# Heuristic search wrap-up, and local search

# Announcements

- Assignment 0 released
- Survey released (closing tonight)

# A* Search

- Best First Search with f(s) = g(s) + h(s), where:
  - g(s) = cost of best path found so far to s
  - h(s) = **admissible** heuristic function

```
1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node, FRINGE)
3. Repeat:
4.  If empty(FRINGE) then return failure
5.      s ← REMOVE(FRINGE)
6.      If GOAL?(s) then return s and/or path
7.     For every state s' in SUCC(s):
8.       INSERT(s', FRINGE)
```

# Reminder: Admissible Heuristic

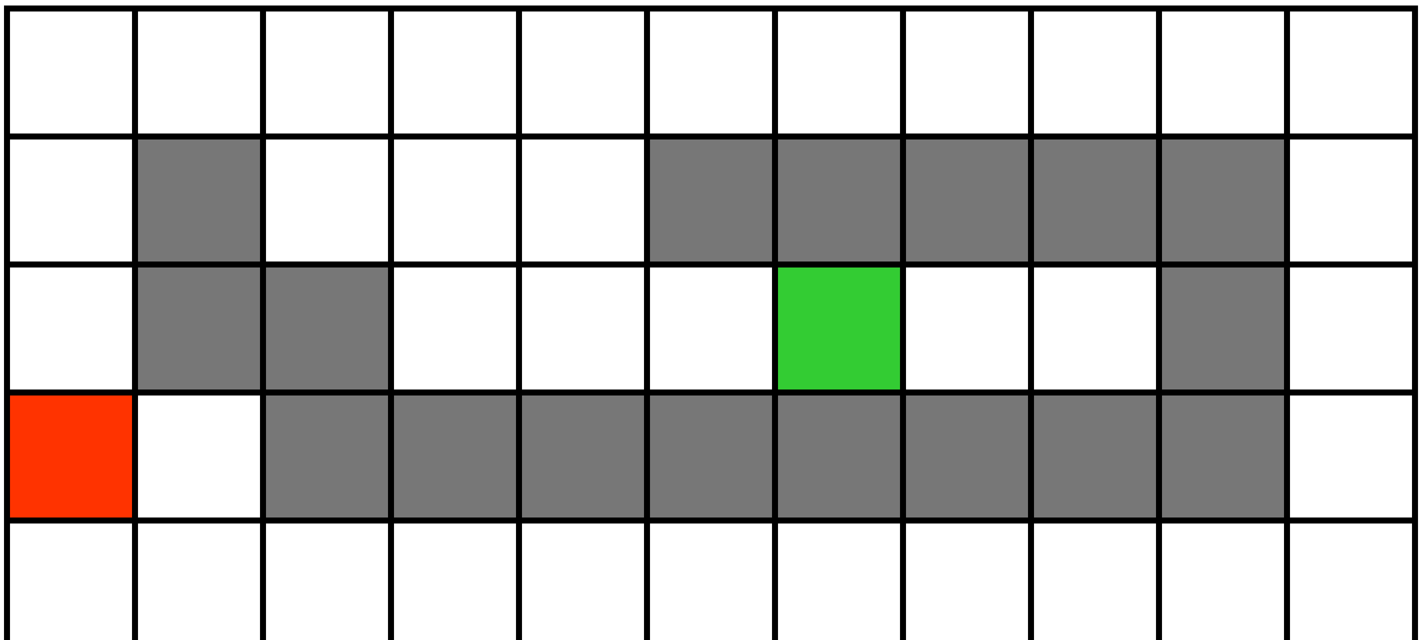
- An heuristic h(s) is **admissible** if for any state s,

$$\mathbf{0 \leq h(s) \leq h^*(s)}$$

  where h*(s) is the optimal cost from s to a goal.

- In other words, an admissible heuristic **never overestimates the cost to the goal**
  - We'll need to design h(s) so that it's always less than h*(s), even though we don't know h*(s)!

# Robot Navigation



# Robot Navigation

f(s) = h(s), with h(s) = Manhattan distance to the goal
(not A*)

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |
| 7 | | 5 | 4 | 3 | | | | | | 5 |
| 6 | | | 3 | 2 | 1 | 0 | 1 | 2 | | 4 |
| 7 | 6 | | | | | | | | | 5 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |

# Robot Navigation

f(s) = h(s), with h(s) = Manhattan distance to the goal
(not A*)

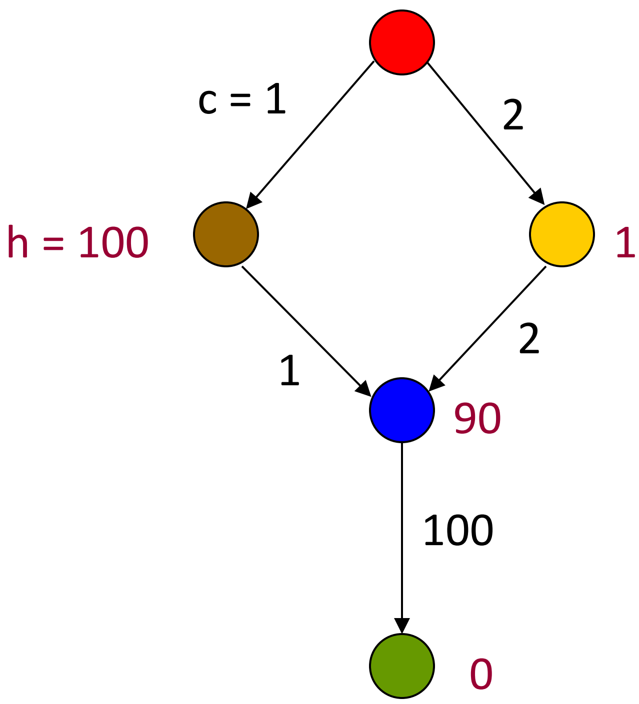
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |
| 7 | | 5 | 4 | 3 | | | | | | 5 |
| 6 | | | 3 | 2 | 1 | 0 | 1 | 2 | | 4 |
| 7 | 6 | | | | | | | | | 5 |
| 8 | 7 | 6 | 5 | 4 | 3 | 2 | 3 | 4 | 5 | 6 |

# Robot Navigation

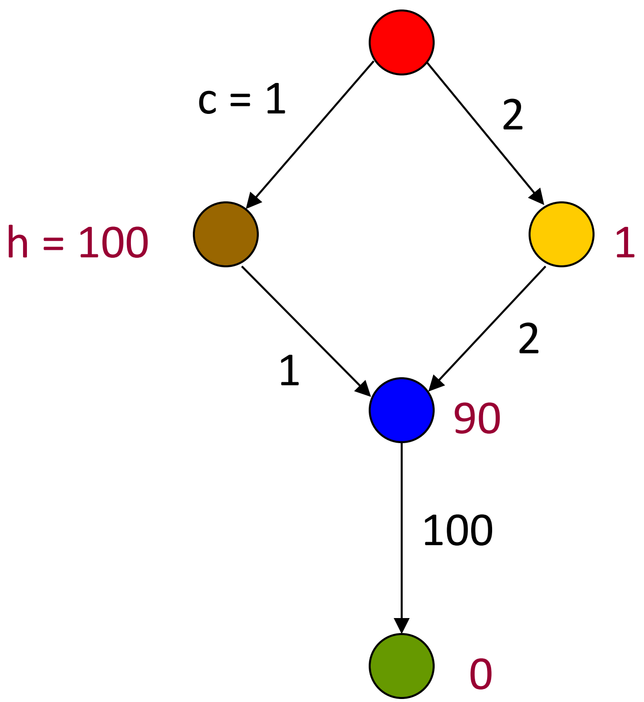
f(s) = g(s)+h(s), with h(s) = Manhattan distance to goal (A*)

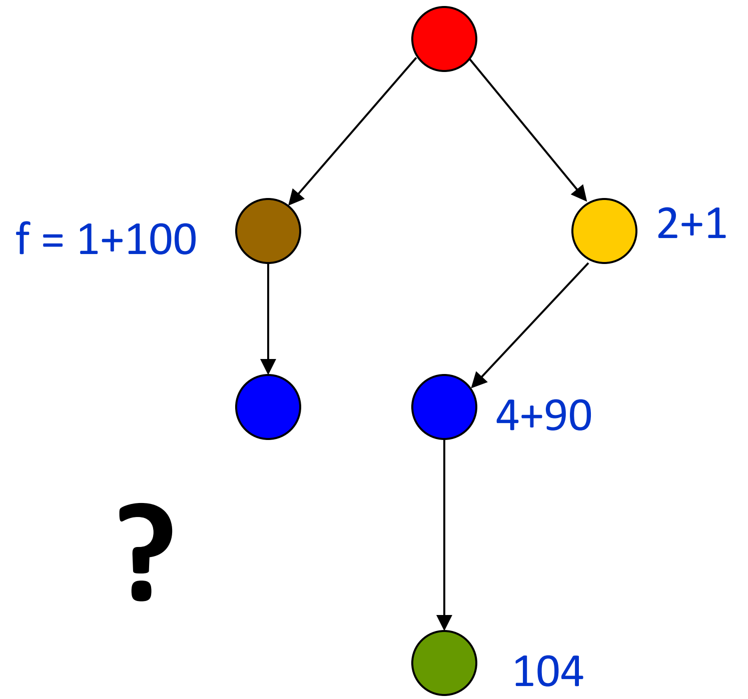
| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 8+3 | 7+4 | 6+5 | 5+6 | 4+7 | 3+8 | 2+9 | 3+10 | 4 | 5 | 6 |
| 7+2 | | 5+6 | 4+7 | 3+8 | | | | | | 5 |
| 6+1 | | | 3 | 2+9 | 1+10 | 0+11 | 1 | 2 | | 4 |
| 7+0 | 6+1 | | | | | | | | | 5 |
| 8+1 | 7+2 | 6+3 | 5+4 | 4+5 | 3+6 | 2+7 | 3+8 | 4 | 5 | 6 |

# What to do with revisited states?



# What to do with revisited states?




If we discard this new node, A* expands the goal next, returning a non-optimal solution

1. Is A* complete?

2. Is A* optimal?

3. What is the running time of A*?

4. What are the memory requirements of A*?

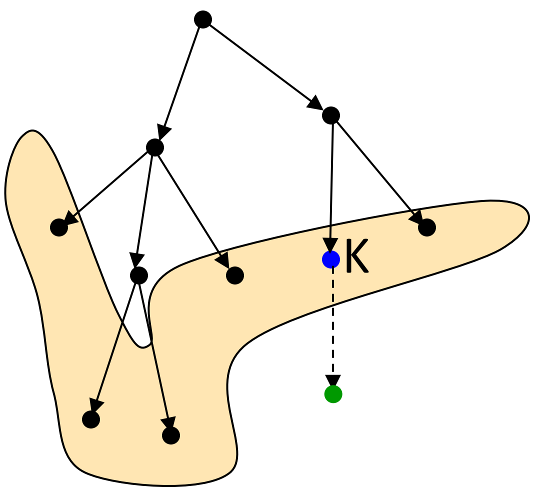
# A* is complete and optimal if:

- Duplicate states are revisited, and
- h is admissible.
- Proof sketch:
  - First show that if a solution exists, A* terminates and finds a solution.
  - Then show that whenever A* expands a node, the path to that node is optimal.

# A* is complete and optimal if h is admissible.

Proof sketch:

- **If a solution exists, A* terminates and returns a solution**

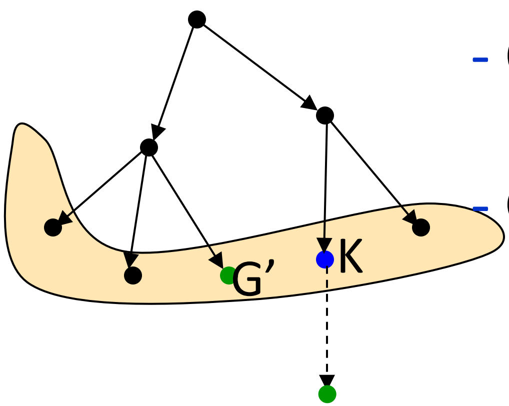



- As long as A* hasn't terminated, a node K on the fringe lies on a solution path
- Since each node expansion increases the length of one path, K will eventually be selected for expansion, unless a solution is found along another path

# A* is complete and optimal if h is admissible.

Proof sketch:

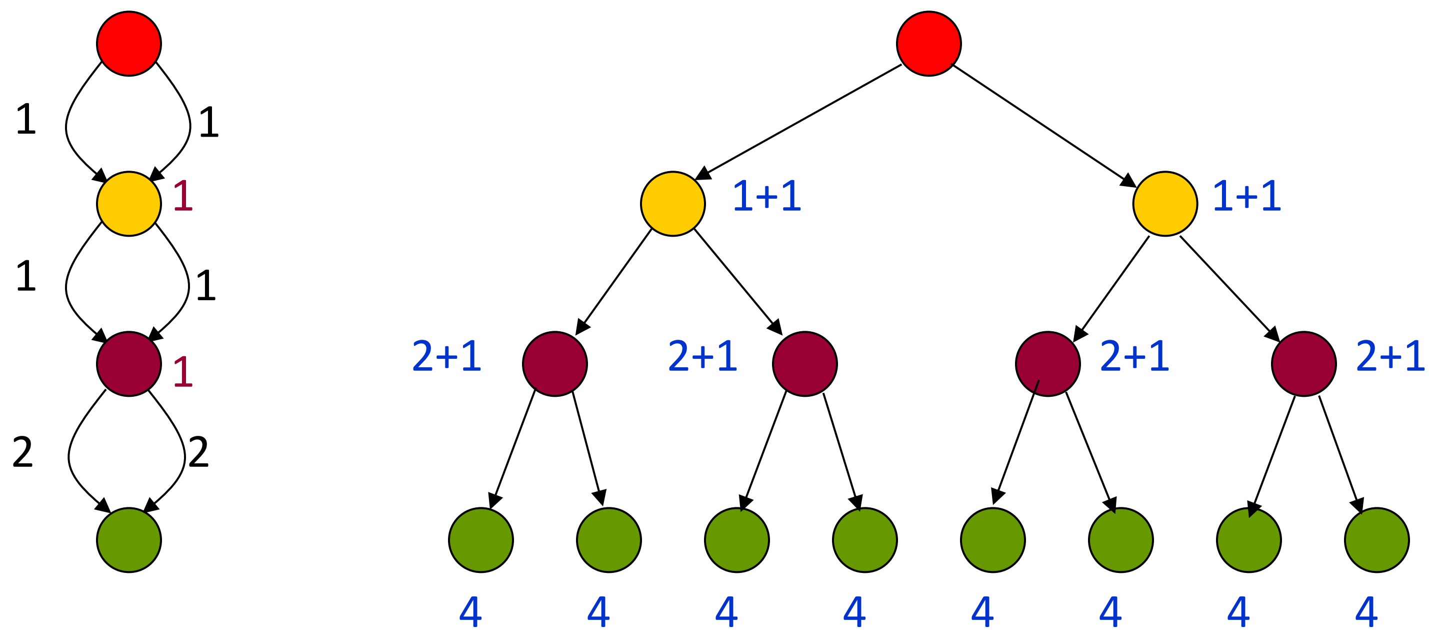
- **Whenever A* chooses to expand a goal node, the path to this node is optimal**




- C*= cost of the optimal solution path
- G’: non-optimal goal node in the fringe
  f(G’) = g(G’) + h(G’) = g(G’) > C*
- A node K in the fringe lies on an optimal path:
  f(K) = g(K) + h(K) $\leq$ C*
- So, G’ will not be selected for expansion

# Complexity of A*

- A* expands all nodes with f(s) < C*
  - May also expand non-goal states with f(s) = C*
  - May be an exponential number of nodes unless the heuristic is sufficiently *accurate*

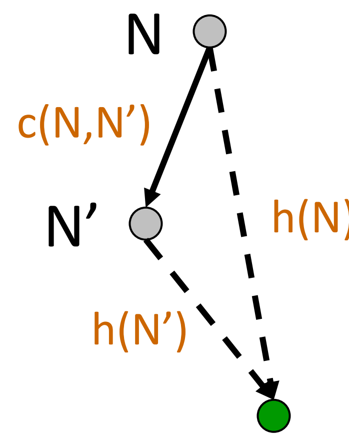


# Consistent Heuristic

- An admissible heuristic h is consistent (or monotone) if for each node N and each child N' of N:

$$h(N) \leq c(N,N') + h(N')$$

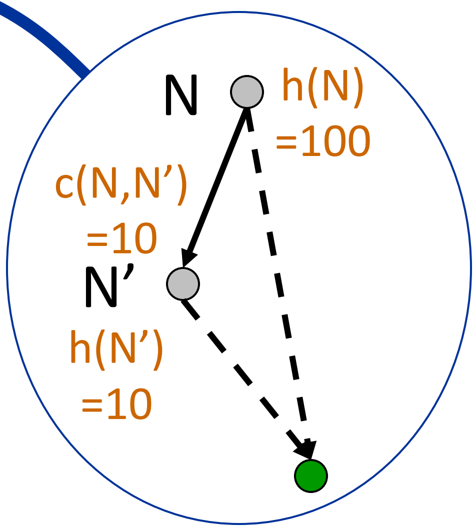
(triangle inequality)




$$\text{Or: } h(N) - h(N') \leq c(N,N')$$

# Consistency Violation

If h says that N is 100 units from the goal, then moving from N along an edge costing 10 units should **not** lead to a node N' that h estimates to be 10 units away from the goal




(triangle inequality)

**Should satisfy: $h(N) - h(N') \leq c(N,N')$**

# 8-Puzzle

| | | |
|---|---|---|
| 5 | | 8 |
| 4 | 2 | 1 |
| 7 | 3 | 6 |

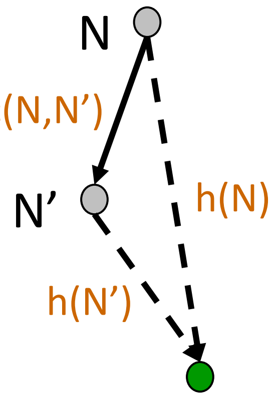
STATE(N)

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

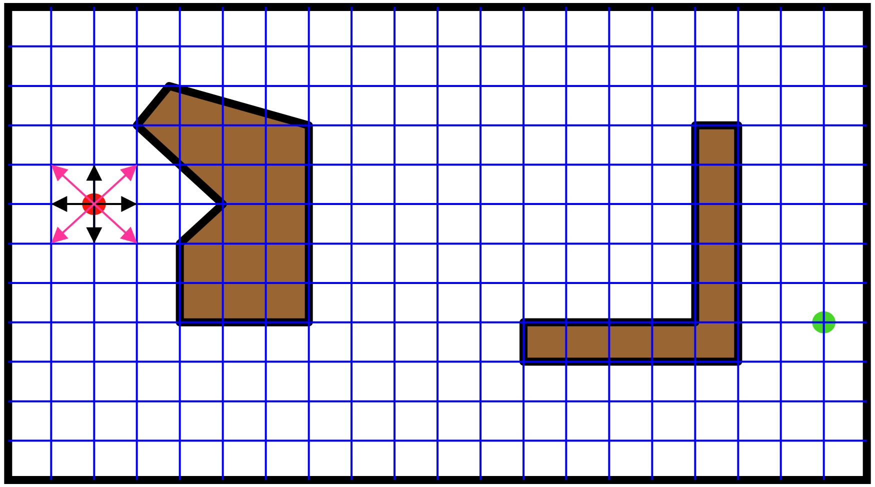
goal

- $h_1(N)$ = number of misplaced tiles
- $h_2(N)$ = sum of the (Manhattan) distances of every tile to its goal position

# Robot Navigation




$h(N) \leq c(N,N') + h(N')$



Cost of one horizontal/vertical step = 1

Cost of one diagonal step = $\sqrt{2}$

$$h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$$

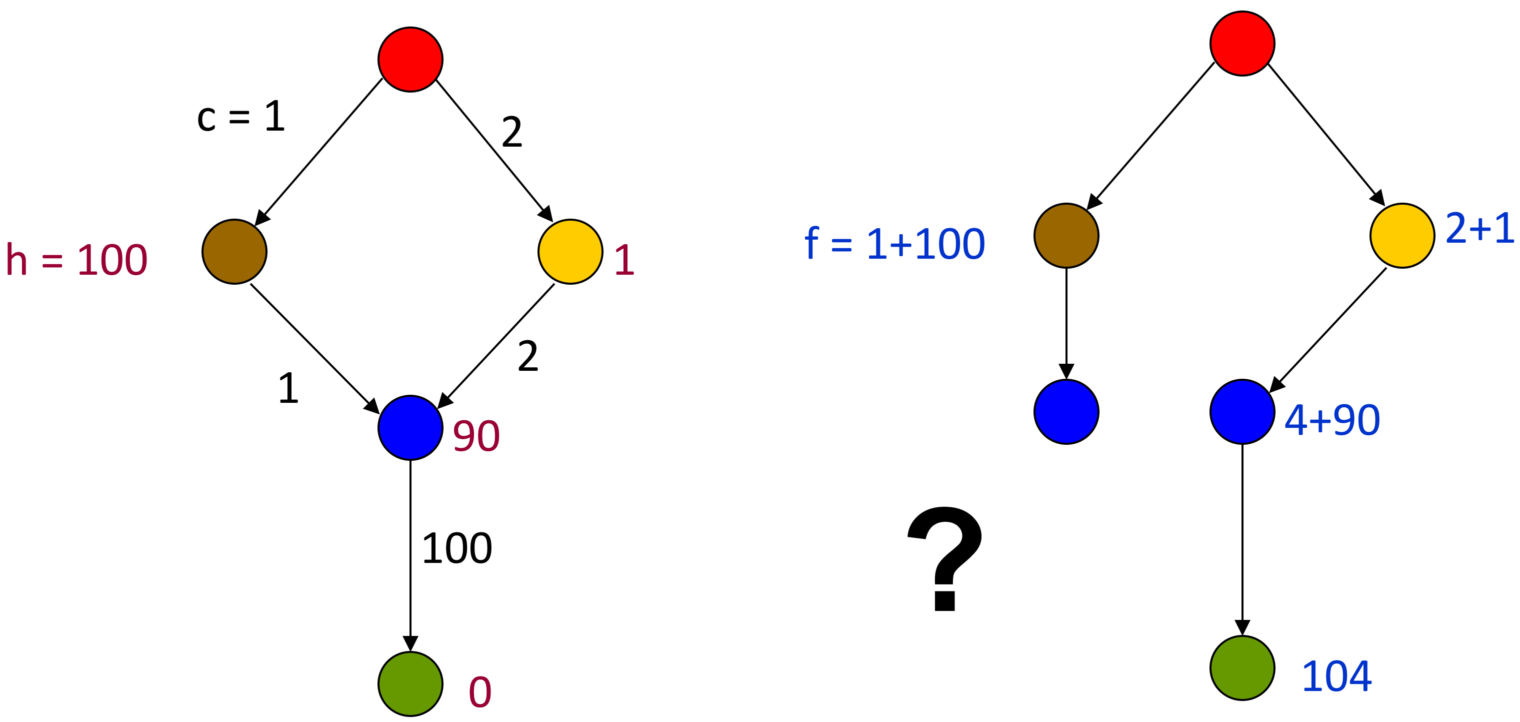
$$h_2(N) = |x_N - x_g| + |y_N - y_g|$$

# Admissibility and Consistency

- A consistent heuristic is also admissible

- An admissible heuristic may not be consistent, but many admissible heuristics are consistent

# Revisiting - Is h(.) consistent?



# Updated Algorithm

1. If GOAL?(initial-state) then return initial-state
2. INSERT(initial-node, FRINGE)
3. Repeat:
4. If empty(FRINGE) then return failure
5. s ← REMOVE(FRINGE)
6. INSERT(s, CLOSED)
7. If GOAL?(s) then return s and/or path
8. For every state s’ in SUCC(s):
9. If s’ in CLOSED, discard s’
10. If s’ in FRINGE with larger s’, remove from FRINGE
11. If s’ not in FRINGE, INSERT(s’, FRINGE)

# A* is optimal if

- h is **admissible** (but not necessarily consistent)
  - Revisited states not discarded

```
If GOAL?(initial-state) then return initial-state
INSERT(initial-node, FRINGE)
Repeat:
 If empty(FRINGE) then return failure
    s ← REMOVE(FRINGE)
    If GOAL?(s) then return s and/or path
   For every state s' in SUCC(s):
      INSERT(s', FRINGE)
```

- h is **consistent**
  - (Many) revisited states discarded:

```
If GOAL?(initial-state) then return initial-state
INSERT(initial-node, FRINGE)
Repeat:
 If empty(FRINGE) then return failure
 s ←  REMOVE(FRINGE)
 INSERT(s, CLOSED)
 If GOAL?(s) then return s and/or path
 For every state s' in SUCC(s):
    If s' in CLOSED, discard s'
    If s' in FRINGE with larger s', remove from FRINGE
    If s' not in FRINGE, INSERT(s', FRINGE)
```

# Heuristic Accuracy

- If $h_1$ and $h_2$ are consistent heuristics such that $h_1(N) \leq h_2(N)$ for all nodes N, then $h_2$ is more **accurate** (or **informative**) than $h_1$
  - The more accurate h is, the less work A* has to do!

| | | |
|---|---|---|
| 5 | | 8 |
| 4 | 2 | 1 |
| 7 | 3 | 6 |

STATE(N)

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

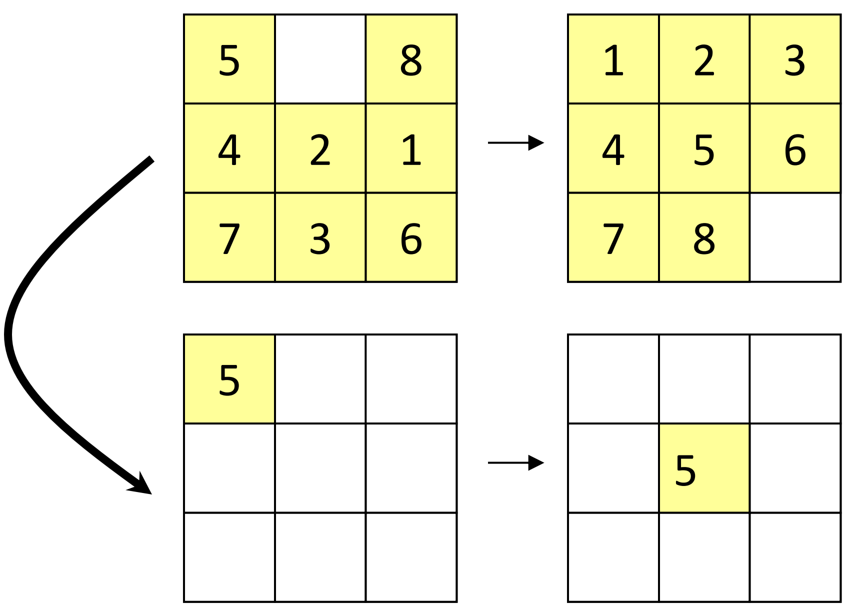
Goal state

- $h_1(N)$ = number of misplaced tiles
- $h_2(N)$ = sum of distances of every tile to its goal position

- Which is more accurate?

# How to create good heuristics?

- One approach: solve "relaxed" problems that ignore some constraints
  - E.g. ignore interactions among parts of the problem
  - In the 8-puzzle, the sum of the distances of each tile to its goal position ($h_2$) corresponds to solving 8 simple problems:



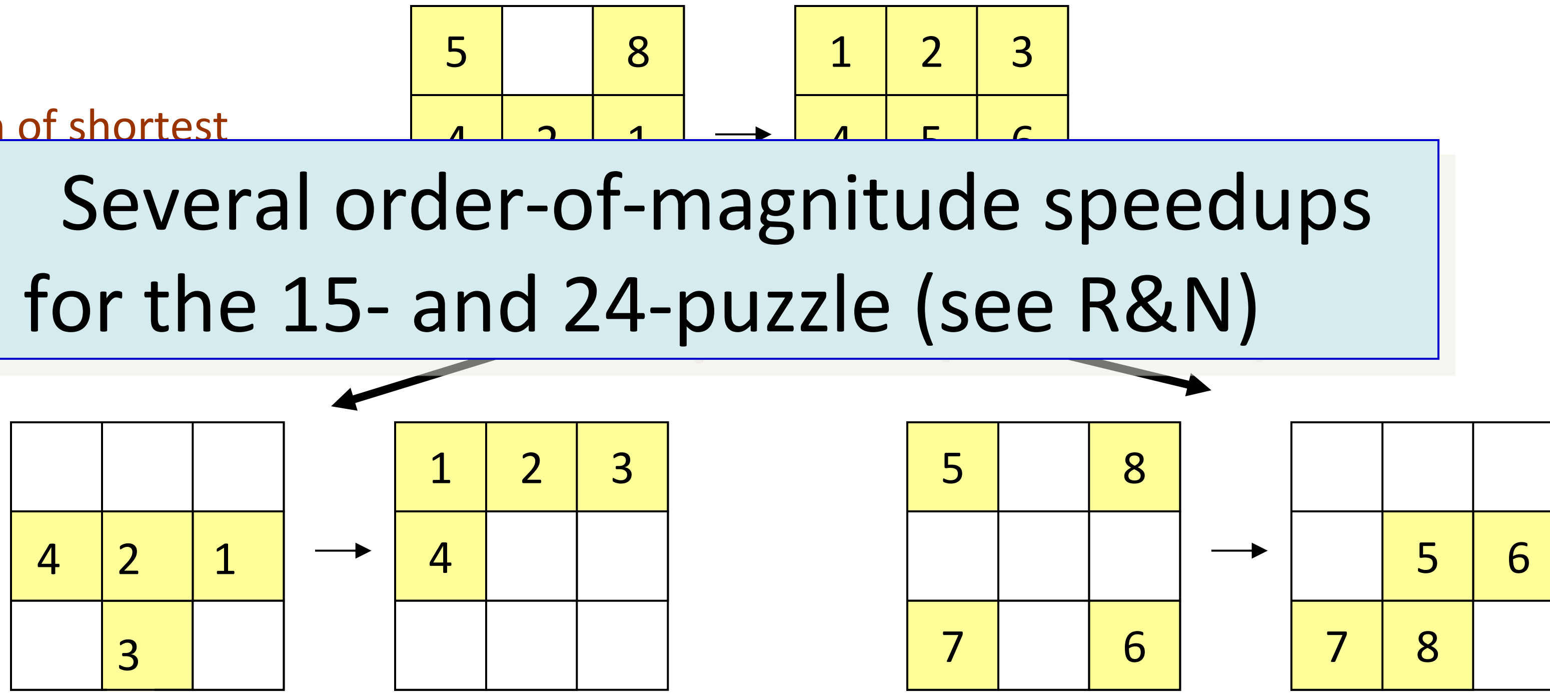

$d_i$ is length of shortest path to move tile i to its goal position, ignoring the other tiles, and

$h_2 = \Sigma_{i=1,\ldots 8}\, d_i$

# Can we do better?

- For example, we could consider two more complex relaxed problems:

$d_{1234}$ = length of shortest
path
to go
Ignor




→ Several order-of-magnitude speedups for the 15- and 24-puzzle (see R&N)

- → $h = d_{1234} + d_{5678}$ [disjoint pattern heuristic]
- These distances can be pre-computed and stored

[Each requires generating a tree of 3,024 states (breadth-first search)]

# Experimental Results

(see R&N for details)

- Random 8-puzzles with:
  - $h_1$ = number of misplaced tiles
  - $h_2$ = sum of distances of tiles to their goal positions
- Average "effective branching factors" based on actual # of nodes expanded:

| d | IDS | $A_1$* | $A_2$* |
|---|---|---|---|
| 2 | 2.45 | 1.79 | 1.79 |
| 6 | 2.73 | 1.34 | 1.30 |
| 12 | 2.78 (3,644,035) | 1.42 (227) | 1.24 (73) |
| 16 | -- | 1.45 | 1.25 |
| 20 | -- | 1.47 | 1.27 |
| 24 | -- | 1.48 (39,135) | 1.26 (1,641) |

# Next class

- Local search
- Start with adversarial search and game playing