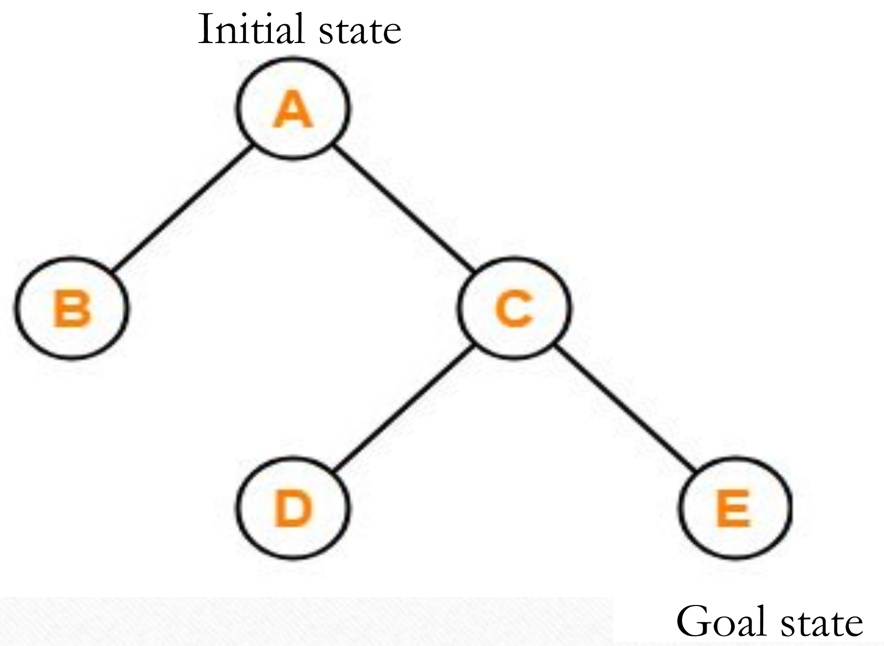
CSE 422
ARTIFICIAL INTELLIGENCE

HEURISTICS

Uninformed Search

Blind Search Strategy

- BFS, DFS, IDS etc.
- "*Blindly*" looks for the goal state
- No sense of how much closer or distant we are getting from the goal



The Heuristic Function

Heuristics

- **An estimate of the cheapest path cost from any node to a goal node.**
  - Represented by h(n)
- **Reduces time complexity**
  - By visiting only most promising nodes
  - Thus reducing the search space
- **To solve an NP problem in Polynomial time. For e.g.,**
  - 3^{22} possible states for an 8 puzzle problem
  - Search space and time increases exponentially
  - Can be completed in polynomial time using heuristics

The Heuristic Function

Heuristics

- **Greedy Best First Search,** **$f(n) = h(n)$**
- **A* Search,** **$f(n) = h(n) + g(n)$;**

Here **h(n)** is the estimated path cost from ***current node n*** to the ***goal node*** and **g(n)** is the actual path cost from ***initial state*** to ***current node n***

Some other examples may include :

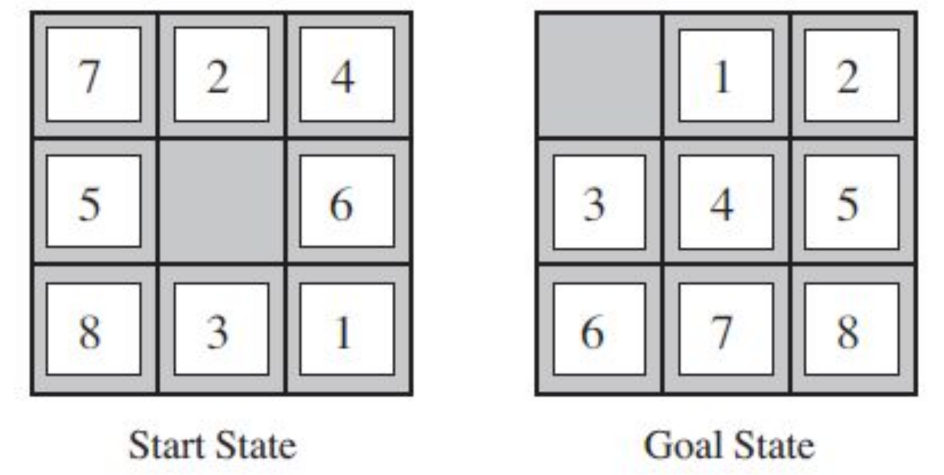
- **Euclidean Distance**
- **Manhattan Distance**
- **Misplaced Tiles**

Misplaced Tiles

8 puzzle problem

h(n) = number of tiles that are out of position in *state/node n*

- For the start state **h(n) = 8** as all tiles are out of position in comparison to the goal
- Following a best first search procedure, expanding the start state, we will get 4 newer states.
- Calculate value of h(n) for each state and expand the state with the least value of h(n) i.e. **the best state**
- Continue same process until we reach goal state



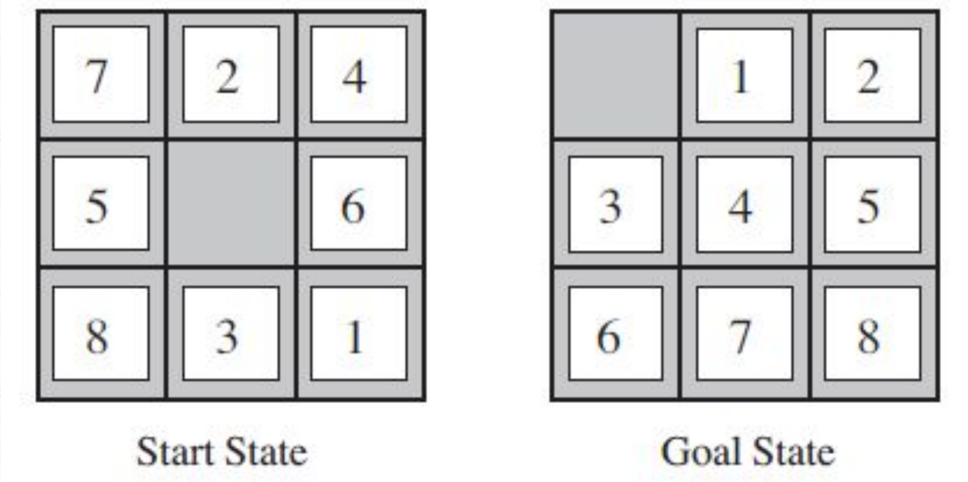
8 puzzle problem

Manhattan Distance

8 puzzle problem

h(n) = sum of the distances of the tiles from their current position to the goal positions for a *state n*

- For the start **state h(n) = 3 + 1 + 2 + 2+ 2 + 3 + 3 + 2 = 18** (distances for tiles 1 to 8).
- Remember that tiles can only move in right angles i.e. either vertically or horizontally
- Following a best first search procedure, expanding the start state, we will get 4 newer states.
- Calculate value of h(n) for each state and expand the state with the least value of h(n) i.e. **the best state**
- Continue same process until we reach goal state



8 puzzle problem

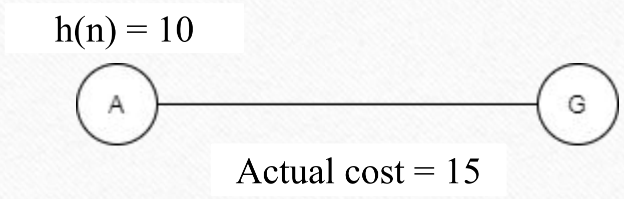
Choosing a Good Heuristic

Admissibility

- **If a heuristic, h(n) is admissible, then that means it will always underestimate the actual cost to the goal. An admissible heuristic will always give optimal results**

Lets say if,

- h(n) is the estimated cost from node n to the goal state
- h*(n) is the actual cost from node n to the goal state



Then h(n) will be admissible if h(n) ≤ h*(n) is true for all *states/nodes n*

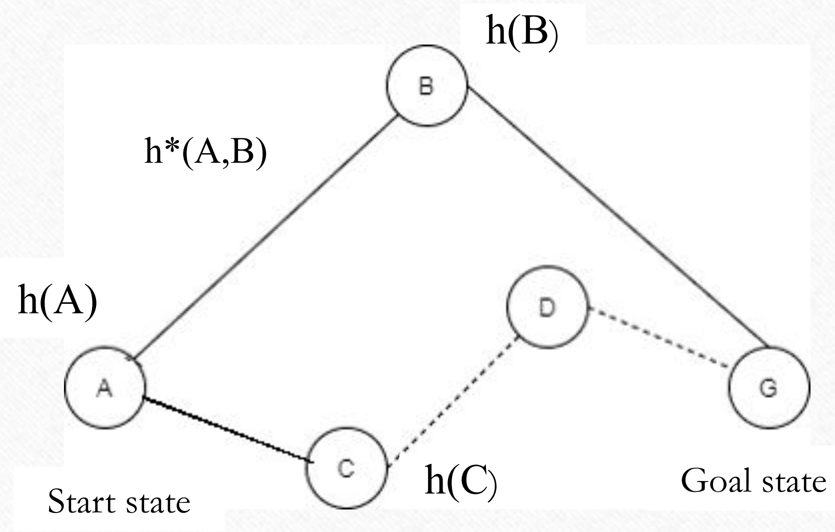
- **For e.g. Euclidean / Straight line distance is an admissible heuristic. Because the shortest path between any two points is the straight line distance between the points. Therefore it is the minimum possible distance between the two points and so it can never be greater than the actual distance.**

Choosing a Good Heuristic

Consistency

$$h(n) \leq h^*(n, n^*) + h(n^*)$$

- **h(n)** is the estimated heuristic from node n to the goal
- **n*** represents any neighbor of node n
- **h*(n, n*)** is the actual path cost from node n to any of its neighbors n*
- **h(n*)** is the estimated heuristic from node n* to the goal



And this should be true for all nodes n and for all of their neighbors n*

$$h(A) \leq h^*(A,B) + h(B)$$
$$h(A) \leq h^*(A,C) + h(C)$$

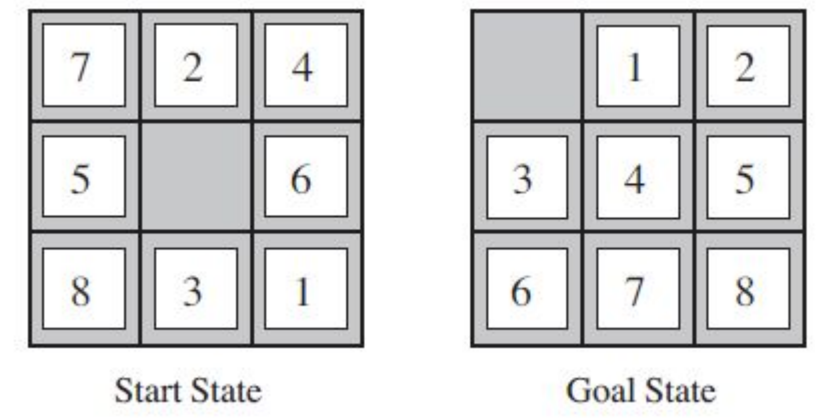
Choosing a Good Heuristic

Dominance

- **Total Misplaced tiles, $h_1 = 8$**
- **Total Manhattan distance, $h_2 = 18$**

Now if both $h_1(n)$ and $h_2(n)$ are admissible and $h_2(n) \geq h_1(n)$ for all nodes n, then we say **$h_2(n)$ *dominates* $h_1(n)$.**

- $h_2(n)$ will be better for search and will search less nodes than $h_1(n)$
- If there are several admissible heuristic, the one with highest value should be chosen
- The estimated cost, h (n) should be made as large as possible without exceeding the actual cost, h*(n)



8 puzzle problem

Choosing a Good Heuristic

Effective Branching Factor

$$N+1 = 1 + b^* + (b^*)^2 + (b^*)^3 + \ldots\ldots + (b^*)^d$$

- **N** is the total number of nodes generated by A* search
- **d** is the solution depth
- **b*** is the branching factor that a uniform tree of depth d would have to have in order to contain N + 1 nodes

For e.g. if we reach a goal state at depth 5 using 52 nodes after using A*, then the effective branching factor is 1.92

- The branching factor directly corresponds with the heuristic being used
- Choose the heuristic which results in **lesser branching factor**

Generate an Admissible Heuristic

Constraint Relaxation

- A **Relaxed problem** is created by removing preconditions or constraints from the original problem. This creates an approximation of the original problem by dropping the constraints. Newer edges are created in the relaxed problem making it simpler than the original problem. A solution to the relaxed problem will be a lower bound of the solution for the original problem

Therefore, an optimal solution to the relaxed problem will be an admissible heuristic for the original problem

Generate an Admissible Heuristic

Constraint Relaxation

- Identify the preconditions or constraints
- Create relaxed problems by dropping the preconditions or constraints
- Solve the relaxed problems without searching
- An optimal solution of a relaxed problem will be an admissible heuristic for the original problem.

Let us understand this with the help of the 8 puzzle problem

Generate an Admissible Heuristic

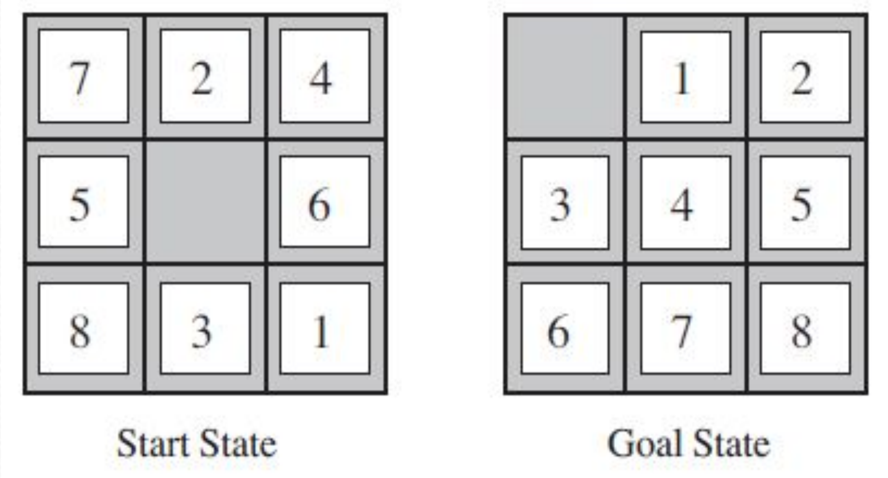
Constraint Relaxation

- **Preconditions/constraints:**

I. A tile will be on a specific cell

II. A tile can move to a new cell if it is horizontally or vertically adjacent to it

III. A tile can only move to a new cell if the new cell is clear/empty



8 puzzle problem

Generate an Admissible Heuristic

Constraint Relaxation

Keep precondition/constraint (I), (II) and drop (III) :

- A newer relaxed version of the problem is thus created
- A tile will be placed initially in a specific cell and any tile can move either vertically or horizontally without taking into account whether destination cell is clear/empty
- An optimum solution can be found by calculating the shortest path between the initial position and the goal positions for each tile
- This solution is exactly the same as Manhattan Distance heuristic.

Therefore, the Manhattan Distance can be a admissible heuristic for the original 8 puzzle problem

Generate an Admissible Heuristic

Constraint Relaxation

Drop precondition/constraint (II) and (III) :

- Again another newer relaxed version of the problem is thus created
- A tile will be placed initially in a specific cell and now can move into any other position in any way without worrying about adjacency or a clear/empty destination
- An optimum solution can be found by simply counting the number of tiles who are not in their goal positions
- Accordingly Misplaced Tiles can be used to calculated the length of the optimum solution here

Therefore, the solution using Misplaced Tiles will be a admissible heuristic for the original problem