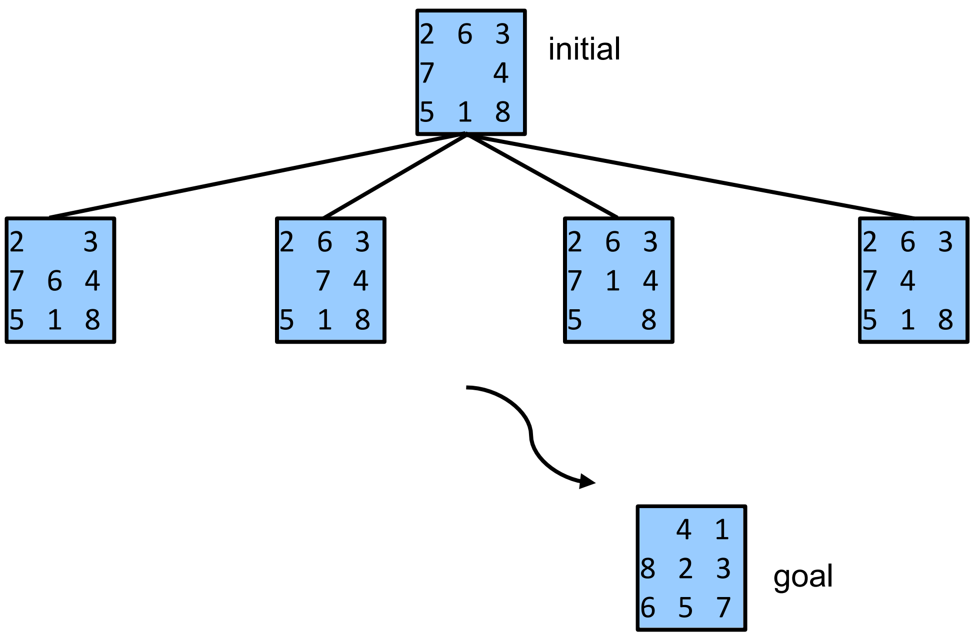
Informed / heuristic search

- Search is done intelligently

- Search is likely to run fast

- Informed search algorithms
    - best first search
    - A* search

Best first search



- We like to reach the goal fast

- Which of the four boards to search
    - search the board that is closer to the goal

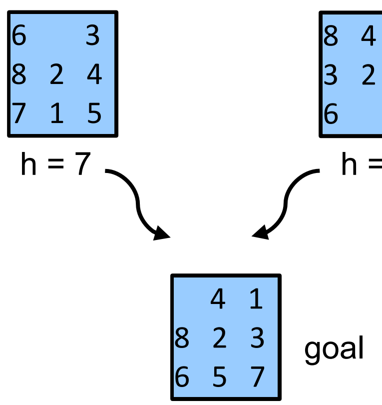
- How to decide which board is closer to the goal
    - use a heuristic

Heuristic

- Many heuristics are possible
    - mismatch heuristic
    - taxi distance heuristic

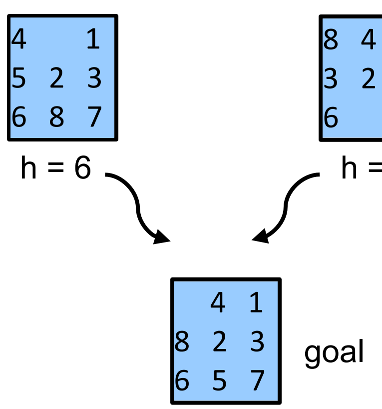
- Heuristic value of a board = number of mismatches between the board and the goal board

-



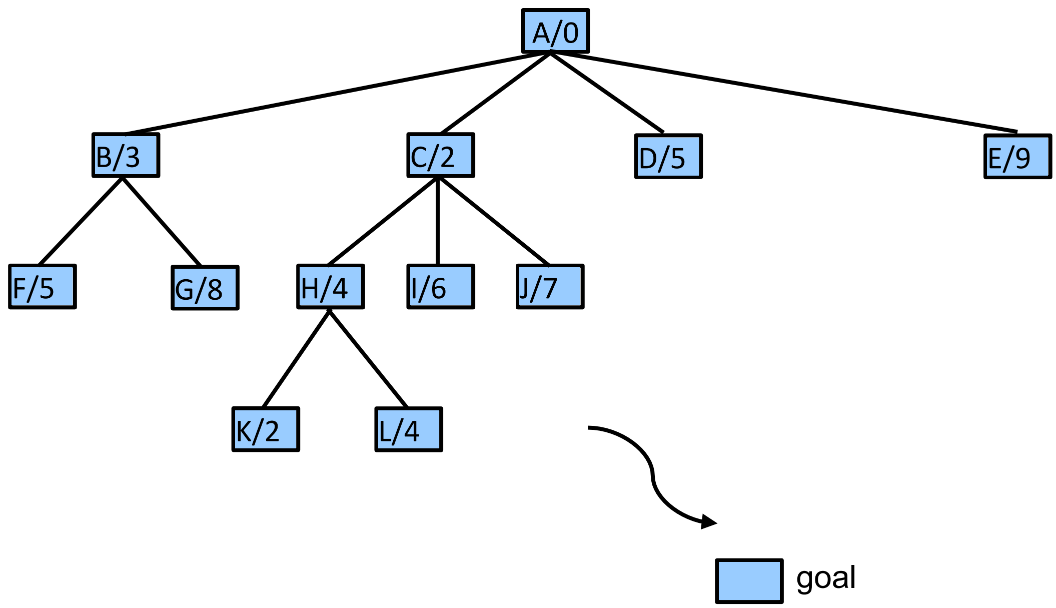
- Heuristic value of a board = sum of taxi distances between mismatches between the board and the goal board

-



- If h value is smaller then the board is closer to the goal
  If h value is larger then the board is farther from the goal

Best first search example



- choose A, expand A

  choose minimum B/3 C/2 D/5 E/9
  choose C, expand C

  choose minimum B/3 D/5 E/9 H/4 I/6 J/7
  choose B, expand B

  choose minimum D/5 E/9 H/4 I/6 J/7 F/5 G/8
  choose H, expand H

  choose minimum D/5 E/9 I/6 J/7 F/5 G/8 K/2 L/4
  choose K, expand K

  keep doing until goal is reached

- Heuristic values h are made up in the example above

Best first search algorithm

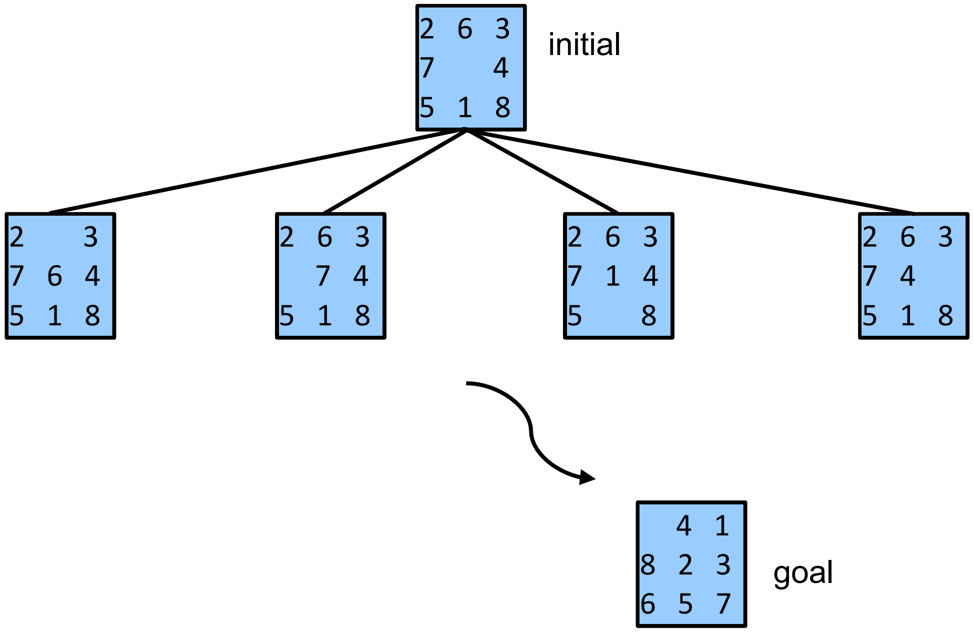
- open list has initial state with h = 0

```
closed list is empty
while open list is not empty
{
     S = remove the state with minimum h value from open list
     insert S into closed list
     if S is goal state
          display path from initial state to S
          stop search
     else
     {
          generate successors of S
          for each successor C
               h value of C = heuristic value of C
               if C is not in open list and C is not in closed list
                    insert C into open list
     }
}
say there is no path
```

- Best first search is likely to run fast

- Best first search may not find the shortest path

A* search



- We like to
  - reach the goal fast
  - find the shortest path to the goal

- Which of the four boards to search
  - search the board that is closer to the goal and has a shorter path

- How to decide which board is closer to the goal and has a shorter path
  - use an evaluation function

Evaluation function

- Evaluation function f has two components
  - heuristic function h
  - path cost function g

- Evaluation function

    f(board) = h(board) + g(board)

- Heuristic function

    h(board) = heuristic value of board

- Path cost function

    g(board) = path cost of board (length of path from initial)

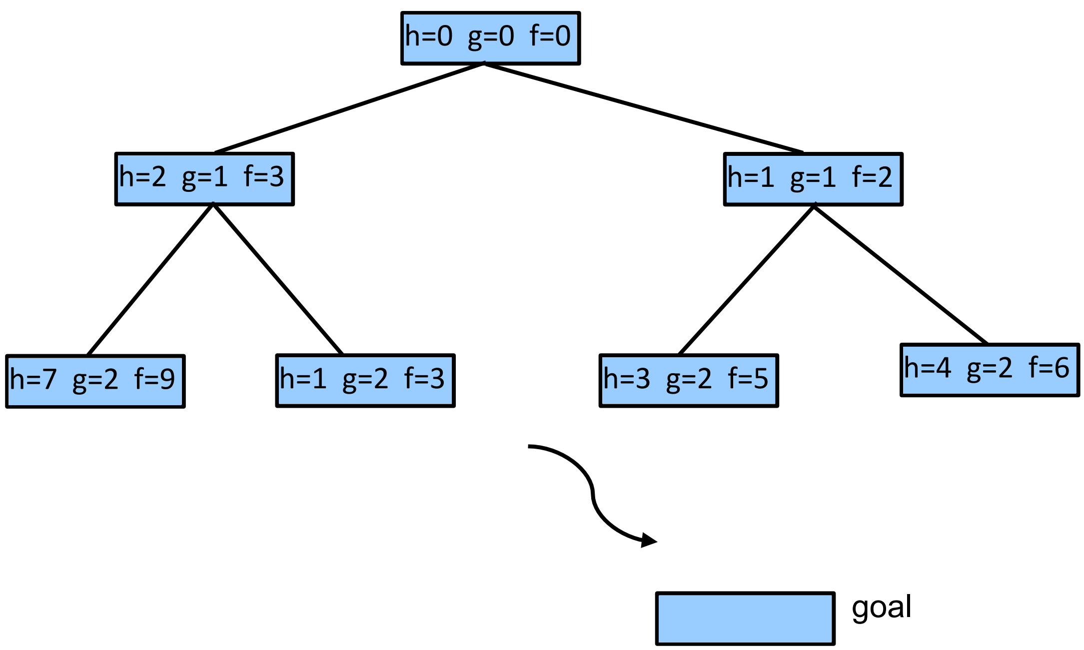
- f = h + g

  h value helps to find a path fast

  g value helps to find the shortest path

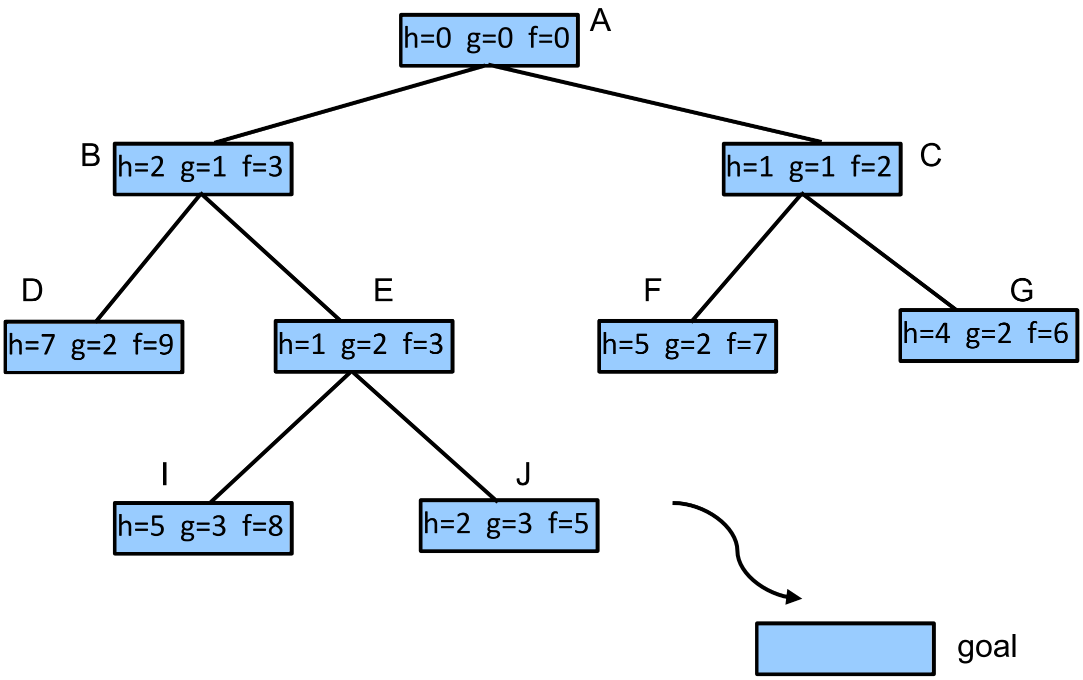
  f value helps to do both

-



- In the above example
    - heuristic values h are made up
    - each edge cost is 1

A* search example



- choose A, expand A

  choose minimum B/3 C/2
  choose C, expand C

  choose minimum B/3 F/7 G/6
  choose B, expand B

  choose minimum F/7 G/6 D/9 E/3
  choose E, expand E

  choose minimum F/7 G/6 D/9 I/8 J/5
  choose J, expand J

  keep doing until goal is reached

- In the example - Heuristic values are made up, edge cost is 1

A* search algorithm

```
- open list has initial state with f = 0
  closed list is empty
  while open list is not empty
  {
        S = remove the state with minimum f value from open list
        insert S into closed list
        if S is goal state
             display path from initial state to S
             stop search
        else
        {
              generate successors of S
              for each successor C
              {
                     h value of C = heuristic value of C
                     g value of C = g value of S + edge cost from S to C
                     f value of C = h value of C + g value of C
                     if C is not in closed list
                     {
                           if C is not in open list
                                 insert C into open list
                           else
                           {
                                compare f values of new copy of C and
                                old copy of C in open list
                                if new f value < old f value
                                     replace old copy of C in open list with
                                     new copy of C
                            }
                     }
              }
        }
  }
```

say there is no path

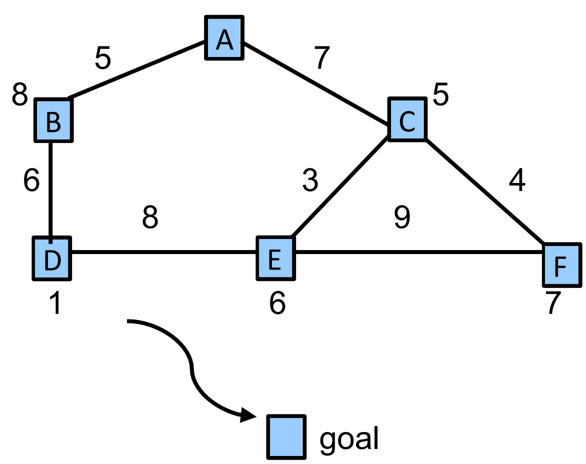
A* search on graph

- A* search can be used on graphs

- How to compute g values / path cost
    - use given edge lengths

- How to compute h values / heuristic
    - h value of node = straight line distance between node and goal

-



- choose A, expand A

  choose minimum B/5+8=13 C/7+5=12
  choose C, expand C

  choose minimum B/5+8=13 E/10+6=16 F/11+7=18
  choose B, expand B

  choose minimum E/10+6=16 F/11+7=18 D/11+1=12
  choose D, expand D

keep doing until goal is reached

- In the above example - heuristic values h (straight line distance) are made up

A* search, best first search, uniform cost search

- Best first search and uniform cost search are special cases of A* search
    - A* search uses f = h + g
    - A* search becomes best first search when g = 0, f = h
    - A* search becomes uniform cost search when h = 0, f = g

- A* search - finds shortest path, likely to run fast

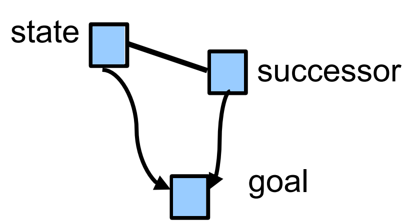
- Best first search - likely to run fast, may not find shortest path

- Uniform cost search - finds shortest path, may not run fast

Heuristic function condition

- Heuristic function h must satisfy a condition for A* to work

- h(state) $\leq$ h(successor) + edge(state, successor)

- 

- Condition is satisfied by
    - mismatch heuristic
    - taxi distance heuristic
    - straight line distance heuristic