Alpha-Beta Pruning Example
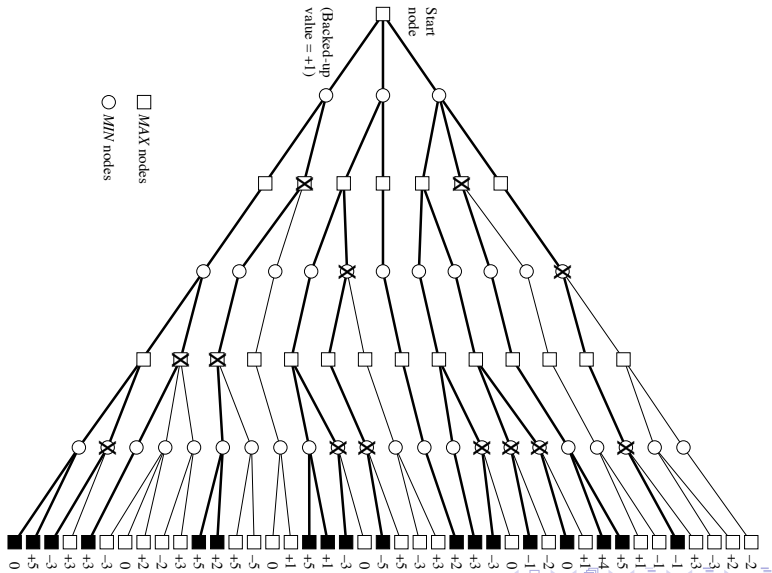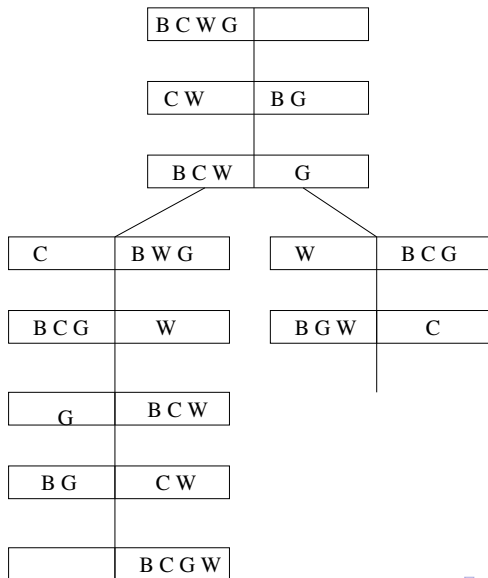
Problem #3

A boy brings a basket of cabbage, a wolf, and a goat to a river. There is a small boat on their river bank that they can use to cross to the other side. However, the boat is so small that it can hold only the boy and the cabbage or the boy and one of the animals. The boy cannot leave the goat alone with either the cabbage or the wolf. How should the boy use the boat to take the cabbage and the animals to the other side of the river in such a way that the number of river crossings is minimal and that the goat is never left alone with either the wolf or the cabbage on any side of the river? Use iterative deepening search to solve this problem.

Problem #3: Solution

B C W G	

C W	B G

B C W	G

C	B W G

W	B C G

B C G	W

B G W	C

G	B C W

B G	C W

	B C G W

Problem #5

A simple sliding-tile puzzle consists of 3 numbered tiles and an empty
space. The puzzle has two legal moves with associate costs: (i) A tile may
move into an adjacent empty location. This has a cost of 1. (ii) A tile can
move over one other tile into the empty position. This has a cost of 2.
The start and goal positions are given below.

| 1 | 2 | 3 | |

| 3 | 1 | 2 | |

Problem #5 (cont.)

Set up a state-space search formulation for this puzzle:

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description. Write a lisp function that takes a state as input and outputs a list of states that can be reached from that state. Include the parent and the cost fields.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path using your heuristic function.

e) Is your solution path optimal? Give a formal argument.

Problem #5: Solution

a) A list of three numbers and a white space. Start: $(1\ 2\ 3\ -)$, Goal: $(3\ 1\ 2\ -)$

b) $Swap1(i, j)$: swaps a tile at position i with an empty tile at position j, where $|i - j| = 1$
$Swap2(i, j)$: swaps a tile at position i with an empty tile at position j, where $|i - j| = 2$

c) City-block/Manhattan distance: sum of absolute distances of all numbered tiles form their goal positions
$$\hat{h}((1\ 2\ 3\ -)) = 1 + 1 + 2 = 4$$

d) $Swap2(2, 4)$, $Swap1(1, 2)$, $Swap2(3, 1)$, $Swap1(4, 3)$

e) $\hat{h}$ is admissible, A^* with an admissible heuristics produces optimal solution

Problem #5: Solution (d)
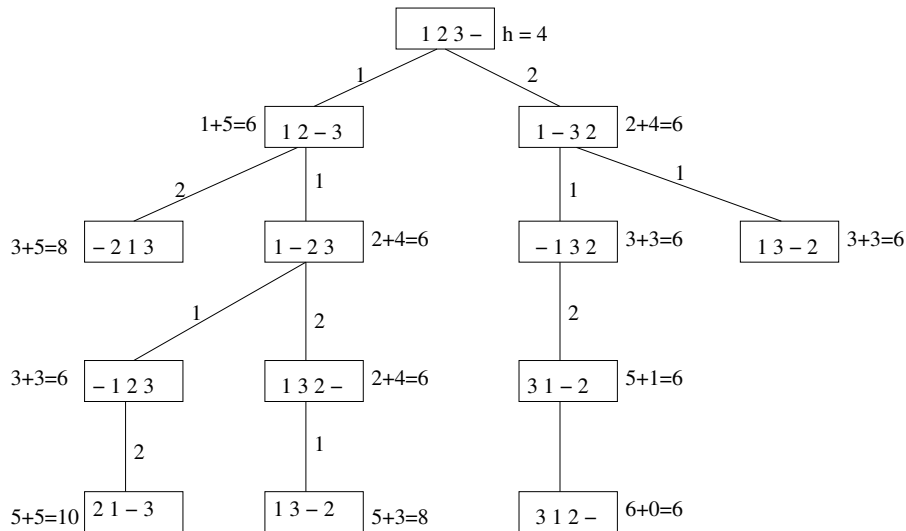
Problem #13

The 8-puzzle consists of eight numbered, movable tiles set in a 3×3 frame. One cell of the frame is always empty thus making it possible to move and adjacent numbered tile into the empty cell—or, we could say, to move the empty cell. Two configurations of tiles are given. Consider the problem of changing the initial configuration into the goal configuration.

2	4	3
1	6	8
5		7

start

1	2	3
4		8
5	6	7

goal

Problem #13 (cont.)

Set up a state-space search formulation for this puzzle:

  a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

  b) Name the operators on states and describe what each operator does to a state description.

  c) Propose a heuristic function $\hat{h}$ for solving this problem.

  d) Use the A^* algorithm to find a solution path.

  e) Is your solution path optimal? Give a formal argument.

Problem #13: Solution

Solution:

a) 3×3 array

$$start: \begin{bmatrix} 2 & 4 & 3 \\ 1 & 6 & 8 \\ 5 & - & 7 \end{bmatrix}, \quad goal: \begin{bmatrix} 1 & 2 & 3 \\ 4 & - & 8 \\ 5 & 6 & 7 \end{bmatrix}$$

b) $MU(i, j)$: move empty tile at (i, j) up, empty tile is now at $(i - 1, j)$
$MD(i, j)$: move empty tile at (i, j) down, empty tile is now at $(i + 1, j)$
$ML(i, j)$: move empty tile at (i, j) left, empty tile is now at $(i, j - 1)$
$MR(i, j)$: move empty tile at (i, j) right, empty tile is now at $(i, j + 1)$

c) City-block/Manhattan distance: sum of the absolute i and j distances of all tiles from their goal positions

d) $MU(3, 2)$, $MU(2, 2)$, $ML(1, 2)$, $MD(1, 1)$, $MR(2, 1)$

e) h is admissible since it solves an easier problem, therefore A^* results in an optimal path

Problem #13: Solution (d)
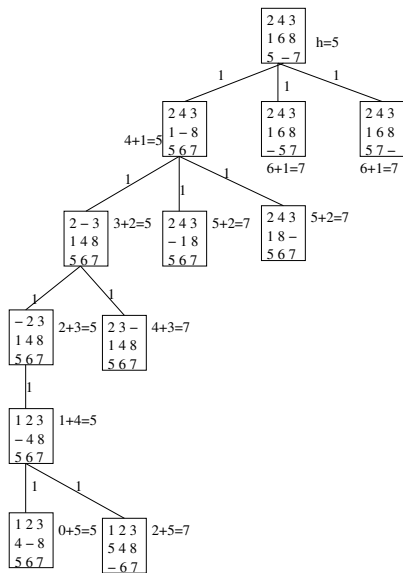
Problem #29

A simple sliding-tile puzzle consists of 4 numbered tiles. The puzzle has two legal moves. (i) Two adjacent tiles can switch positions. This has a cost of 2. (ii) Two non-adjacent tiles can switch positions. This has a cost of 3.

1	2	3	4

start

4	3	1	2

goal

Problem #29 (cont.)

Set up a state-space search formulation for this puzzle:

a) Specify the form of state descriptions, the starting state, and the goal state for this problem.

b) Name the operators on states and describe what each operator does to a state description.

c) Propose a heuristic function $\hat{h}$ for solving this problem.

d) Use the A^* algorithm to find a solution path.

e) Is your solution path optimal? Give a formal argument.

Problem #29: Solution

Solution:

a) A list of 4 numbers

$$start : (1\ 2\ 3\ 4), \quad goal : (4\ 3\ 1\ 2)$$

b) $Swap2(i, j)$: swap tiles at positions i and j, where $|i - j| = 1$
   $Swap3(i, j)$: swap tiles at position i and j where $|i - j| > 1$

c) $h = \#$ of tiles out of place by $1 + 1.5 \times \#$ of tiles out of place by 2 or more

d) $Swap2(2, 3)$, $Swap2(3, 4)$, $Swap3(1, 3)$

e) h is admissible since it solves an easier problem, therefore A^* results in an optimal path

Problem #13: Solution (d)
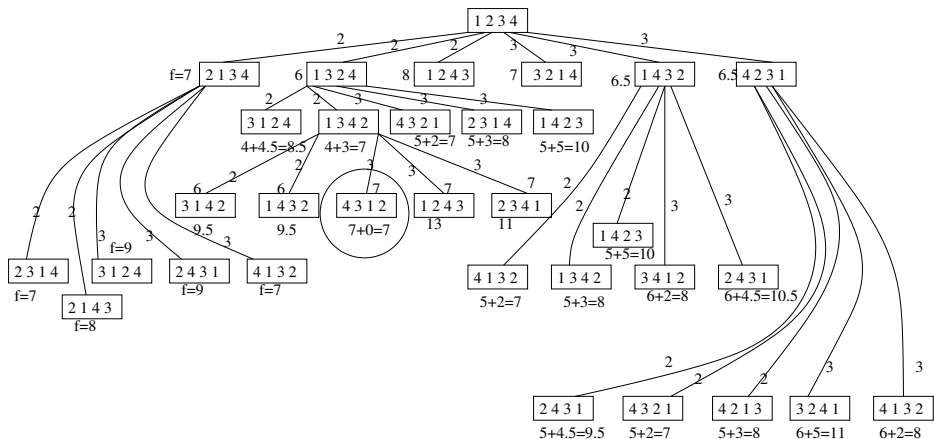
Problem #31

Show how different techniques can be used to solve the 4-queens
placement problem (place 4-queens on a 4×4 board so that there is at
most one queen in each row, column, or a diagonal of the board. Solve
this problem using the following tecniques:

  a. Backtracking search.
  b. Forward checking.
  c. Min-conflicts algorithm. To initialize the problem place all four
     queens on the main diagonal of the board.

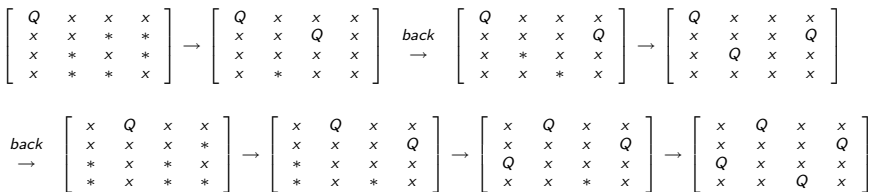# Problem #31: Solution

a. Backtracking search.

$$\begin{bmatrix} Q & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} Q & * & * & * \\ * & * & Q & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \dots \overset{back}{\rightarrow} \begin{bmatrix} Q & * & * & * \\ * & * & * & Q \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} Q & * & * & * \\ * & * & * & Q \\ * & Q & * & * \\ * & * & * & * \end{bmatrix}$$

$$\dots \overset{back}{\rightarrow} \begin{bmatrix} * & Q & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & Q & * & * \\ * & * & * & Q \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & Q & * & * \\ * & * & * & Q \\ Q & * & * & * \\ * & * & * & * \end{bmatrix} \rightarrow \begin{bmatrix} * & Q & * & * \\ * & * & * & Q \\ Q & * & * & * \\ * & * & Q & * \end{bmatrix}$$

b. Forward checking.

$$\begin{bmatrix} Q & x & x & x \\ x & x & * & * \\ x & * & * & * \\ x & * & * & x \end{bmatrix} \rightarrow \begin{bmatrix} Q & x & x & x \\ x & x & Q & x \\ x & x & x & x \\ x & * & x & x \end{bmatrix} \overset{back}{\rightarrow} \begin{bmatrix} Q & x & x & x \\ x & x & x & Q \\ x & * & x & x \\ x & x & * & x \end{bmatrix} \rightarrow \begin{bmatrix} Q & x & x & x \\ x & x & x & Q \\ x & Q & x & x \\ x & x & x & x \end{bmatrix}$$

$$\overset{back}{\rightarrow} \begin{bmatrix} x & Q & x & x \\ x & x & x & * \\ * & x & * & x \\ * & x & * & * \end{bmatrix} \rightarrow \begin{bmatrix} x & Q & x & x \\ x & x & x & Q \\ * & x & x & x \\ * & x & * & x \end{bmatrix} \rightarrow \begin{bmatrix} x & Q & x & x \\ x & x & x & Q \\ Q & x & x & x \\ x & x & * & x \end{bmatrix} \rightarrow \begin{bmatrix} x & Q & x & x \\ x & x & x & Q \\ Q & x & x & x \\ x & x & Q & x \end{bmatrix}$$

Problem #31: Solution

c. Min-conflicts algorithm. To initialize the problem place all four queens on the main diagonal of the board.

$$
\begin{bmatrix} Q' & * & * & * \\ * & Q' & * & * \\ * & * & Q' & * \\ * & * & * & Q' \end{bmatrix}, \; c_1 \begin{bmatrix} 3 \\ 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} * & * & * & * \\ Q' & Q' & * & * \\ * & * & Q' & * \\ * & * & * & Q' \end{bmatrix}, \; c_2 \begin{bmatrix} 1 \\ 3 \\ 2 \\ 2 \end{bmatrix} \begin{bmatrix} * & Q' & * & * \\ Q' & * & * & * \\ * & * & Q' & * \\ * & * & * & Q' \end{bmatrix},
$$

$$
c_3 \begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \end{bmatrix} \begin{bmatrix} * & Q' & Q' & * \\ Q' & * & * & * \\ * & * & * & * \\ * & * & * & Q' \end{bmatrix}, \; c_4 \begin{bmatrix} 2 \\ 2 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} * & Q' & Q' & * \\ Q' & * & * & * \\ * & * & * & * \\ * & * & * & Q' \end{bmatrix},
$$

$$
c_1 \begin{bmatrix} 3 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} * & Q' & Q' & * \\ Q' & * & * & * \\ * & * & * & * \\ * & * & * & Q' \end{bmatrix}, \; c_2 \begin{bmatrix} 2 \\ 3 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} * & * & Q' & * \\ Q' & * & * & * \\ * & * & * & * \\ * & Q' & * & Q' \end{bmatrix},
$$

$$
c_3 \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} * & * & Q' & * \\ Q' & * & * & * \\ * & * & * & * \\ * & Q' & * & Q' \end{bmatrix}, \; c_4 \begin{bmatrix} 1 \\ 3 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} * & * & Q' & * \\ Q' & * & * & * \\ * & * & * & Q' \\ * & Q' & * & * \end{bmatrix},
$$