] <- 0, f[initial] <- h(initial)
Q  <- priority queue(states, f[])

while Q not empty
  curr <- Q.extractMin()
  if curr is goal
    return goal, pi[goal], pi[pi[goal]], ..., initial
  else
    for every state n reachable from curr
      d <- g[curr] + cost(curr, n)
      if d < g[n] then
        if g[n] = infinity then
          f[n] <- d + h(n)
        else
          f[n] <- f[n] - (g[n] - d)
        g[n] <- d
        pi[n] = curr
        if not Q.contains(n) then
          f[n] <- d + h(n)
          Q.add(n, f[n])
        else
          Q.decreasePriority(n, f[n])
return NIL
```

g[s] = cost of best path so far init ⇝ s
f[s] = est of best path init ⇝ s ⇝ goal

INV: $f[n] = g[n] + h(n)$

BASIS: $f[i] = h(i) = h(i) + 0 = h(i) + g[i]$

$f[s] = \infty$

$g[s] = \infty = \infty + h(s)$

IND: outer if F → no changes

1st inner if T →

$f[n] = d + h(n) = g[n] + h(n)$

1st inner if F →

$f_{new}[n] = f_{old}[n] - g_{old}[n] + d$

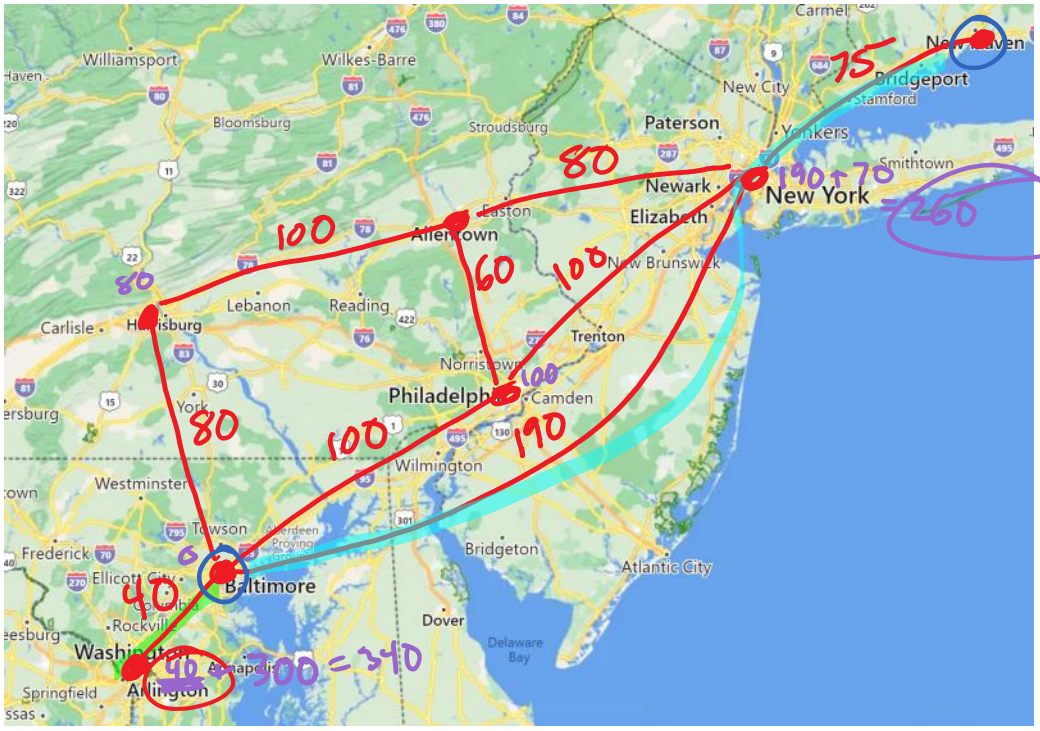
$= g_{old}[n] + h(n) - g_{old}[n] + d$

$= d + h(n)$

$= g_{new}[n] + h(n)$

heuristic h is admissible if it never overestimates

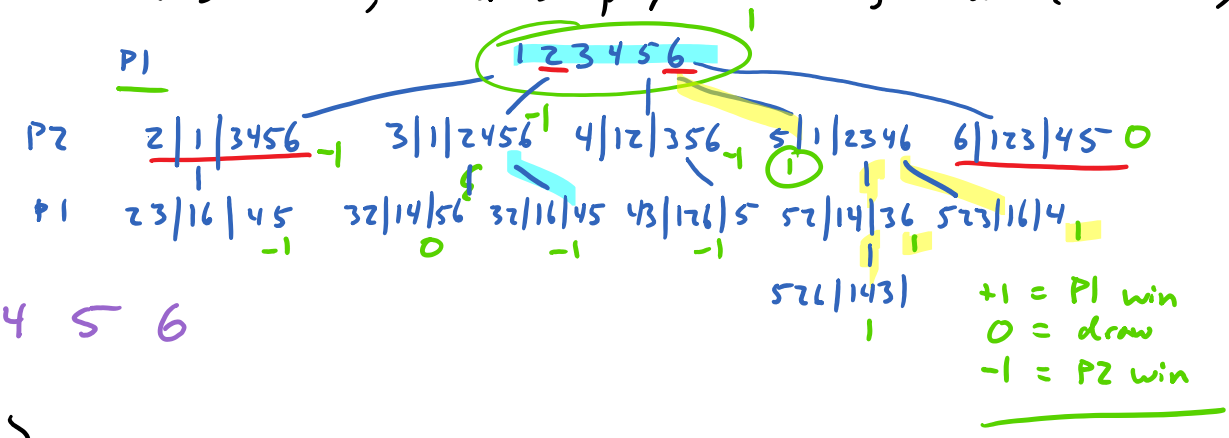
if h is admissible then A* returns optimal path



h(s) = great circle distance to goal

15-puzzle: $h(s) = \sum_{\text{tiles } t}$ distance from curr loc of t to its position in goal

Divisors: Start with 1...n, players take turns taking a number with remaining divisors; opponent gets all the remaining divisors. Game is over when no moves remain; winner is player with higher sum (draw if =)



1 2 3 4 5 6

Minimax(s)

if s is end of game
    return value according to rules
else
    S ← states reachable in 1 move from s
    if s is P1's move return max Minimax(s') s'∈S
    else return min Minimax(s') s'∈S

Graph Game

Graph: take turns coloring a vertex in a graph with your color
player who covers the most edges wins (draw if =)