Informed Search

- Blind search - no notion concept of the "right direction"
  - can only recognize goal once it's achieved
- Heuristic search – we have rough idea of how good various states are, and use this knowledge to guide our search
- Can find solutions more efficient than uninformed
- General approach is *best-first-search*
- A node is selected based on an *evaluation function f(n)*
- A node that **seems** to be best is picked and it may not be the actual best

Best First Search

- The Idea:
  - use an *evaluation function* for each node... estimate of ``desirability''
  - Expand most desirable unexpanded node

Implementation

- Fringe: is a queue sorted in decreasing order of desirability

Special cases

Greedy

A*

Cost function $f(n)$

- A function f is maintained for each node

$f(n) = g(n) + h(n)$, n is the node in the open list

- "Node chosen" for expansion is the one with least f value
- $g(n)$ is the cost from root S to node n
- $h(n)$ is the estimated cost from node n to a goal
- For BFS: $f = 0$,
- For DFS: $f = 0$,
- For greedy g =0

Greedy search

- Expands a node it sees closest to a the goal
- $f(n) = h(n)$
- Resembles DFS in that it prefers to follow a single path all the way to the goal
- Also suffers from the same defects of DFS, it may stuck in a loop i.e. not complete As well as it is not optimal.

Hill climbing

This is a *greedy* algorithm

Expands a node it sees closest to a goal

f(n) =h(n)

The algorithm

select a heuristic function;

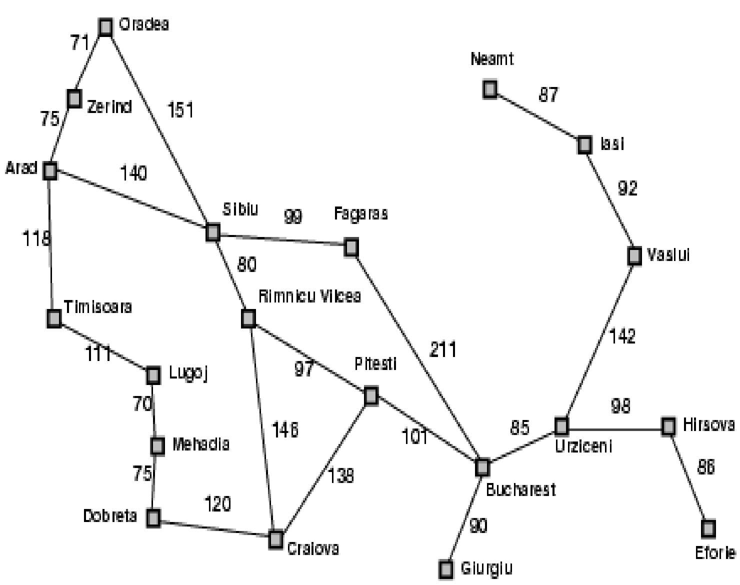
set C, the current node, to the highest-valued initial node;

Loop until success or no more children(fail)

select N, the highest-value child of C;

return C if its value is better than the value of N;

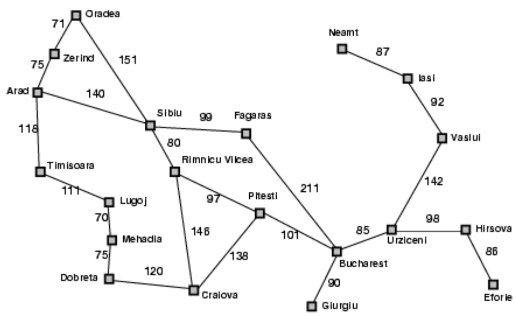
Hill Climbing search example



Straight–line distance to Bucharest

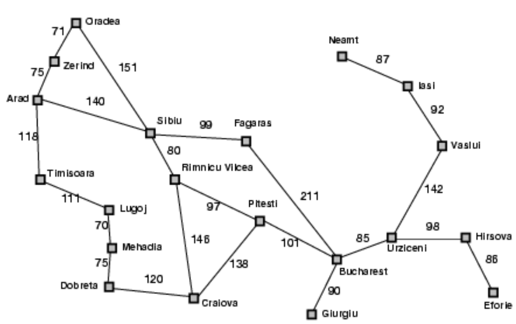
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Hill Climbing search example



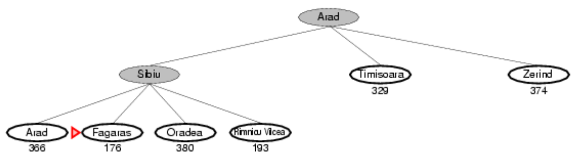
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

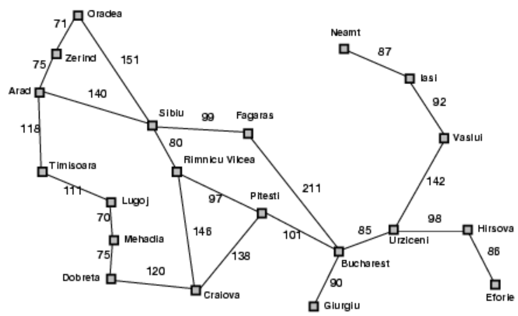
Hill Climbing search example



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Hill Climbing search example

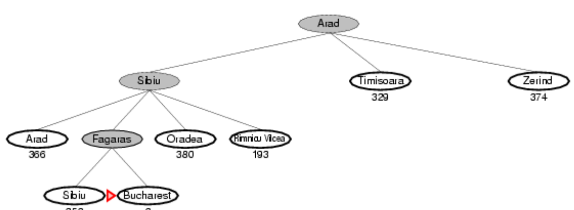


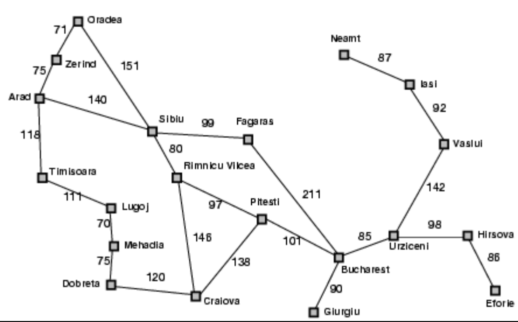


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Hill Climbing search example





Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Hill climbing

Complete:

No, Can get stuck in loop. Complete if loops are avoided.

Time complexity?

O(b^m), but with some good heuristic, it could give better results

Space complexity?

O(b^m), keeps all nodes in memory

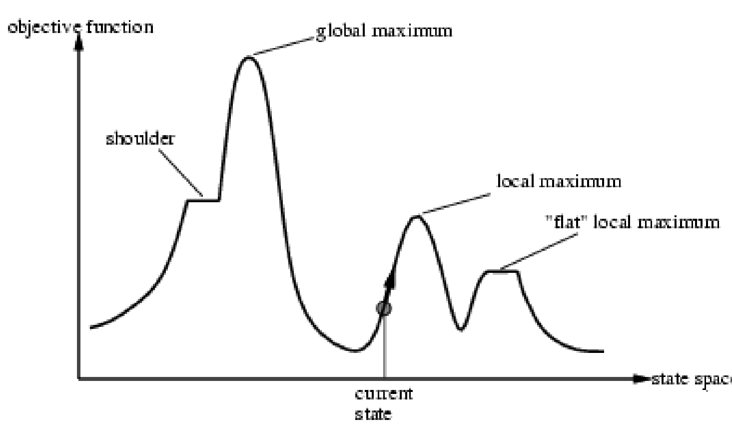
Optimality?

No

e.g. Arad→Sibiu→Rimnicu Virea→Pitesti→Bucharest is shorter!

Hill-climbing search

- Problem: depending on initial state, can get stuck in local maxima,…etc



Problems with hill climbing

1. Local maximum problem: there is a peak, but it is lower than the highest peak in the whole space.
2. The plateau problem: all local moves are equally unpromising, and all peaks seem far away.
3. The ridge problem: almost every move takes us down.

Solution:

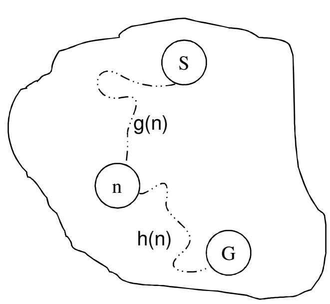
Random-restart hill climbing is a series of hill-climbing searches with a randomly selected start node whenever the current search gets stuck.

Algorithm A*

- One of the most important advances in AI search algs.
- Idea: avoid expanding paths that are already expensive

$$f(n) = g(n) + h(n)$$

- $g(n)$ = least cost path to n from S found so far
- $h(n)$ = estimated cost to goal from n
- $f(n)$ = estimated total cost of path through n to goal



The A* procedure

Hill-climbing (and its improved versions) may miss an optimal solution. Here is a search method that ensures **optimality** of the solution.

The algorithm

keep a list of partial paths (initially root to root, length 0);

repeat

succeed if the first path P reaches the goal node;

otherwise remove path P from the list;

extend P in all possible ways, add new paths to the list;

sort the list by the sum of two values: the real cost of P till now, and an estimate of the remaining distance;

prune the list by leaving only the shortest path for each node reached so far;

until

success or the list of paths becomes empty;

The A* procedure

A heuristic that never overestimates is also called **optimistic** or **admissible**.

We consider three functions with values ≥ 0:

- g(n) is the actual cost of reaching node n,
- h(n) is the actual *unknown* remaining cost,
- h*(n) is the optimistic estimate of h(n).

Admissible heuristics

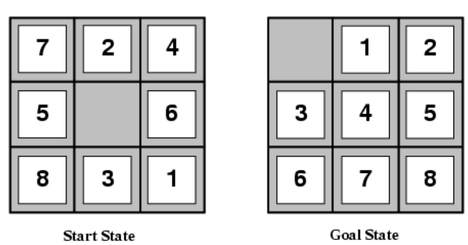
- A heuristic $h(n)$ is admissible if for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.
- An admissible heuristic never overestimates the cost to reach the goal, i.e., it is optimistic
- Theorem: If $h(n)$ is admissible, A^* using is optimal

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



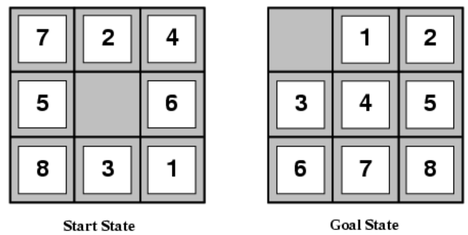
- $h_1(S) = ?$
- $h_2(S) = ?$

Admissible heuristics

E.g., for the 8-puzzle:

- $h_1(n)$ = number of misplaced tiles
- $h_2(n)$ = total Manhattan distance

(i.e., no. of squares from desired location of each tile)



- $h_1(S) = ?$ 8
- $h_2(S) = ?$ 3+1+2+2+2+3+3+2 = 18

Admissible heuristics

- If $h_2(n) >= h_1(n)$ for all n, both are admissible
- Then h_2 dominates h_1 and is usually better for search

Typical Costs

- d = 14 IDS = 3,473,941 nodes
  - A*(h_1) = 539 nodes
  - A*(h_2) = 113 nodes
- d = 24 IDS ~ 54,000,000,000 nodes
  - A(h_1) = 39,135 nodes
  - A(h_2) = 1,641 nodes

Remark: Given h_1 and h_2 any two admissible functions

then $h(n) = \max\{h_1(n), h_2(n)\}$ is also admissible

Admissible heuristics and relaxed problems

- Admissible heuristics can be derived from the exact solution cost of a relaxed version of the problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere,
- then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution
- Remark:
  - the optimal solution cost of a relaxed problem is less than the optimal solution cost of the real problem

A* Algorithm- Properties

- **Admissibility**: An algorithm is called admissible if it always terminates and terminates in optimal path
- **Theorem**: A* is admissible.
- **Lemma**: Any time before A* terminates there exists on *OL* a node *n* such that $f(n) <= f^*(s)$
- **Observation**: For optimal path $s \rightarrow n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow g$,
  1. $h^*(g) = 0, g^*(s)=0$ and
  2. $f^*(s) = f^*(n_1) = f^*(n_2) = f^*(n_3)\ldots = f^*(g)$

Algorithm A*

- $f^*(n) = g^*(n) + h^*(n)$, where,
- $g^*(n)$ = actual cost of the optimal path (s, n)
- $h^*(n)$ = actual cost of optimal path (n, g)
- $g(n) \leq g^*(n)$
- By definition, $h(n) \leq h^*(n)$
- *h(n) <= h*(n) where h*(n) is the actual cost of optimal path to G(node to be found) from n*

Lemma

Any time before A* terminates there exists in the open list a node n' such that $f(n') <= f^*(S)$

For any node n_i on optimal path,

$f(n_i) = g(n_i) + h(n_i)$

$<= g^*(n_i) + h^*(n_i)$

Also $f^*(n_i) = f^*(S)$

Let n' be the fist node in the optimal path that is in OL. Since all parents of n' have gone to CL,

$g(n') = g^*(n')$ and $h(n') <= h^*(n')$

$=> f(n') <= f^*(S)$

A* always terminates

Proof

If A* does not terminate

Let e be the least cost of all arcs in the search graph.

Then $g(n) >= e.l(n)$ where $l(n)$ = # of arcs in the path from S to n found so far. If A* does not terminate, $g(n)$ and hence $f(n) = g(n) + h(n)$ $[h(n) >= 0]$ will become unbounded.

This is not consistent with the lemma. So A* has to terminate.

Admissibility of A*

The path formed by A* is optimal when it has terminated

Proof

Suppose the path formed is not optimal

Let G be expanded in a non-optimal path.

At the point of expansion of G,

$f(G) = g(G) + h(G)$

$= g(G) + 0$

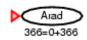
$> g^*(G) = g^*(S) + h^*(S)$

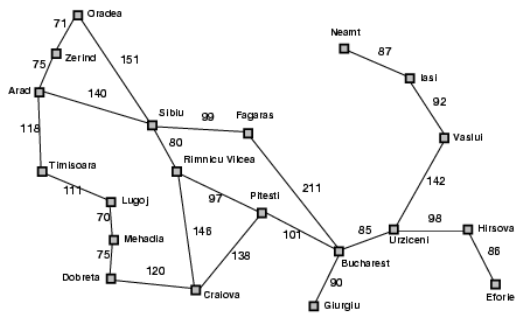
$= f^*(S)$ [$f^*(S)$ = cost of optimal path]

This is a contradiction

So path should be optimal

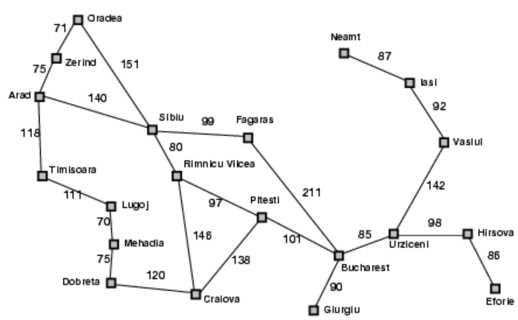
A* search example





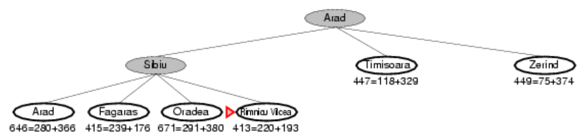
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

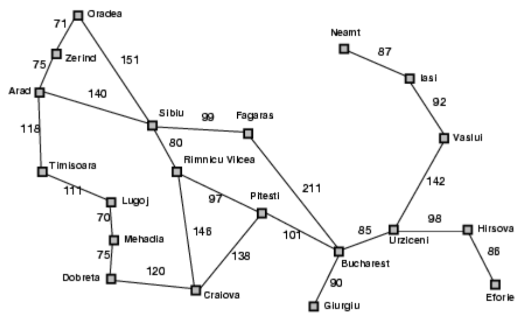
A* search example



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

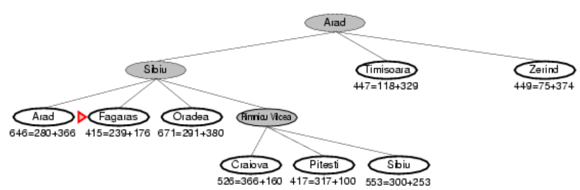


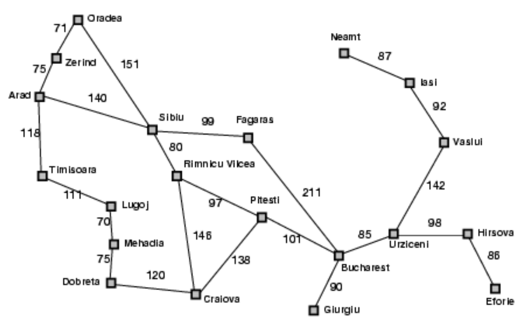


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example

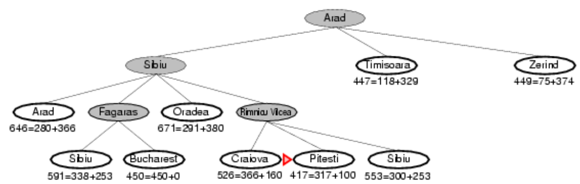


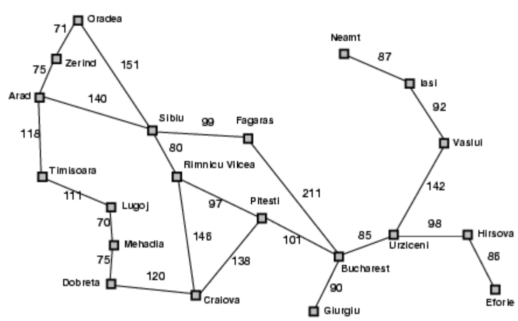


Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

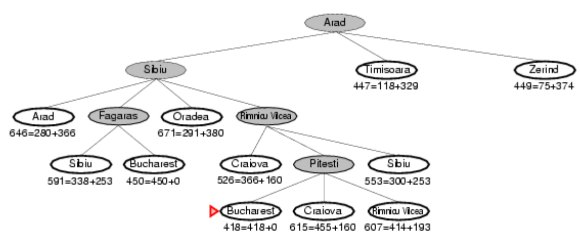
A* search example

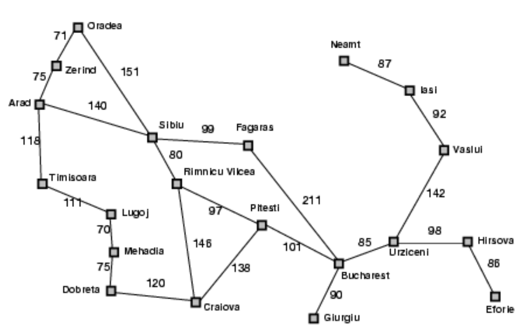




Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* search example





Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	10
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Properties of A*

- Complete?

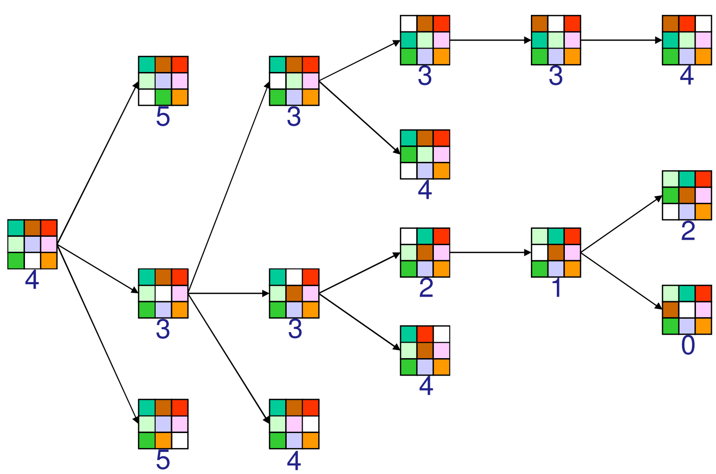
Yes (unless there are infinitely many)

- Time/Space?
- Exponential mostly
- Optimal?

Yes

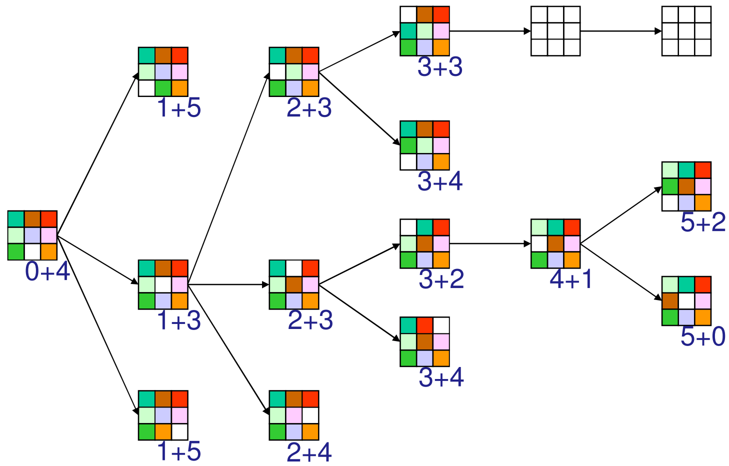
8-Puzzle

f(N) = h(N) = number of misplaced tiles



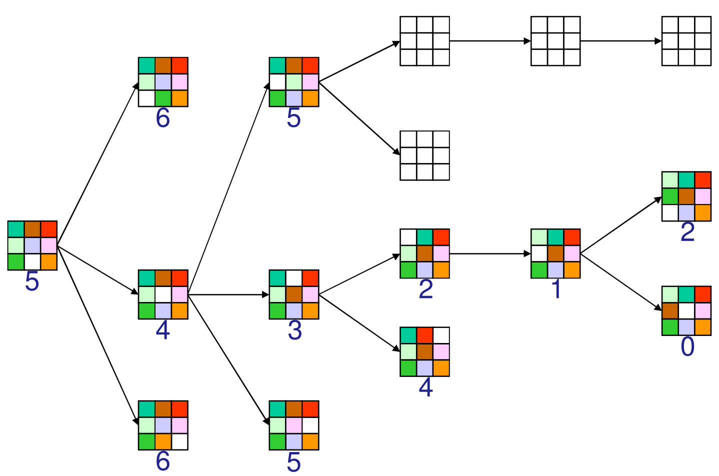
8-Puzzle

f(N) = g(N) + h(N)
with h(N) = number of misplaced tiles



8-Puzzle

f(N) = h(N) = Σ distances of tiles to goal



Relaxed problems

- A problem with fewer restrictions on the actions is called a relaxed problem
- If the rules of the 8-puzzle are relaxed so that a tile can move anywhere, then $h_1(n)$ gives the shortest solution
- If the rules are relaxed so that a tile can move to any adjacent square, then $h_2(n)$ gives the shortest solution