Informed Search

- Reading: Russell and Norvig, ch. 4.1-4.2
  - The material here is an edited version of Russell's slides with some different examples
- Best-first search
- A* search

Pseudocode for tree search

- Search strategy affects which node is expanded next

```
// edited to look a bit more object-oriented
function tree_search(problem, fringe) returns a solution or failure
  fringe.insert(make_node(problem.initial_state()))
  loop do
    if fringe is empty return failure
    node ← remove_front(fringe)
    if problem.goal_test(node.state()) return node
    fringe.insert_all(node.expand(problem))
```

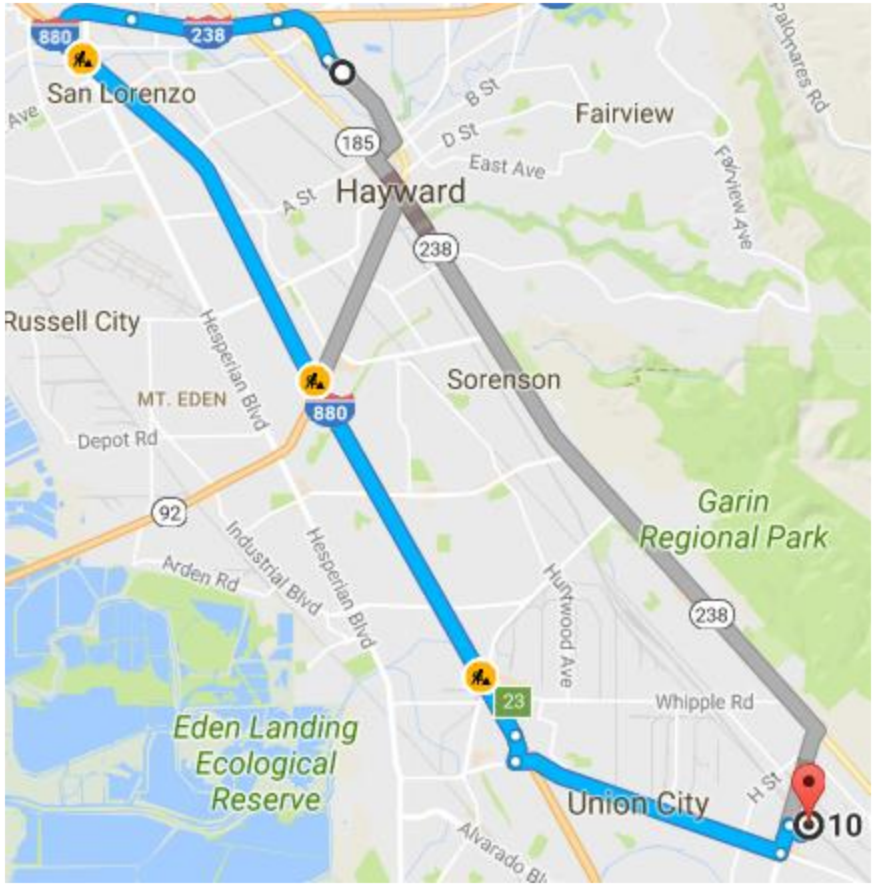
Best-first search

- Estimate how likely a node is to lead to the optimal solution using an evaluation function *h(n)*
  - “Estimate” – evaluation function is a *heuristic*
  - Implement fringe as priority queue ordered by evaluation function
  - Example of a greedy search
- Ex: for travel, distance between node’s location and destination (if you could fly between the 2 points)
  - Is the travel time computed by google maps exact?

Best-first search example

- Travel from 21500 Foothill Blvd, Hayward to Union City BART station (10 Union Square, Union City)
- Could first head towards DNA Motor Lab at 21739 Mission Blvd or 238 North on-ramp (close to 21300 Foothill Blvd)
  - If “best” choice is estimated from physical distance heuristic, choose DNA
  - This route only uses local roads, so it is actually slower (usually)
    - When would physical distance be a better heuristic?

Maps for best-first example



Best-first search

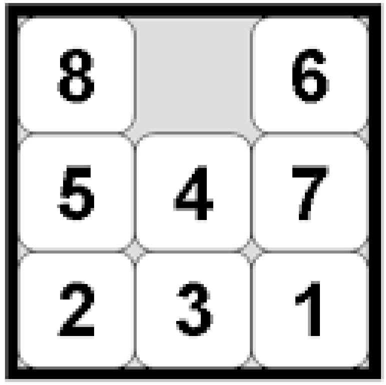
- Complete?
  - How might we end up in a loop? (Consider what might be the cause of this reported bug: https://steamcommunity.com/app/402310/discussions/2/405693392926963728/)
  - Yes if in a finite state space and checking for duplicate states
- Time, space
  - Both $O(b^m)$ – may end up keeping all nodes in memory, choice of heuristic has a significant impact on time and space
- Not optimal

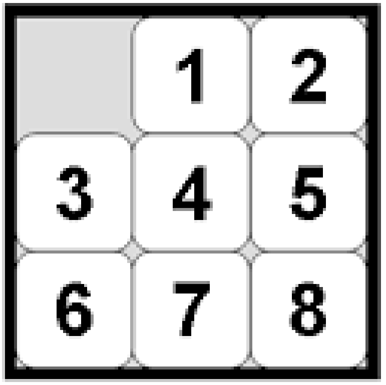
A* search

- Evaluation function $f(n) = g(n) + h(n)$
- $g(n)$ = cost to reach node n
  - reduce chance of expanding already expensive path
- $h(n)$ = estimated cost to reach node with goal state (“goal node”)
  - $h(n) \leq h^*(n)$, $h^*(n)$ = actual cost to reach goal node
    - This property makes $h(n)$ an *admissible* heuristic
  - $h(n) \geq 0$
  - $h(n) = 0$ for any goal node

Examples of admissible heuristics

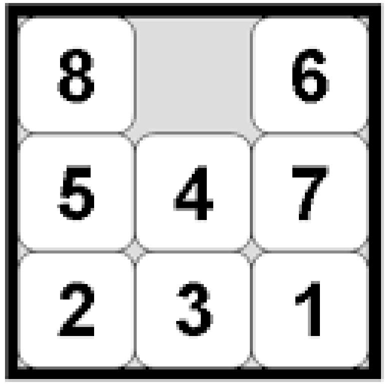
- 8-puzzle
  - Number of tiles in the wrong position
  - Total “Manhattan” distance of tiles from their correct positions
  - Compute each for the following example from Gerhard Wickler (http://www.aiai.ed.ac.uk/~gwickler/eightpuzzle-inf.html), goal state on right

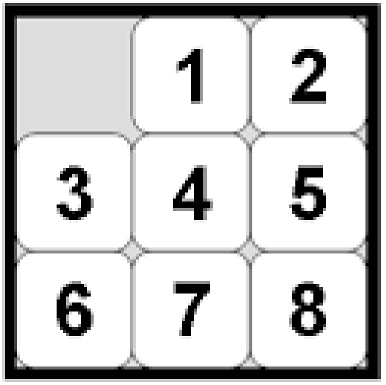




Examples of admissible heuristics

- Number of tiles in the wrong position = 7
  - Only the 4 is in the right place
- Total Manhattan distance (for tiles 1, 2, 3, …)
  - $f = 3 + 4 + 2 + 0 + 2 + 4 + 2 + 4 = 21$





Dominance

- If heuristic $h_2(n) \geq h_1(n)$ for all n, then h_2 *dominates* h_1 and is better for search
- Time/space complexity is still exponential, but can be much faster/use much less memory
- Russell's sample numbers of nodes for the 8-puzzle problem:

Depth of solution	**Iterative deepening**	**A* using # misplaced tiles**	**A* using Manhattan distance**
14	3.5M	539	113
24	54G	39K	1.6K

Developing admissible heuristics

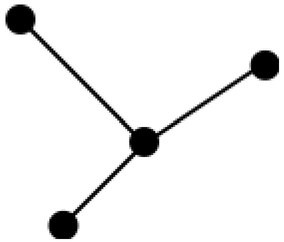
- Useful approach: find exact solution for relaxed version of problem
- For 8-puzzle:
  - If allowed to move tiles anywhere, # misplaced tiles is exact
  - If allowed to move tile to any adjacent square, Manhattan distance is exact
- Requirement: cost of optimal solution of relaxed problem ≤ cost of optimal solution of real problem

Consistency

- A heuristic h is consistent if for any node n and its possible successor nodes n'
  - $h(n) \leq$ cost of action to go to node n' + $h(n')$
  - So heuristic always underestimates cost
- If a heuristic is consistent, it will also be admissible, but not necessarily vice versa

Relaxed problem examples

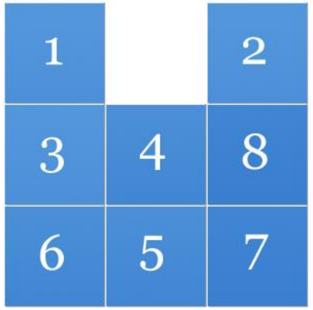
- Another example: Traveling Salesman Problem – for N cities, find shortest sequence that visits each city exactly once
  - Classic NP-Complete problem – best known algorithm takes exponential time
  - Minimum spanning tree problem as relaxed version of problem
    - Selects edges of graph that connect all vertices with minimum cost
      - May not be possible to traverse edges without revisiting vertex



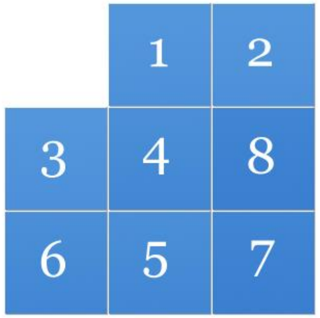
    - Solvable in $O(n^2)$

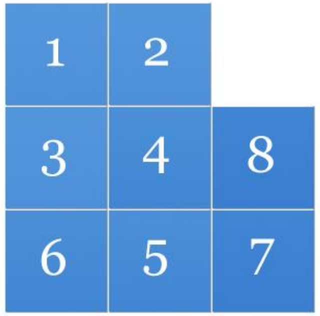
A* example (uses Wickler’s app for order of expansion)

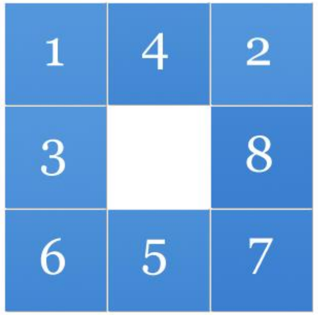
- Start at node A: (screenshots use http://mypuzzle.org/sliding) (f = 5)



- Possible successor nodes B, C, D with states: (f = 5, 7, 7 – includes 1







A* example

- Expand B to get node E with state: (f = 7) (duplicates detected)

3	1	2
	4	8
6	5	7

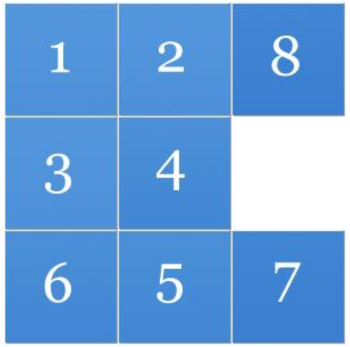
- Expand this node to get nodes F, G: (f = 9, 9)

3	1	2
4		8
6	5	7

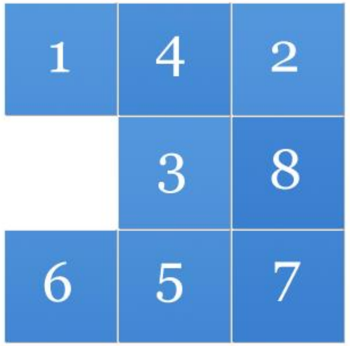
3	1	2
6	4	8
	5	7

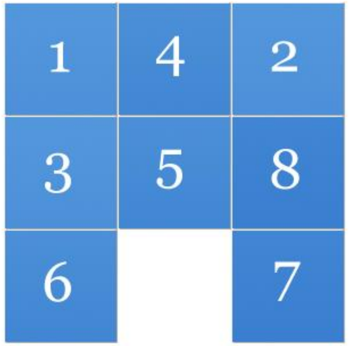
A* example

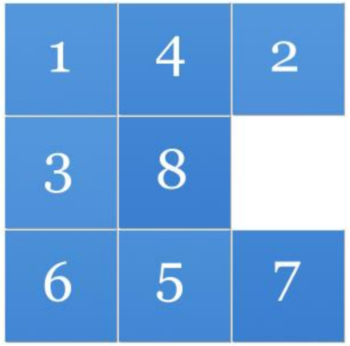
- Expand C to get node H: (*f* = 9)



- Expand D to get nodes I, J, K: (*f* = 9, 7, 9)

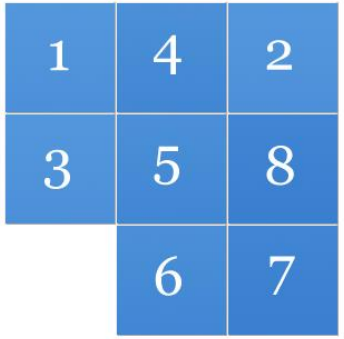


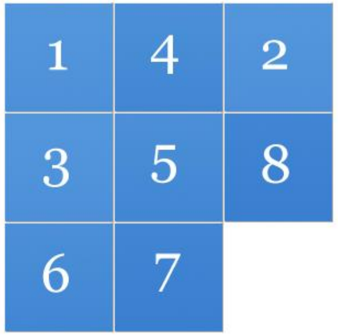




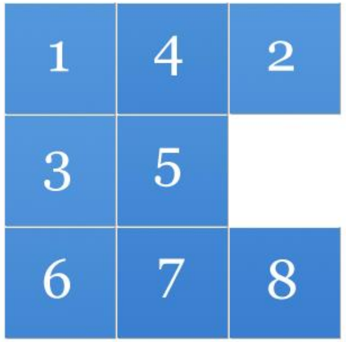
A* example

- Expand J to get nodes L, M: (f = 9, 7)



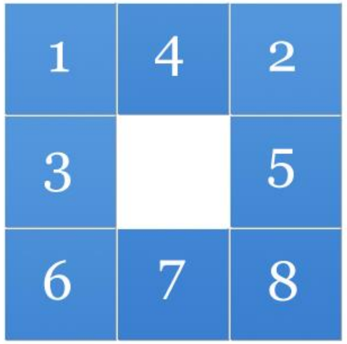


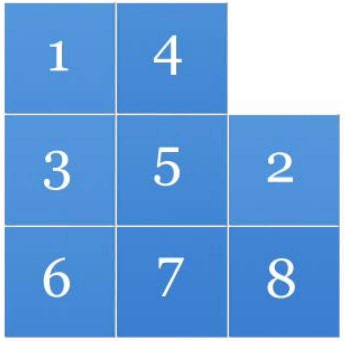
- Expand M to get node N: (f = 7)



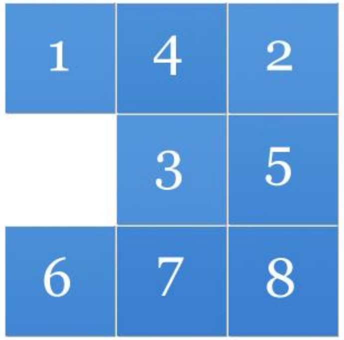
A* example

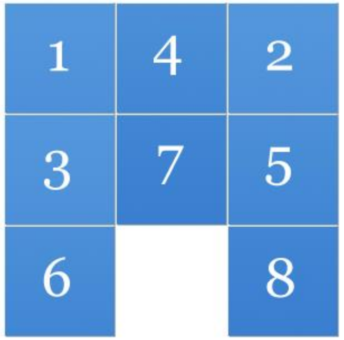
- Expand N to get nodes O, P: (f = 7, 9)

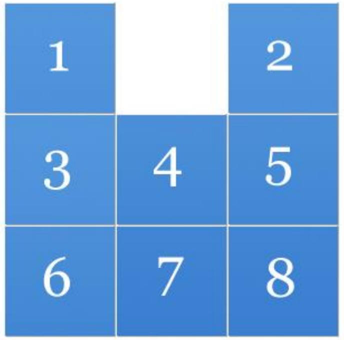




- Expand O to get nodes Q, R, S: (f = 9, 9, 7)

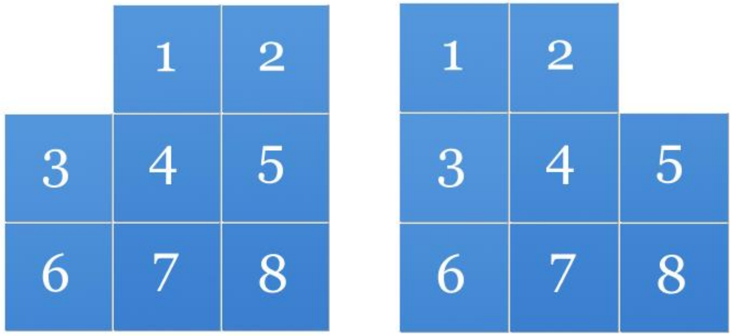






A* example

- Expand S to get nodes T, U: (*f* = 7, 9)



- Node T has the goal state, so done!

A* search optimality

- Standard proof:
  - Suppose a suboptimal node G has been generated and is placed on the queue
  - G will not be expanded ahead of any node n on the path to the optimal solution node G_{opt}
    - $f(G) = g(G)$ because G is a goal node
    - $g(G) > g(G_{opt})$ because G is not optimal
    - $g(G_{opt}) \geq f(n)$ because h is admissible
    - So $f(G) > f(n)$ and n will be expanded before G

A* search optimality

- Alternative explanation
  - similar to breadth-first, but expands in “contours” with the same *f* value
  - In above example:
    - *f* = 5: nodes A, B
    - *f* = 7: nodes C, D, E, J, M, N, O, S, T
    - <u>Not expanded</u>: *f* = 9: nodes F, G, H, I, K, L, P, Q, R, U

A* search evaluation

- Complete
  - Yes, if the # nodes *n* where $f(n) < f(G)$, G = optimal goal node
- Time, space complexity: still exponential
  - Affected by accuracy of $h(n)$ and # nodes with sufficiently low $f(n)$
- Optimal: yes, A* expands
  - All nodes where $f(n) < C^*$, $C^* = f(G)$
  - Some nodes where $f(n) = C^*$,
  - No nodes where $f(n) > C^*$,