Heuristic/Informed Searches

Heuristic Search

- This lecture introduces informed searches.
- The concept of Heuristics .
- The use of domain knowledge during search .
- Examples of informed searches.

Heuristic Search

- Heuristics help us to reduce the size of the search space.
- An evaluation function is applied to each goal to assess how promising it is in leading to the goal.
- Heuristic searches incorporate the use of domain-specific knowledge in the process of choosing which node to visit next in the search process.

Heuristic Search

- Search methods that include the use of domain knowledge in the form of heuristics are described as “weak” search methods.
- The knowledge used is “weak” in that it may help but does not always help to find a solution.
- Examples of heuristic searches : best first search, A* algorithm, hill-climbing.

Calculating Heuristics

- Heuristics are rules of thumb that may find a solution but are not guaranteed to.
- Heuristic functions have also been defined as evaluation functions that estimate the cost from a node to the goal node.
- The incorporation of domain knowledge into the search process by means of heuristics is meant to speed up the search process.
- Heuristic functions are not guaranteed to be completely accurate.

Calculating Heuristics

- Heuristic values are greater than or equal to zero for all nodes.
- Heuristic values are seen as an approximate cost of finding a solution.
- A heuristic value of zero indicates that the state is a goal state.
- A heuristic that never overestimates the cost to the goal is referred to as an **admissible** heuristic.
- Not all heuristics are necessarily admissible.

Calculating Heuristics

- A heuristic value of infinity indicates that the state is a “deadend” and is not going to lead anywhere.
- A good heuristic must not take long to compute.
- Heuristics are often defined on a simplified or relaxed version of the problem, e.g. the number of tiles that are out of place.

Calculating Heuristics

- A heuristic function h1 is better than some heuristic function h2 if fewer nodes are expanded during the search when h1 is used than when h2 is used.
- Experience has shown that it is difficult to devise heuristic functions.
- Furthermore, heuristics are fallible and are by no means perfect.

Example: 8-Puzzle Problem

Initial State

2	8	3
1	6	4
7		5

Goal State

1	2	3
8		4
7	6	5

Heuristics for the 8-Puzzle Problem

- Number of tiles out of place - count the number of tiles out of place in each state compared to the goal .
- Sum the distance that the tiles are out of place.
- Tile reversals - multiply the number of tile reversals by 2.

Examples 8-Puzzle Problem

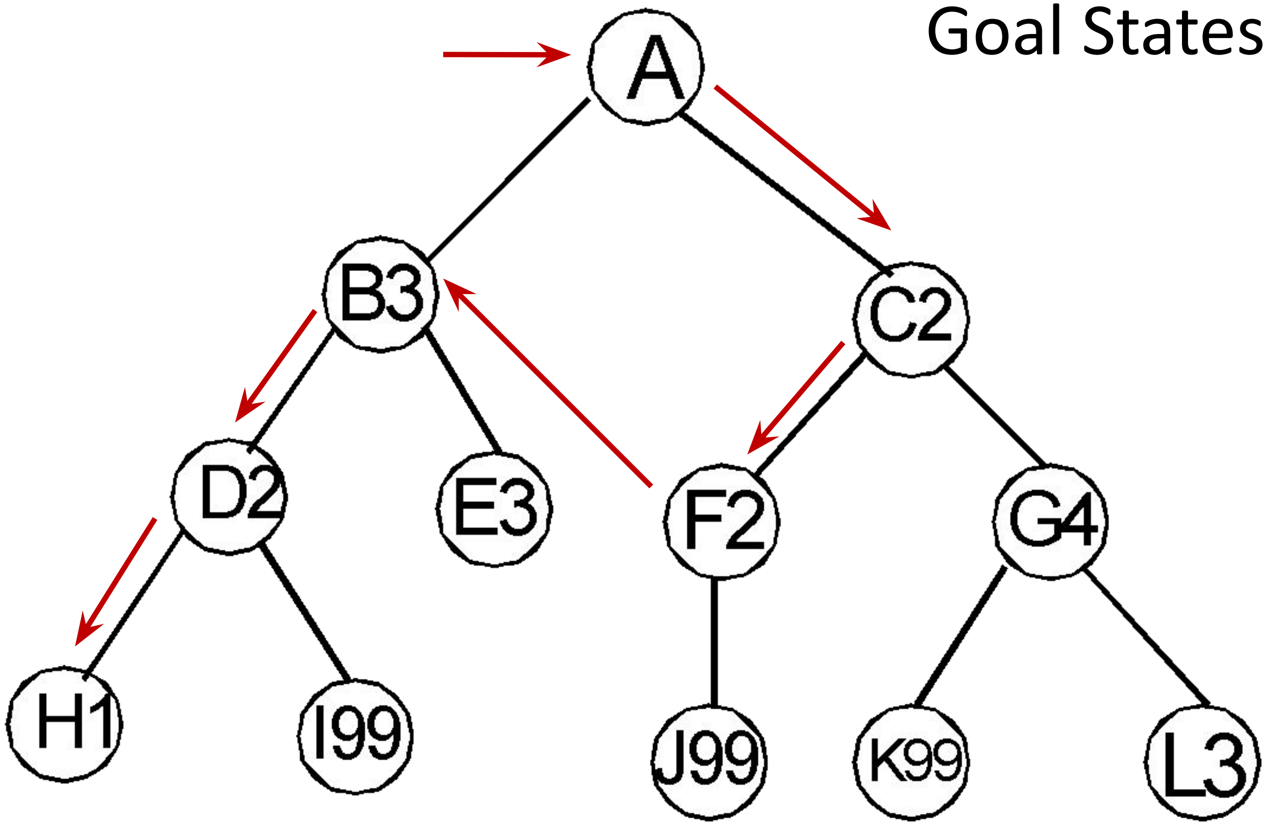
State	Tiles out of place	Sum of distances out of place	2 x the number of direct tile reversals
$\begin{array}{\|c\|c\|c\|}\hline 2 & 8 & 3 \\ \hline 1 & 6 & 4 \\ \hline & 7 & 5 \\ \hline \end{array}$	5	6	0
$\begin{array}{\|c\|c\|c\|}\hline 2 & 8 & 3 \\ \hline 1 & & 4 \\ \hline 7 & 6 & 5 \\ \hline \end{array}$	3	4	0
$\begin{array}{\|c\|c\|c\|}\hline 2 & 8 & 3 \\ \hline 1 & 6 & 4 \\ \hline 7 & 5 & \\ \hline \end{array}$	5	6	0

Best-First Search

- The best-first search is a general search where the minimum cost nodes are expanded first.
- The best- first search is not guaranteed to find the shortest solution path.
- The best-first search attempts to minimize the cost of finding a solution.
- Is a combination of the depth first-search and breadth-first search with heuristics.

Best-First Search

Goal States: H, L



Exercise

A to B4 and C4
B4 to D6 and E5
C4 to F4 and G5
D6 to I7 and J8
I7 to K7 and L8

Start state : A
Goal state : E

Hill-Climbing

- Hill-climbing is similar to the best first search.
- While the best first search considers states globally, hill-climbing considers only local states.
- The hill-climbing algorithm generates a partial tree/graph.

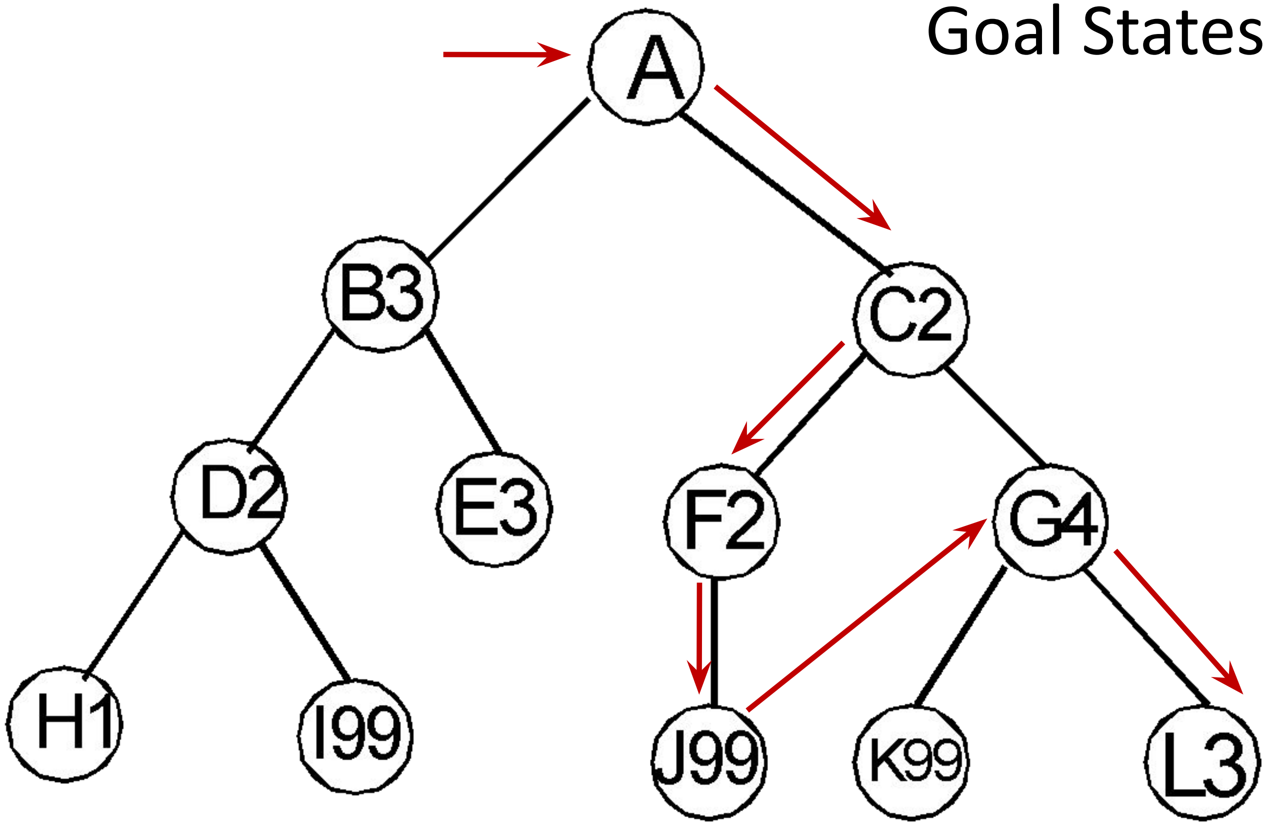
Example

A to B3 and C2
B3 to D2 and E3
C2 to F2 and G4
D2 to H1 and I99
F2 to J99
G4 to K99 and L3

Start state: A
Goal state: H, L

Hill-Climbing

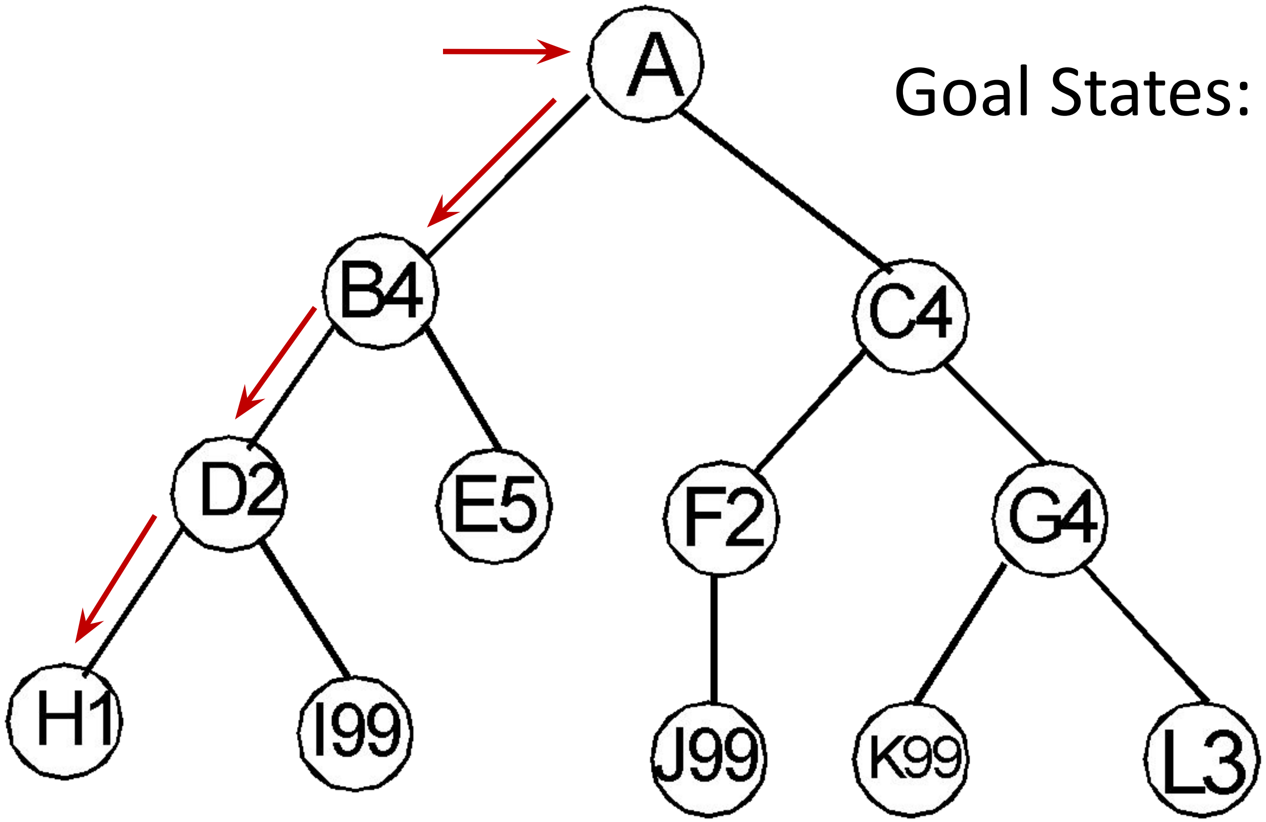
Goal States: H, L



Greedy Hill-Climbing

1. Evaluate the initial state.
2. Select a new operator.
3. Evaluate the new state
4. If it is closer to the goal state than the current state make it the current state.
5. If it is not better ignore
6. If the current state is the goal state or no new operators are available, quit. Otherwise repeat steps 2 to 4.

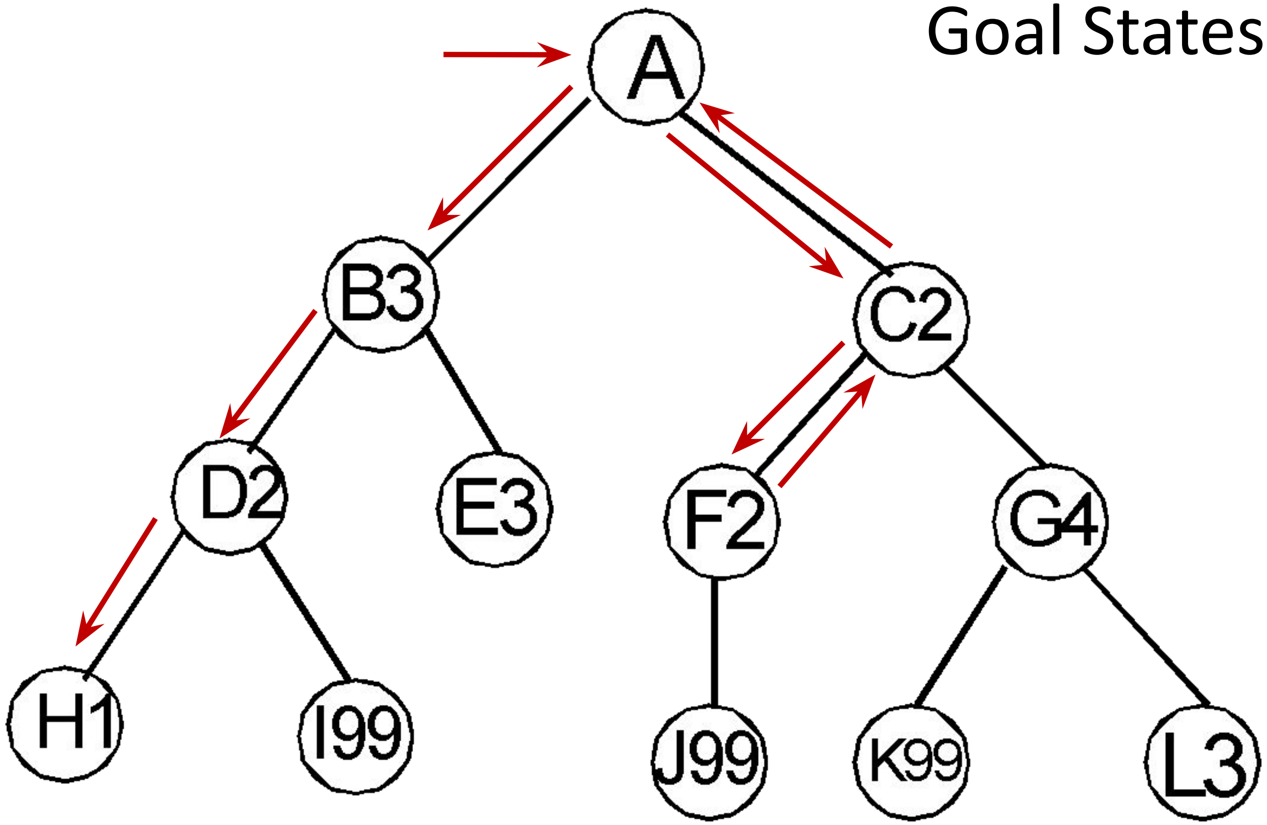
Example 1: Greedy Hill-Climbing without Backtracking



Goal States: H, L

Example 2: Greedy Hill-Climbing with Backtracking

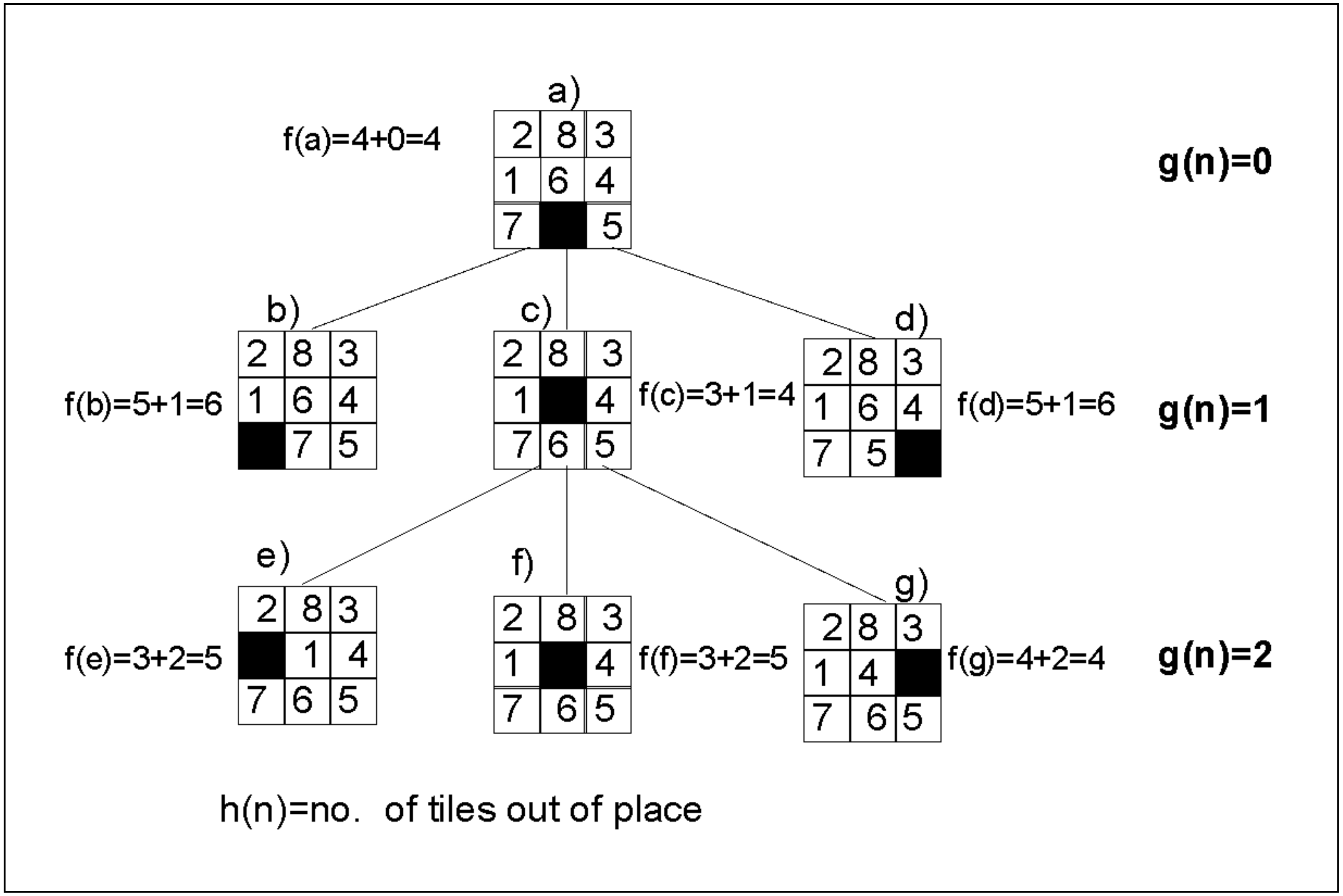
Goal States: H, L



A Algorithm

- The A algorithm is essentially the best first search implemented with the following function: $f(n) = g(n) + h(n)$
  - where $g(n)$ - measures the length of the path from any state n to the start state
  - $h(n)$ - is the heuristic measure from the state n to the goal state

8-Puzzle Example



Admissible Algorithms

- Search algorithms that are guaranteed to find the shortest path are called admissible algorithms.
- Between BFS and DFS which is admissible.
- The evaluation function we have considered with the best first algorithm is $f(n) = g(n) + h(n)$,
  - where $g(n)$ - is an estimate of the cost of the distance from the start node to some node n, e.g. the depth at which the state is found in the graph
  - $h(n)$ - is the heuristic estimate of the distance from n to a goal

Admissible Function

- An evaluation function used by admissible algorithms is:

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

  – where $g^*(n)$ – actual cost of the path from the start node to the node n.

  – $h^*(n)$ -actual cost from the node n to the goal node

Admissible Function

- f* does not exist in practice and we try to estimate f as closely as possible to f*.
- g(n) is a reasonable estimate of g*(n).
- Usually g(n) >= g*(n). It is not usually possible to compute h*(n).
- Instead we try to find a heuristic h(n) which is bounded from above by the actual cost of the shortest path to the goal, i.e. h(n) <= h*(n).
- The best first search used with f(n) is called the A algorithm.
- If h(n) <= h*(n) in the A algorithm then the A algorithm is called the A* algorithm.

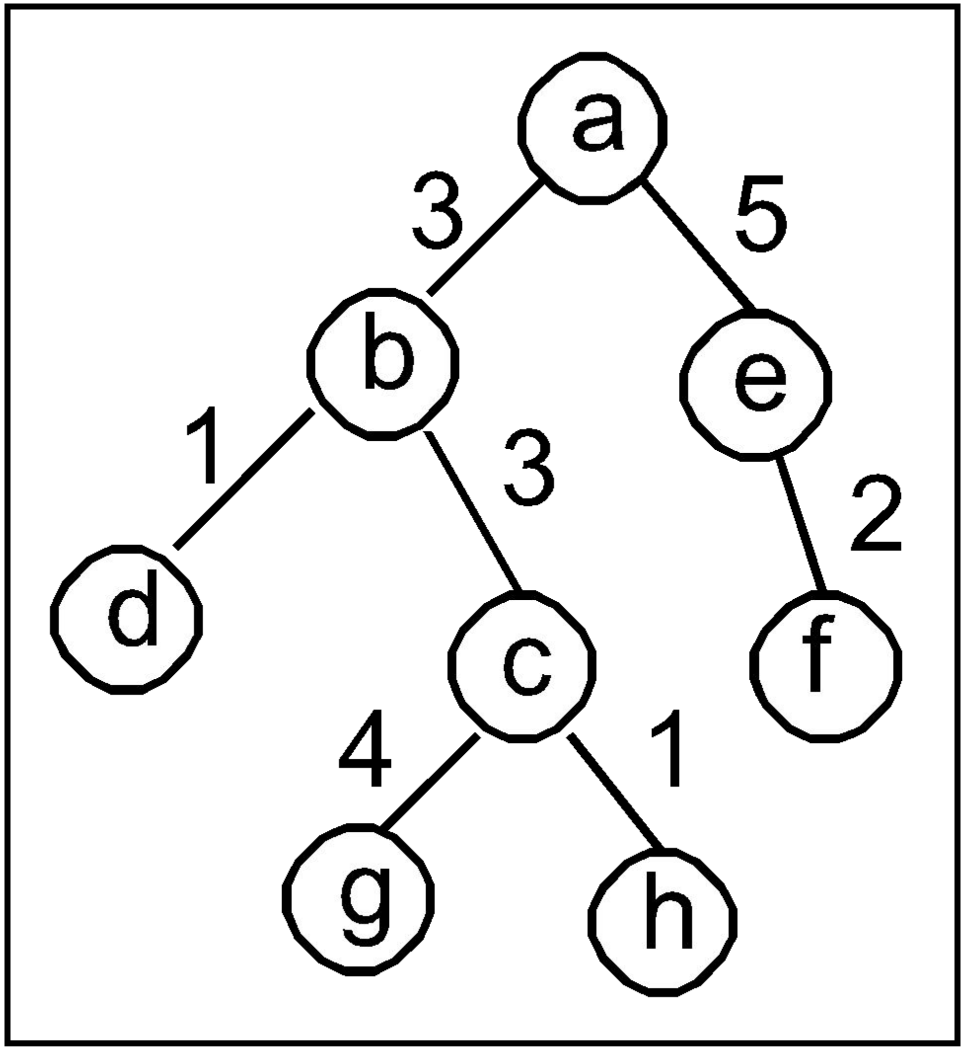
Admissible Heuristic and the 8-Puzzle Problem

- The heuristics that we have developed for the 8- puzzle problem are bounded above by the number of moves required to move to the goal position.
- The number of tiles out of place and the sum of the distance from each correct tile position is less than the number of required moves to move to the goal state.
- Thus, the best first search applied to the 8-puzzle using these heuristics is in fact an A* algorithm.

Branch and Bound Techniques

- Branch and bound techniques are used to find the most optimal solution.
- Each solution path has a cost associated with it.
- Branch and bound techniques keep track of the solution and its cost and continue searching for a better solution further on.
- The entire search space is usually searched.
- Each path and hence each arc (rather than the node) has a cost associated with it.
- The assumption made is that the cost increases with path length.
- Paths that are estimated to have a higher cost than paths already found are not pursued.
- These techniques are used in conjunction with depth first and breadth first searches as well as iterative deepening.

Example



QUESTIONS