COMP 472 Artificial Intelligence
State Space Search part #3
Intro to Heuristics video #3

- Russell & Norvig – Sections 3.5.1, 3.5.2, 4.1.1

Today

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first Search and Depth-first Search
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search 

      1. Intro to Heuristics $h(n)$
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

Informed Search (aka heuristic search)

可行的

- Most of the time, it is not feasible to do a systematic search, the search space is too large
  - e.g. state space of all possible moves in chess = 10^{120}
    - 10^{75} = nb of molecules in the universe
    - 10^{26} = nb of nanoseconds since the "big bang"

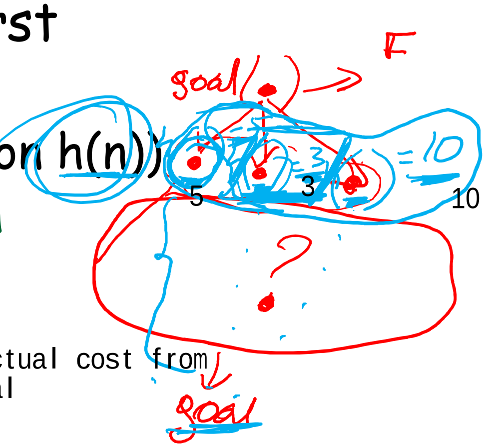
start

very large

- so far, all search algorithms have been uninformed

每个node equally promising goal

- ie. all nodes are equally promising
- we need a way to visit the most promising nodes first
  - most-promising = close to the goal state
  - so we need a estimate function (i.e. a heuristic function h(n))
  - so the search is now called informed/heuristic search



对children进行
h(n)
return的值就是actual cost from this node to goal

因此我们先测中间的那个因为他是3

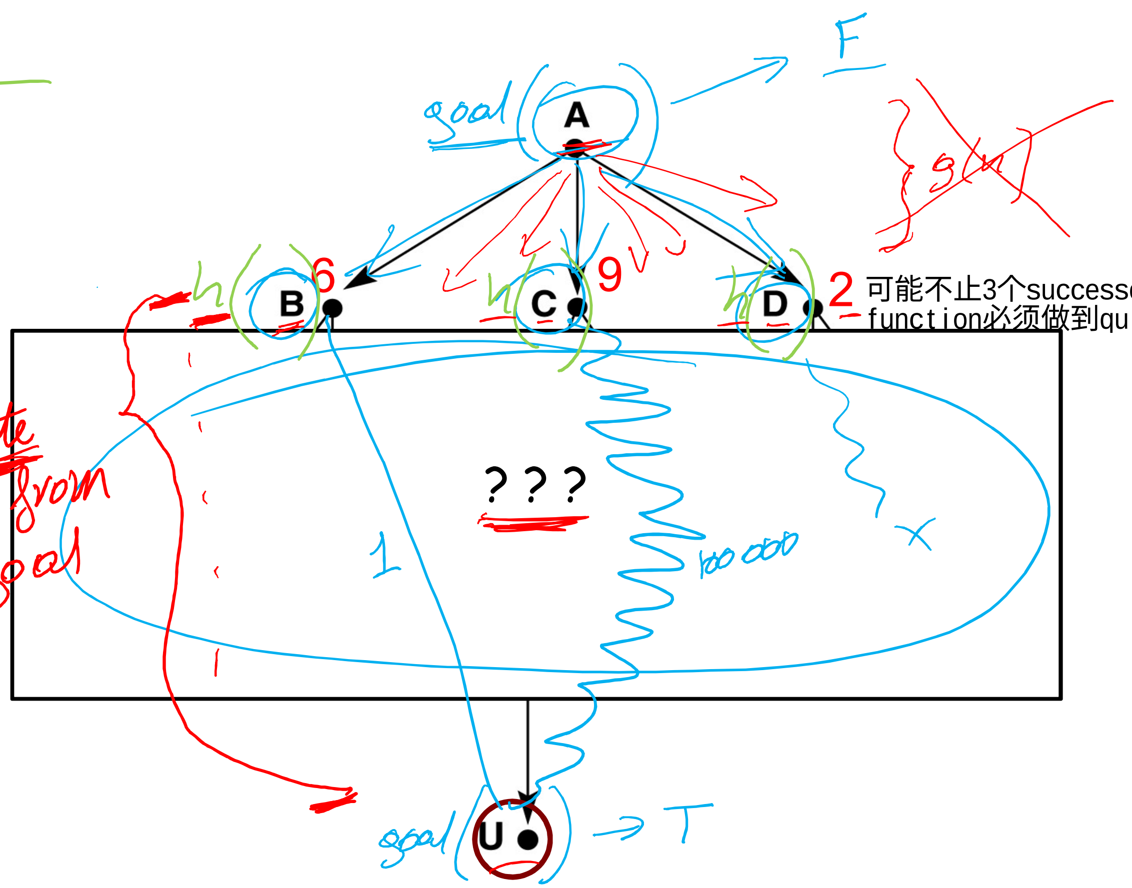
Heuristic - *Heureka!*

- Heuristic search:
  - A technique that improves the efficiency of search
  - Focus on nodes that seem most promising according to some function h(n)
  - Need an evaluation function (heuristic function) to estimate how close a node is to the goal
    of the cost
- Heuristic function *h(n)*:
  - a rule of thumb, a good bet, an estimate 概测法
  - but has no guarantee to be correct 不确保一定对
  - an approximation of the lowest cost from node n to the goal

h(n)

h(n) is an estimate of the cost from n to the goal

h(n)是estimate



可能不止3个successor point，因此我们要run很多次h(n),这个function必须做到quick

实际上B直通goal
c要很长时间
D通往死路

但根据我们的esitimate，他是不知道接下来的情况的，我们estimate D最小，因此选D

- **h(n)** = estimate of the lowest cost from *n* to *goal*

Designing Heuristics

设计h(n)取决于search问题

- h(n) are highly dependent on the search domain
- A h(n) whose value is closer to the actual cost to the goal will lead to:
  - a shorter search path
  - less backtracking
  - i.e. less nodes are visited/searched for nothing
  - but this is not always the best idea...
    - it depends on the computational cost of h(n)

快，比较准

$h_1(n)$ // good estimate but quick to run

vs

$h_2(n)$ // much closer estimate to the actual cost but slow to run

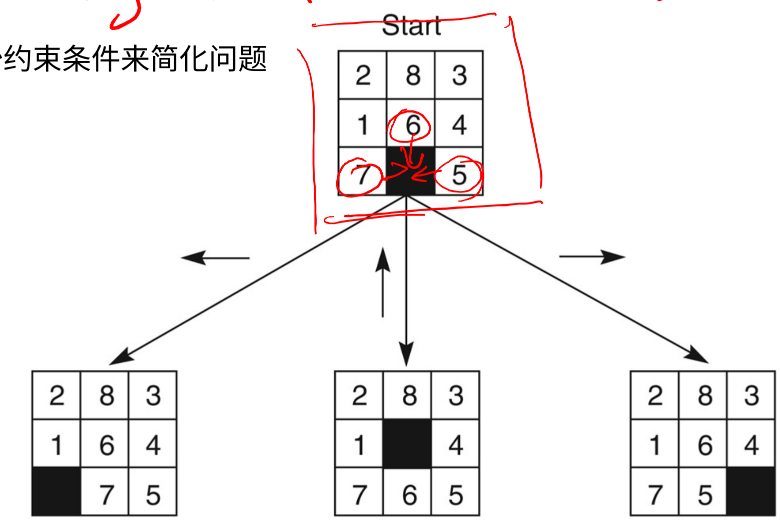
准，慢

权衡 trade-off between length of the search path and total execution time

Example: 8-Puzzle - Heuristic 1

key: relax constraints of the problem to simplify it

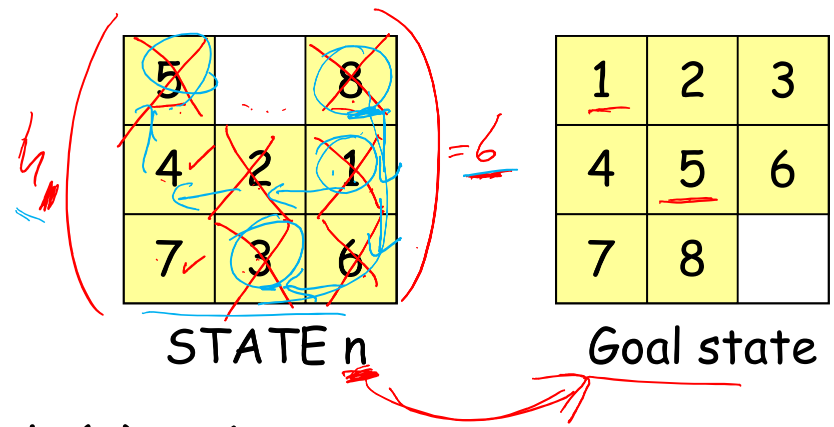
减少约束条件来简化问题



relax的constrait越多，heuristic越快，越不准

- h_1: Simplest heuristic
  - Hamming distance : count numbered number of tiles out of place when compared with goal

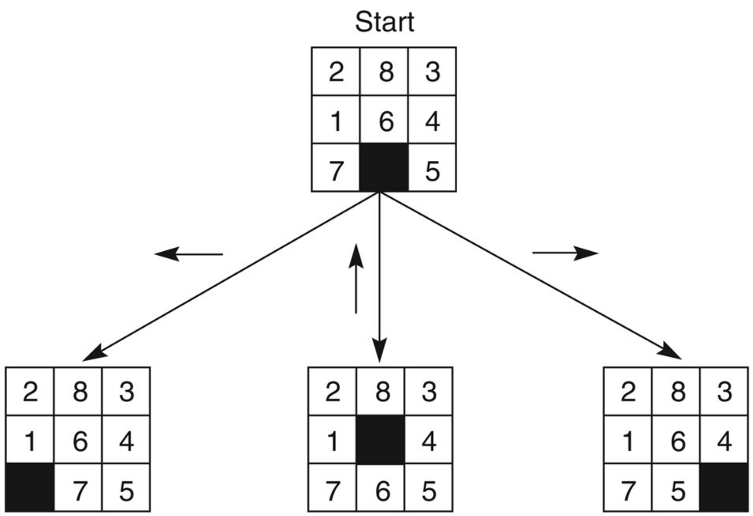
实际情况很复杂，但我们减少约束条件，记录有几个方块不在应该在的位置



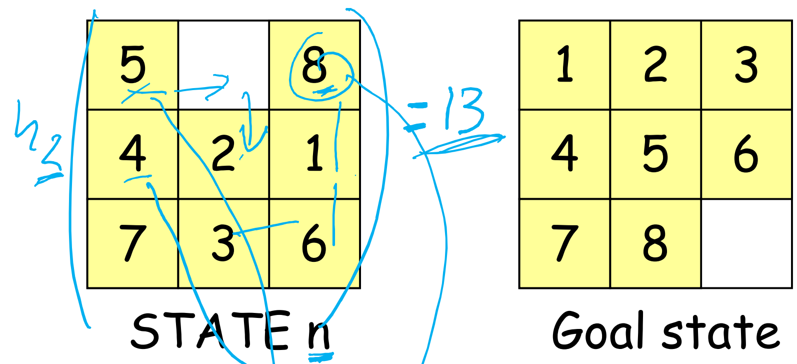
- $h_1(n) = 6$
  - does not consider the distance tiles have to be moved

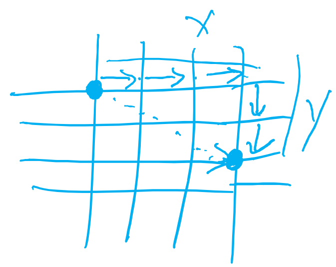
source: G. Luger (2005)

Example: 8-Puzzle - Heuristic 2



- h_2: Better heuristic
  - Manhattan distance: sum up all the distances by which tiles are out of place



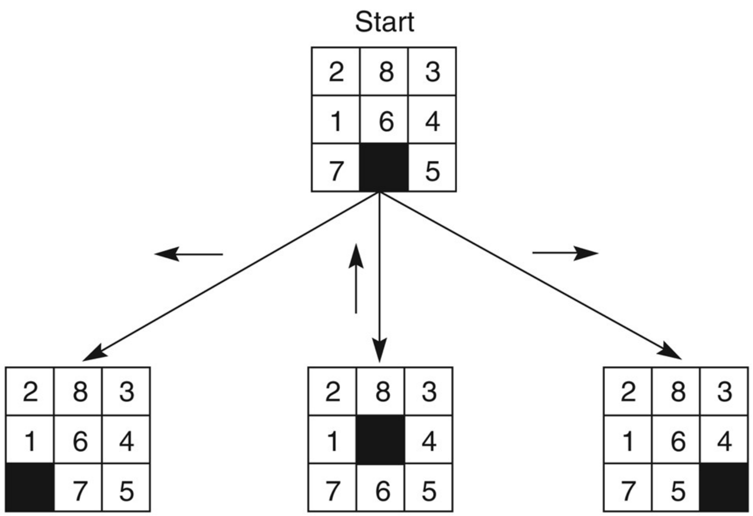


2 1 2 3 6

- $h_2(n) = 2+3+0+1+3+0+3+1$
  $= 13$

source: G. Luger (2005)

Example: 8-Puzzle - Heuristic 3



- h_3: Even Better
  - sum of permutation inversions 排列反演
  - See next slide…

source: *G. Luger (2005)*

$h_3(N)$ = sum of permutation inversions

h_3

5		8
4	2	1
7	3	6

STATE n

$=16$

5		8	4	2	1	7	3	6

对于每个tile，记录他右边有几个tile实际上应该在左边

- For each numbered tile, count how many tiles on its right should be on its left in the goal state.

- $h_3(n) = n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6$
  $= 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0$
  $= 16$

1	2	3
4	5	6
7	8	

Goal state

Heuristics for the 8-Puzzle

这个heuristics function和uniform没区别

$h_0(n) = 1 \ \forall n$ where n is not the goal

$h_0(goal) = 0$

goal是0其他是11

5		8
4	2	1
7	3	6

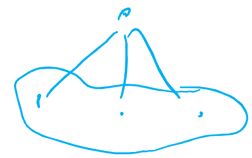
STATE n

1	2	3
4	5	6
7	8	

Goal state

not interesting ≡ an informede search

- $h_1(n)$ = misplaced numbered tiles
  = 6



- $h_2(n)$ = Manhattan distance
  = 2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13

- $h_3(n)$ = sum of permutation inversions
  $= n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6$
  = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16

is $h_3(n)$ better?

- $h_3(n)$ may return a better estimate ≈ closer to the actual cost
  - -> less backtracking / shorter search path
- BUT, $h_3(n)$ may be longer to compute // longer execution time
  - maybe overall longer to get the solution path
  - solution path may NOT be necessarily of lower cost (*more on this later*)

better estimate are not better overall estimate

Today

1. State Space Representation
2. State Space Search
   a) Overview
   b) Uninformed search
      1. Breadth-first and Depth-first
      2. Depth-limited Search
      3. Iterative Deepening
      4. Uniform Cost
   c) Informed search
      1. Intro to Heuristics
      2. Hill climbing
      3. Greedy Best-First Search
      4. Algorithms A & A*
      5. More on Heuristics
   d) Summary

Up Next

1. State Space Representation
2. State Space Search
    a) Overview
    b) Uninformed search
        1. Breadth-first and Depth-first
        2. Depth-limited Search
        3. Iterative Deepening
        4. Uniform Cost
    c) Informed search
        1. Intro to Heuristics
        2. Hill climbing
        3. Greedy Best-First Search
        4. Algorithms A & A*
        5. More on Heuristics
    d) Summary

h(n)