



# Informed (Heuristic) Search

---

- Concept
- Informed search methods



# Objectives

---

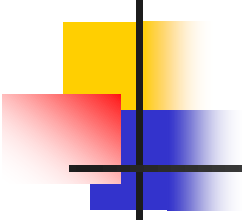
- At the end of the subtopic the learner should be able to:
  - Explain the term informed search and Heuristic Function
  - State the different type of informed search algorithms.
  - Explain how the following algorithms searches for a solution in state space
    - Best First Search
    - Hill climbing
    - Greedy search
    - A\* search algorithm



# Search with Domain Knowledge added

---

- Uninformed (blind) searches are normally very inefficient
- Adding domain knowledge can improve the search process.



# Concept of informed (heuristic) search

---

- **Heuristic (informed) search** -> explore the node that is most “likely” to be the nearest to a goal state.
- There is no guarantee that the heuristic provided most “likely” node will get you closer to a goal state than any other.

# Knowledge/info



- Example:

- City streets

- You are at GPO and would like to the Globe Roundabout to get a matatu - What path will you take? Is it the shortest?
      - Parallel streets on all sides

# Knowledge/info



- Example:
  - An office in a building
    - Find me in Office door number 31 (ART Complex )
      - Wing, floor, left/right

# Knowledge/info



- Example:

- Visit the doctor

- Symptoms: fever, nausea, headache, ...
      - Leading questions: how long?, traveled?, ... (Malaria, typhoid, meningitis, flu,..)
      - x Blood test, y test,...

# Knowledge/info



- Example:

- Climbing a hill in thick Fog

- Heuristic function: check the change in altitude in 4 directions; the strongest increase is the direction in which to move next.





# Heuristic Searches - Characteristics

---

- Has some domain knowledge
- Usually more efficient than blind searches
- Also called informed search
- Heuristic searches work by deciding which is the next best node to expand (there is no guarantee that it is the best node)



# Heuristic Searches - Why Use?

---

- It may be too resource intensive (both time and space) to use a blind search
- Even if a blind search will work we may want a more efficient search method



# Heuristic Search Methods

---

- Methods that use a heuristic function to provide specific knowledge about the problem:
  - Best First Search
  - Hill climbing
  - Greedy search
  - A\* search algorithm



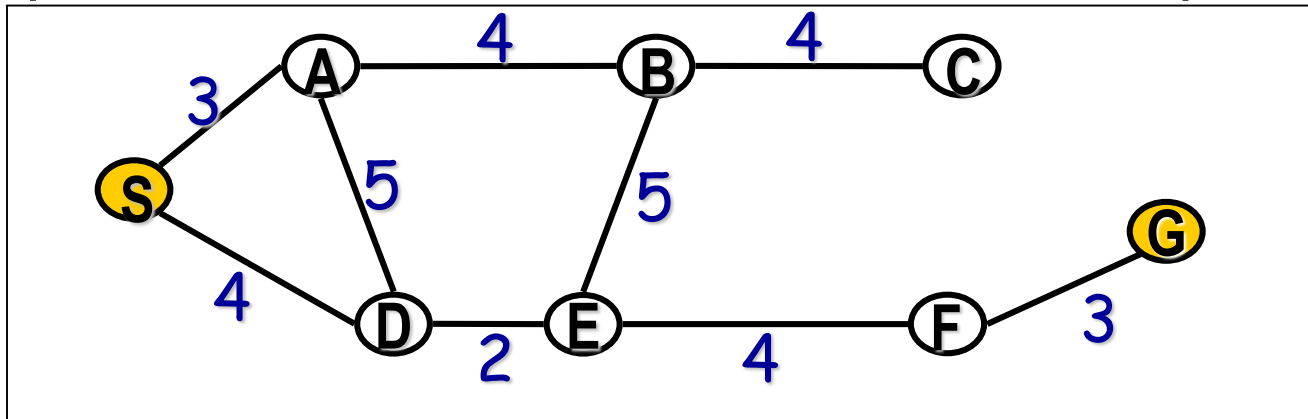
# Heuristic Functions

- To further improve the quality of the previous methods, we need to include problem-specific knowledge on the problem.
  - How can this be done in such a way that the algorithms remain generally applicable ???

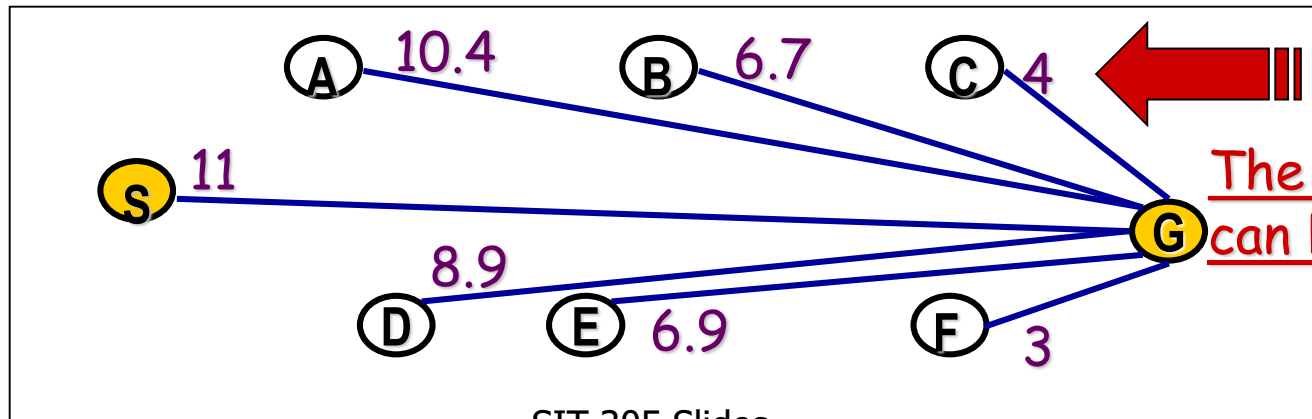
- **HEURISTIC FUNCTIONS:**
  - $f$ : States  $-->$  Numbers
  - $f(T)$  : expresses the quality of the state  $T$ 
    - allow to express problem-specific knowledge in the search method algorithm

# Example 1: road map

- Imagine the problem of finding a route on a road map and that the NET below is the road map:



- Define  $f(T)$  = the straight-line distance from T to G



## Example 2: 8-puzzle

- $f_1(T)$  = the number correctly placed tiles on the board:

$$f_1 \left( \begin{array}{|c|c|c|} \hline 1 & 3 & 2 \\ \hline 8 & & 4 \\ \hline 5 & 6 & 7 \\ \hline \end{array} \right) = 4$$

- $f_2(T)$  = number of incorrectly placed tiles on board:
  - gives (rough!) estimate of how far we are from goal

$$f_2 \left( \begin{array}{|c|c|c|} \hline 1 & 3 & 2 \\ \hline 8 & & 4 \\ \hline 5 & 6 & 7 \\ \hline \end{array} \right) = 4$$

Most often, 'distance to goal' heuristics are more useful !

# Example 2: 8-puzzle

## Manhattan distance

- $f_3(T)$  = the sum of ( the horizontal + vertical distance that each tile is away from its final destination):
  - gives a better estimate of distance from the goal node

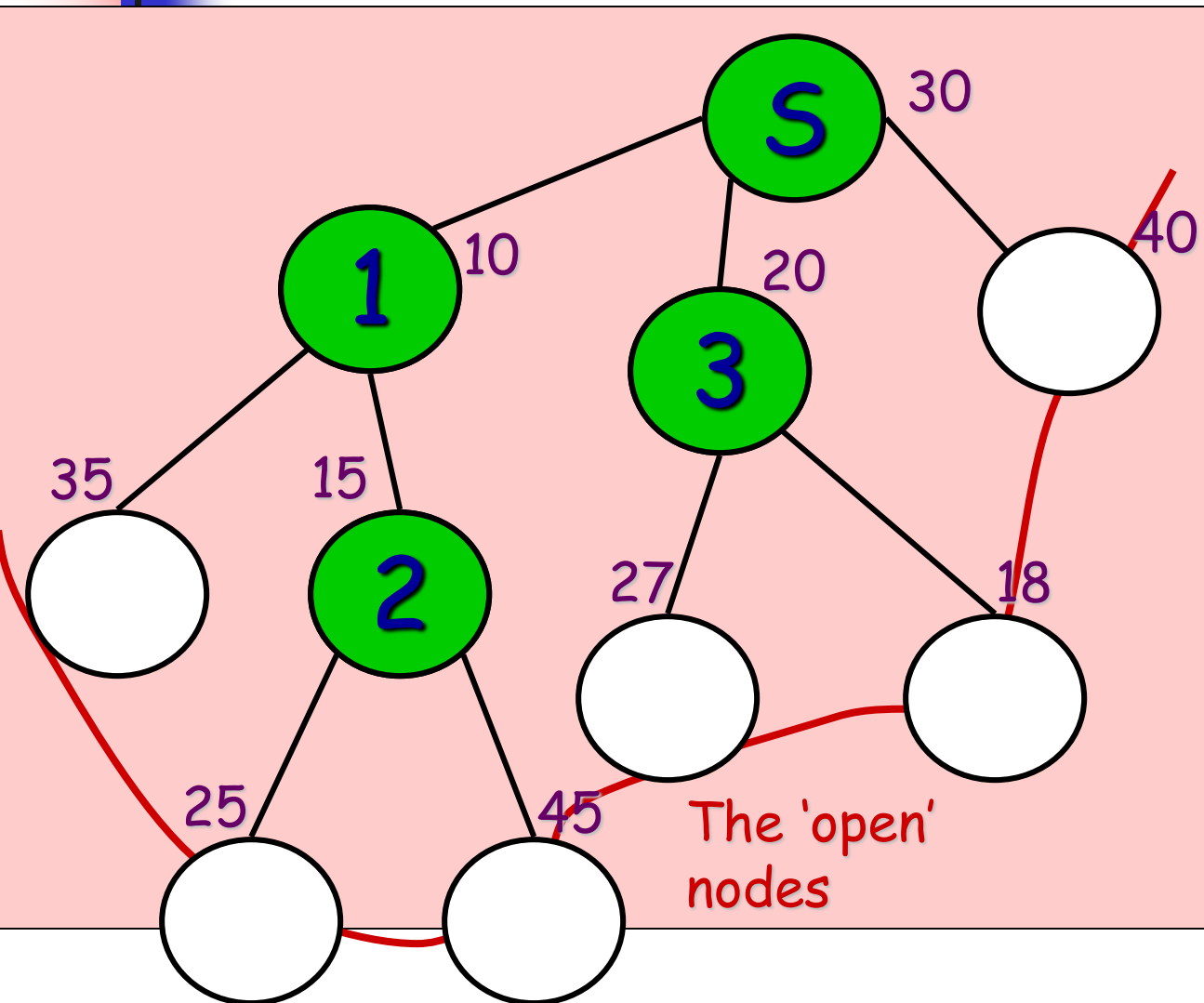
$$f_2 \left( \begin{array}{|c|c|c|} \hline 1 & 3 & 2 \\ \hline 8 & & 4 \\ \hline 5 & 6 & 7 \\ \hline \end{array} \right) = 1 + 1 + 2 + 2 = 6$$

# Heuristic Searches - Greedy search

- Tries to expand the node that is closest to the goal, on the grounds that this is likely to lead to a solution quickly.
  - Thus,  $f(n)=h(n)$ , where
    - $H(n)$ =estimated cost of the cheapest path from node  $n$  to a goal node
- in other words,
- Always expand the heuristically best nodes first.



# Greedy search, or Heuristic best-first search:



- At each step, select the node with the best (in this case: lowest) heuristic value.

- paths with loops;  
paths and sort the entire Q



# Heuristic Searches - Greedy Search

---

- It is only concerned with short term aims
- It is possible to get stuck in an infinite loop, unless you check for repeated states
- It is not optimal
- It is not complete

**Time and space complexity is  $O(B^m)$ ; where  $m$  is the depth of the search tree**

# Heuristic Searches - A\*

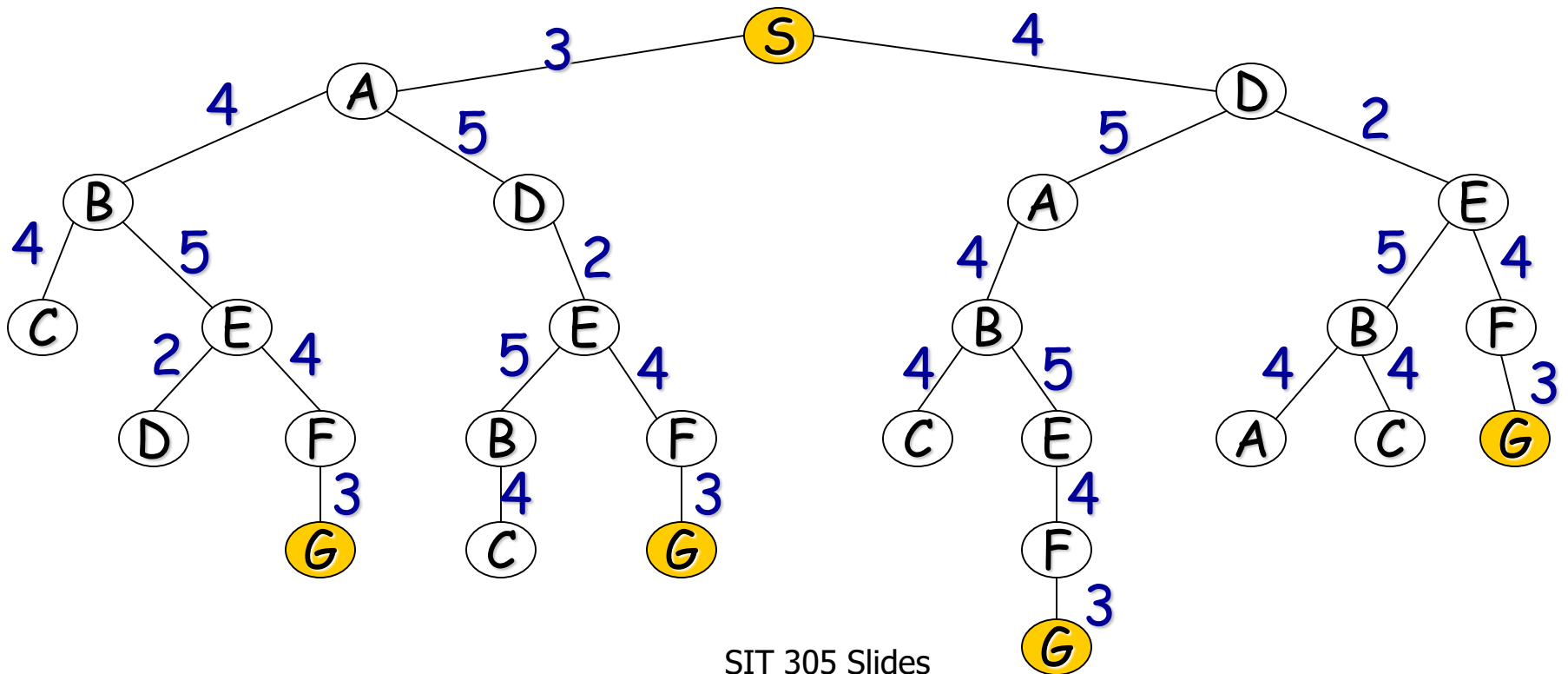
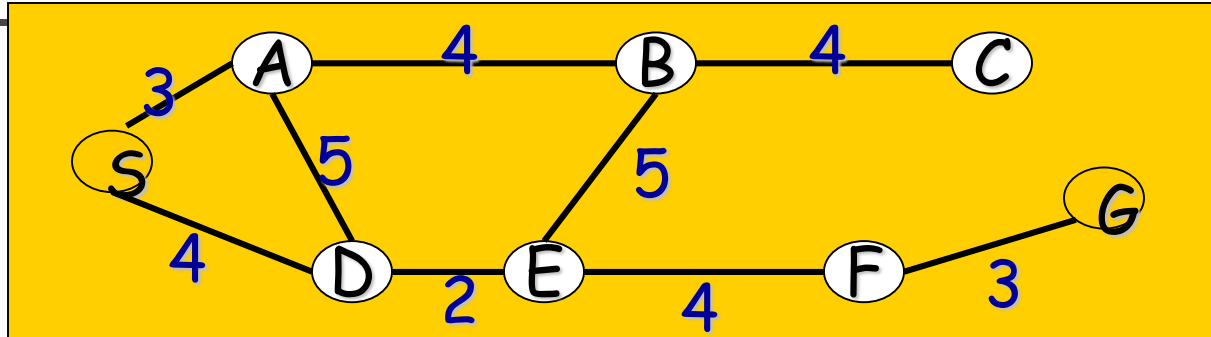
## Algorithm

---

- A combination of Greedy search and Uniform cost search. - to compliment one another
- This search method minimises the cost to the goal using an heuristic function,  $h(n)$ . Greedy search can considerably cut the search time but it is neither optimal nor complete. By comparison uniform cost search minimises the cost of the path so far,  $g(n)$ . Uniform cost search is both optimal and complete but can be very inefficient.

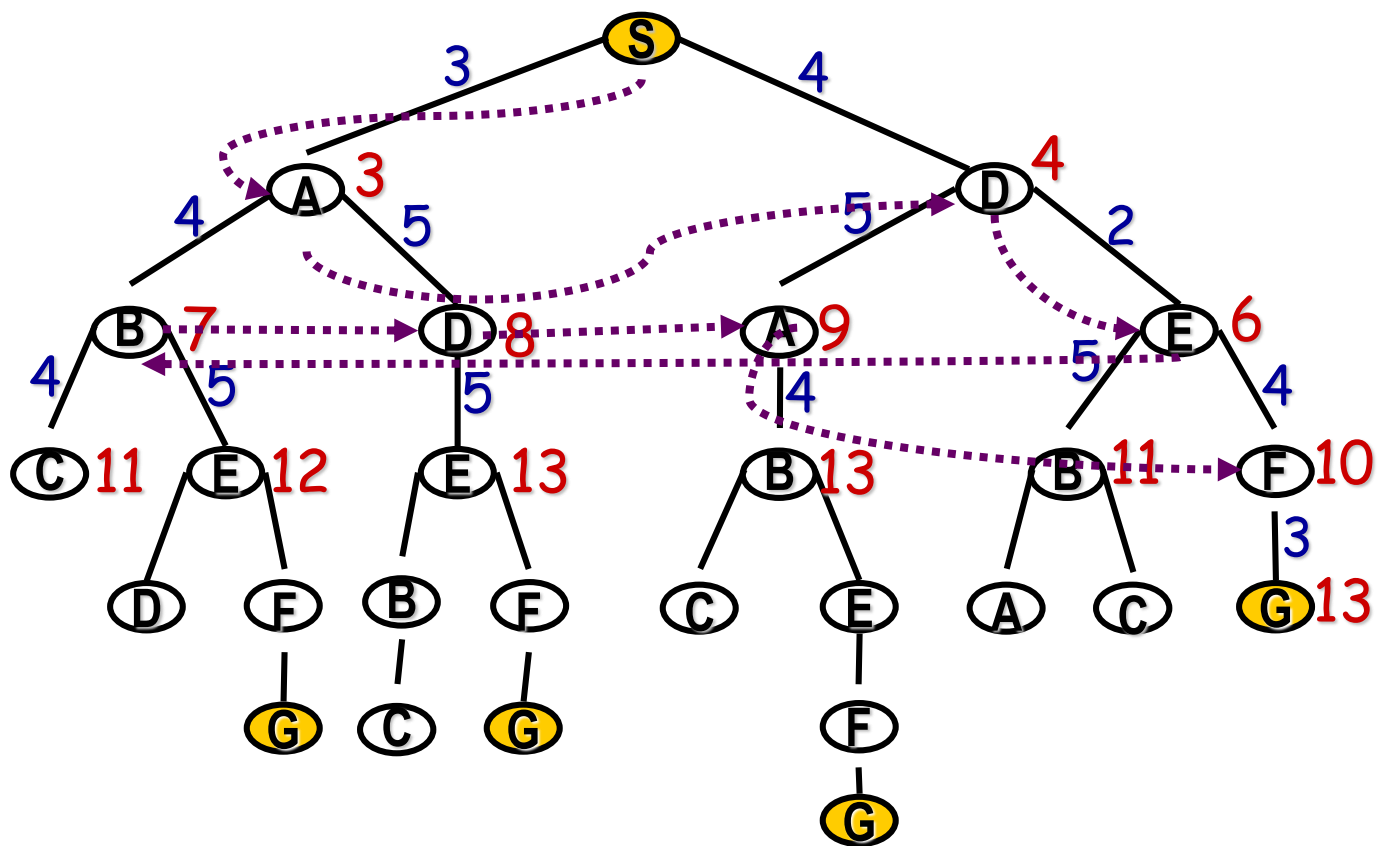
# Road map example:

Re-introduce the costs of paths in the NET



# A look at Uniform cost search

## = uniformed best-first



- At each step, select the node with the lowest accumulated cost.

# Now incorporate heuristic estimates

- Replace the 'accumulated cost' in the 'Uniform cost search' by a function:

$$f(\text{path}) = \text{cost}(\text{path}) + h(\text{endpoint\_path})$$

fn

g(n)

h(n)

- where:

$\text{cost}(\text{path})$  = the accumulated cost of the partial path

$h(T)$  = a heuristic estimate of the cost remaining from  $T$  to a goal node

$f(\text{path})$  = an estimate of the cost of a path extending the current path to reach a goal.

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



# Heuristic Searches - A\*

## Algorithm

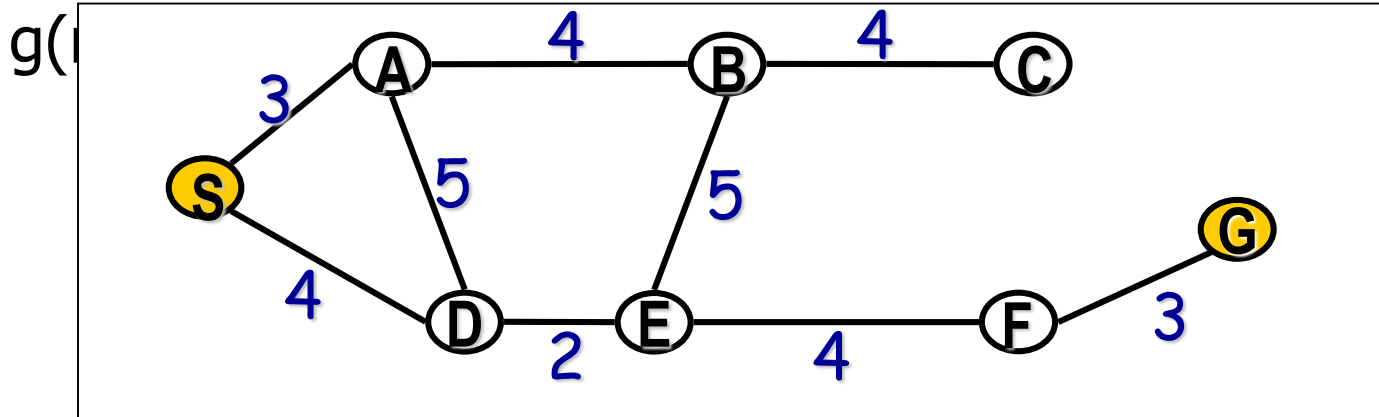
---

- Combines the cost so far and the heuristic estimated cost to the goal. That is  $fn = g(n) + h(n)$   
This gives us estimated cost of the cheapest solution through path  $n$
- It can be proved to be optimal and complete providing that the heuristic is admissible.  
That is the heuristic must never over estimate the cost to reach the goal.
- But, the number of nodes that have to be searched still grows exponentially

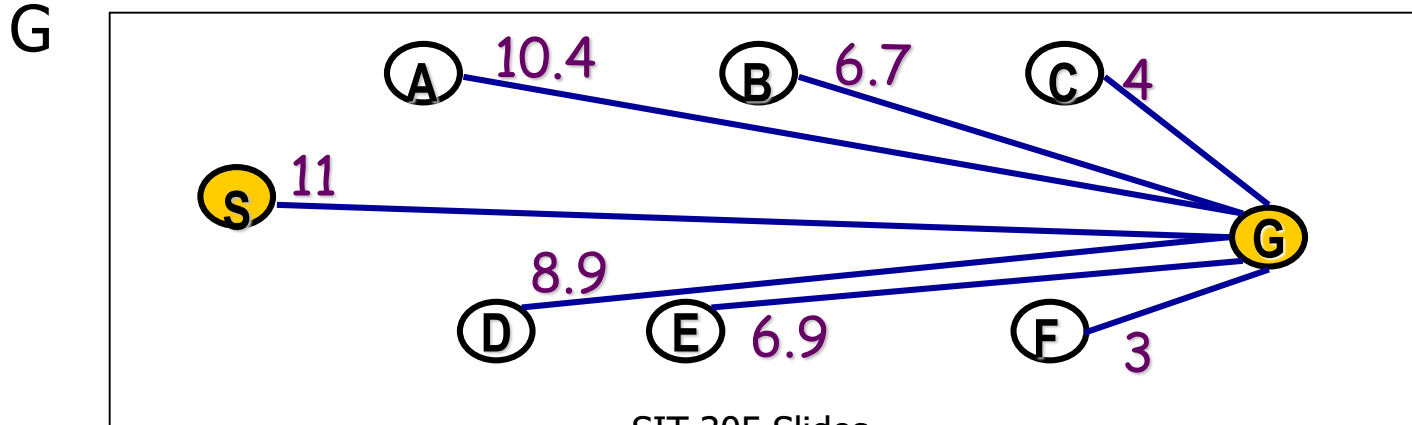


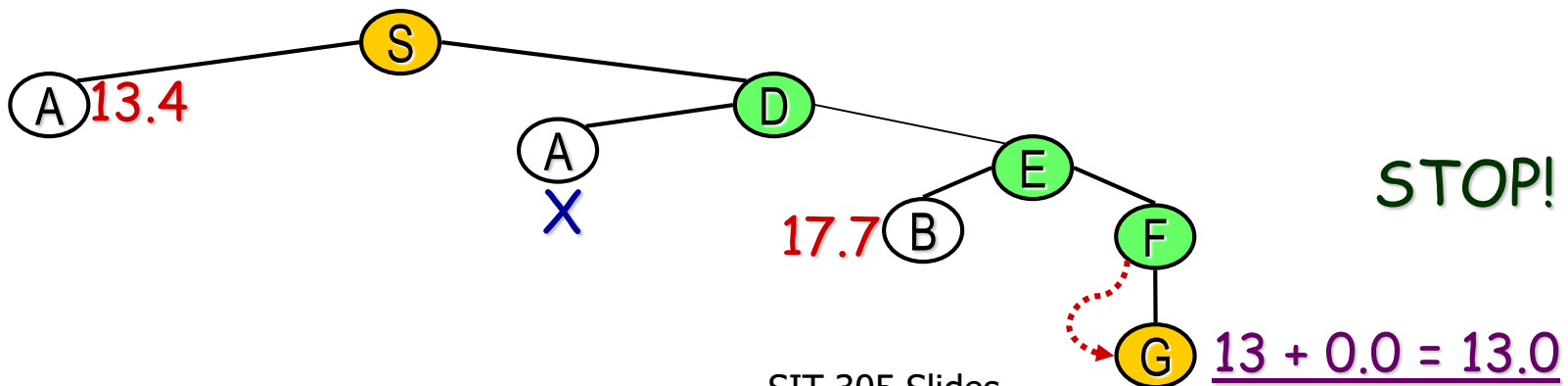
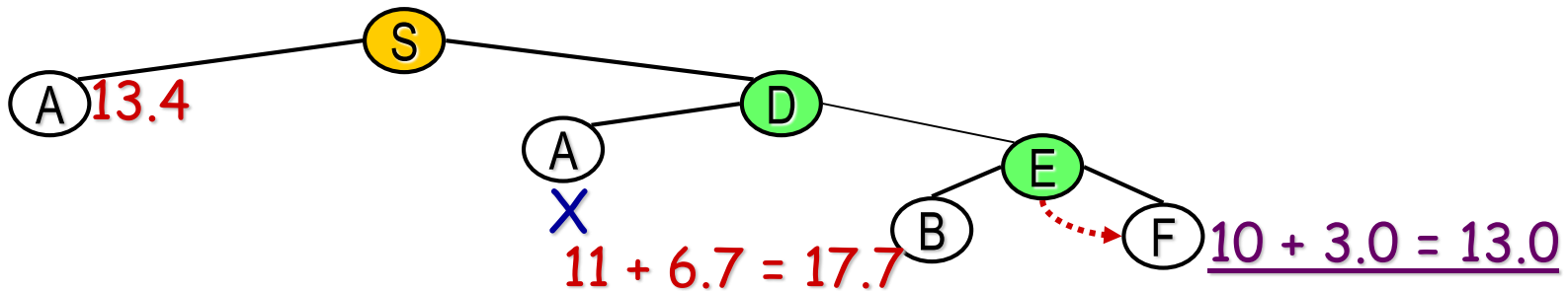
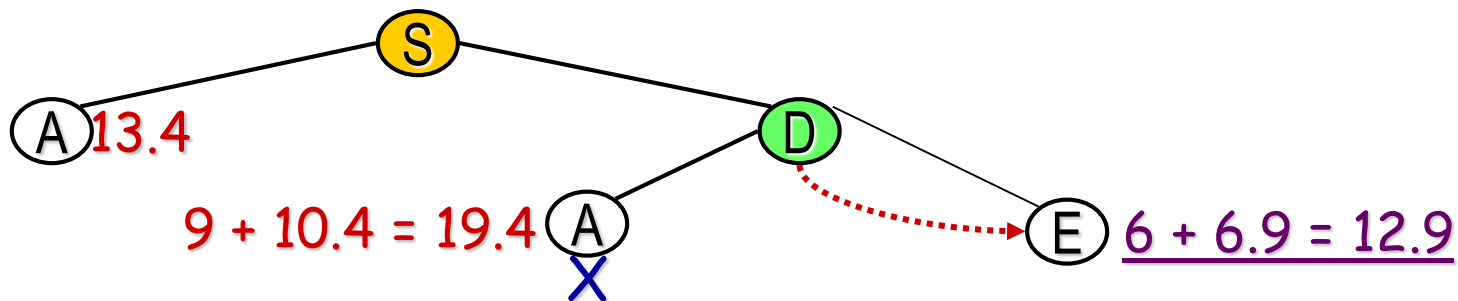
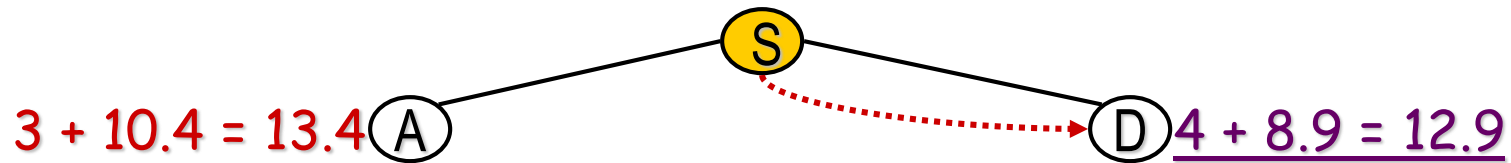
# Example: road map

- Imagine the problem of finding a route on a road map. The paths distances between nodes define



- Define  $h(n)$  = the straight-line distance from node to





# Heuristic Searches - A\*

## Algorithm for 8-Puzzle Example

5	4	
6	1	8
7	3	2

Initial State

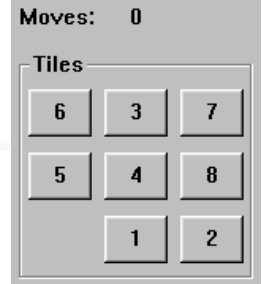
1	2	3
8		4
7	6	5

Goal State



- Typical solution is about twenty steps
- Branching factor is approximately three. Therefore a complete search would need to search  $3^{20}$  states. But by keeping track of repeated states we would only need to search  $9!$  (362,880) states
- But even this is a lot (imagine having all these in memory)
- Our aim is to develop a heuristic that does not over estimate (it is admissible) so that we can use A\* to find the optimal solution

# Heuristic Searches - Possible Heuristics



- $H_1$  = the number of tiles that are in the wrong position (=7)
- $H_2$  = the sum of the distances of the tiles from their goal positions using the Manhattan Distance (=18)

*Both are admissible but which one is best?*

# Test from 100 runs with varying solution depths

Search Cost			
Depth	IDS	A*(h <sub>1</sub> )	A*(h <sub>2</sub> )
2	10	6	6
4	112	13	12
6	680	20	18
8	6384	39	25
10	47127	93	39
12	364404	227	73
14	3473941	539	113
16		1301	211
18		3056	363
20		7276	676
22		18094	1219
24		39135	1641

H<sub>2</sub> looks better as fewer nodes are expanded. But why?



# Effective Branching Factor

Search Cost				EBF		
Depth	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23

- $H_2$  has a lower branching factor and so fewer nodes are expanded
- Therefore, one way to measure the quality of a heuristic is to find its average branching factor
- $H_2$  has a lower EBF and is therefore the better heuristic



# Local search

---

- The path to the goal does not matter
- Does not depend on path: path not retained
- Operate using a single current state and generally move only to neighbors of that state
- Adv:
  - use of little memory and
  - can often find reasonable soln. in large or infinite state spaces.
  - Optimizations problems- find the best sol. According to an objective function



# Local search

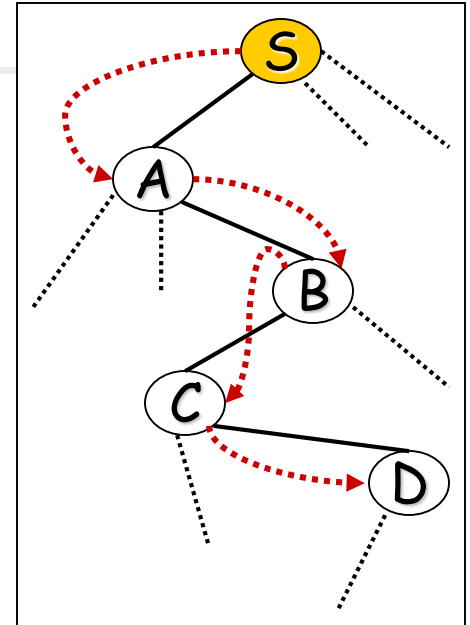
---

- To understand a Local Search Consider a state space landscape
- Diagram
  - Global maximum/minimum
  - Local maximum/minimum
- A landscape has both “location” (defined by the state ) and “elevation” (defined by the value of the heuristic cost function or objective function). If elevation corresponds to cost, then the aim is to find the lowest valley-Global minimum; If elevation corresponds to an objective function, then the aim is to find the highest peak Global maximum;

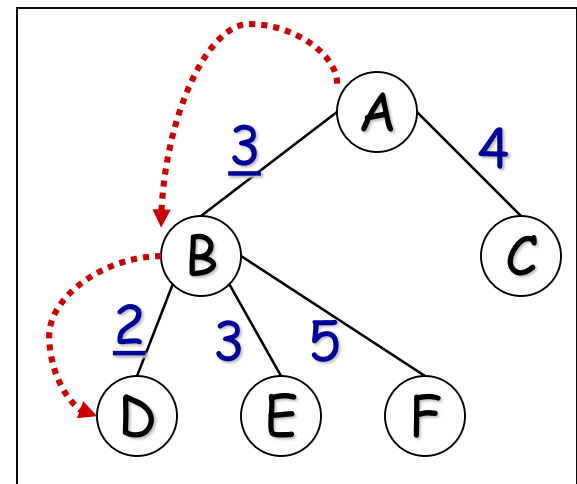


# Local search

- Hill climbing\_2 is an example of local search.
- In local search, we only keep track of 1 path and use it to compute a new path in the next step.
  - QUEUE is always of the form:
    - (  $p$  )



- Another example:
  - **MINIMAL COST** search:
- If  $p$  is the current path:
  - the next path extends  $p$  by adding the node with the smallest cost from the endpoint of  $p$

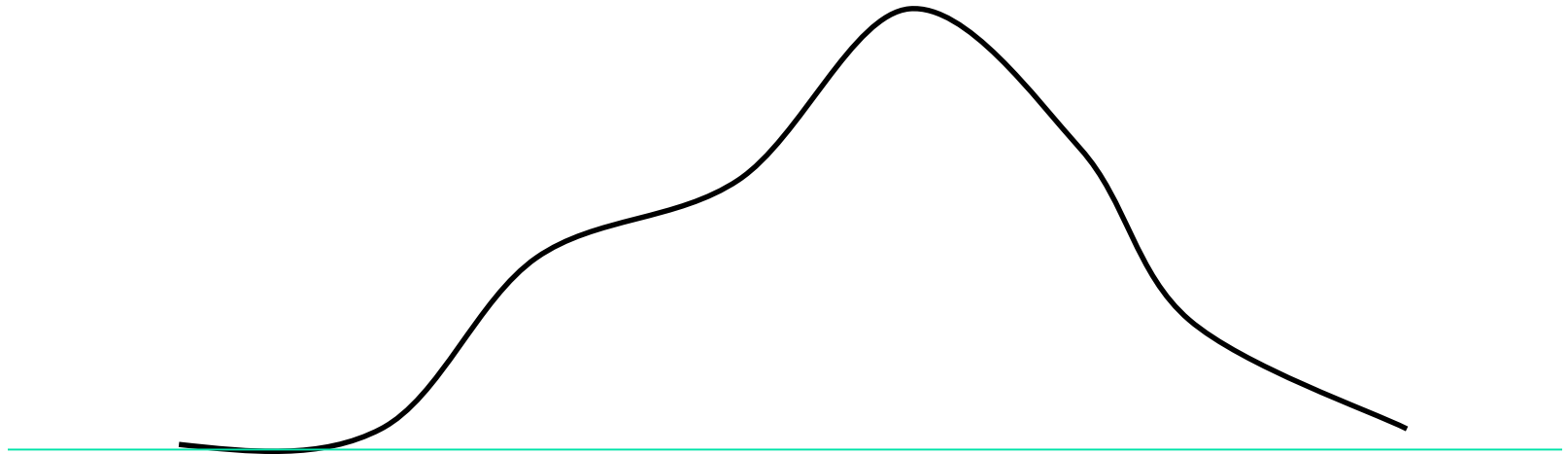




# Hill climbing

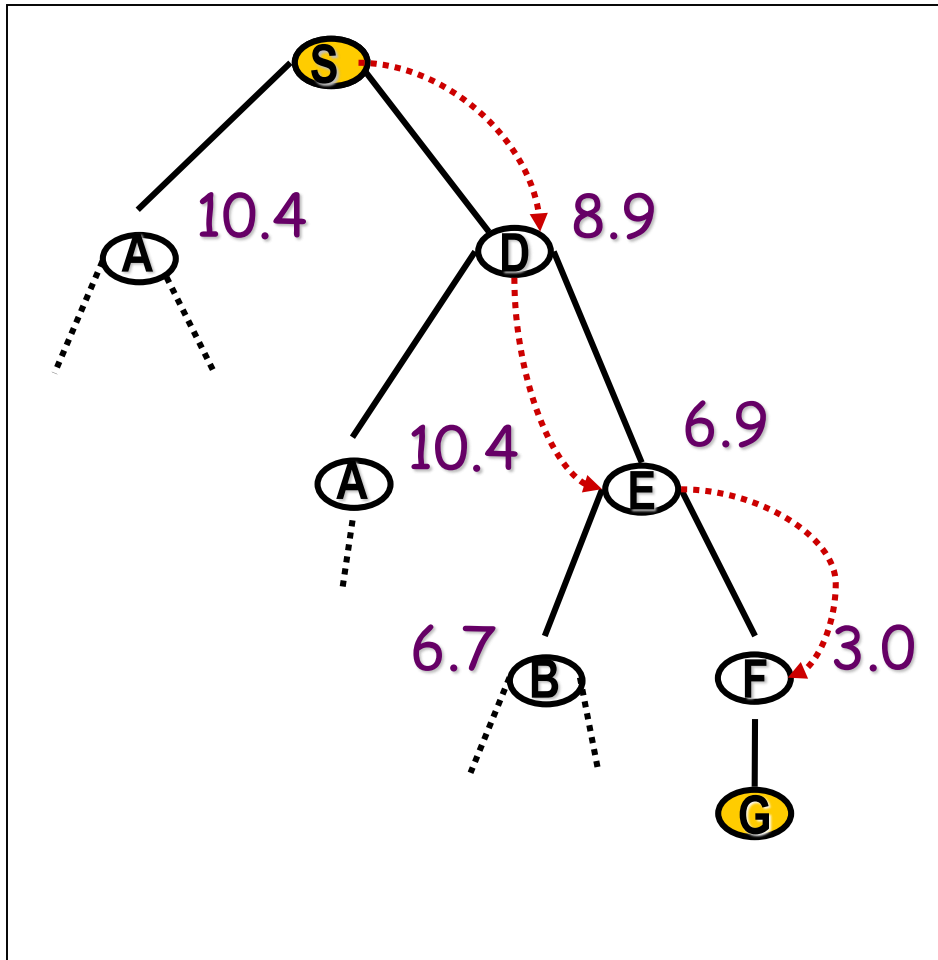
---

- A basic heuristic search method:
  - depth-first + heuristic



# Hill climbing\_1

- Example: using the straight-line distance:



- Perform depth-first, BUT:
- instead of left-to-right selection,
- FIRST select the child with the best heuristic value



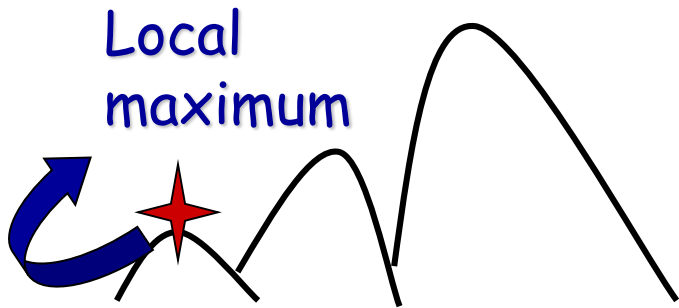
# Hill climbing\_2

---

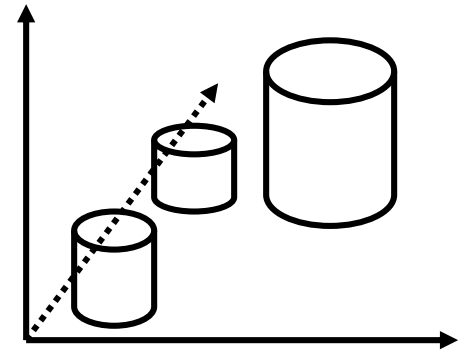
- Inspiring Example: climbing a hill in the fog.
  - Heuristic function: check the change in altitude in 4 directions: the strongest increase is the direction in which to move next.
- Is identical to Hill climbing\_1, except for dropping the backtracking.
- Produces a number of classical problems:

# Problems with Hill climbing\_2:

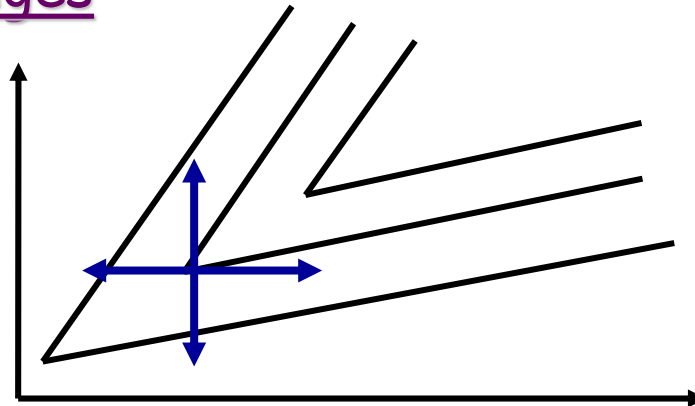
## Foothills:



## Plateaus



## Ridges





# Comments:

---

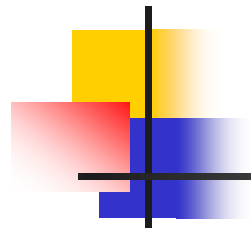
- Foothills are local minima: hill climbing\_2 can't detect the difference.
- Plateaus don't allow you to progress in any direction.
  - Foothills and plateaus require random jumps to be combined with the hill climbing algorithm.
- Ridges neither: the directions you have fixed in advance all move downwards for this surface.
  - Ridges require new rules, more directly targeted to the goal, to be introduced (new directions to move) .



# Summary

---

- Blind Search is very expensive
- A Search with info (heuristics) is far better
- Info can be difficult to incorporate
- The more info, the better

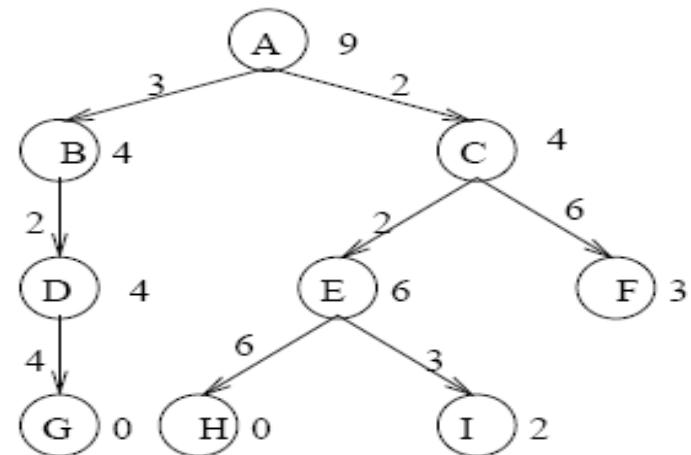




# Discussion Question

Consider the following state space in which the states are shown as nodes labeled A through I. A is the initial state, and G and H are the goal states. The numbers alongside the edges represent the costs of moving between the states. To the right of every state is the estimated cost of the path from the state to the nearest goal. Show how each of the following search strategies finds a solution in this state space by writing down, in order, the names of the nodes expanded. Assume the search halts when a goal state is found. In some cases, multiple answers are possible. You need give only one such answer in each case.

- Greedy search
- A\* search algorithm



# More Resources for self-learning

- A\* Search Algorithm:

<https://www.youtube.com/watch?v=PzEWHH2v3TE> or

<https://www.youtube.com/watch?v=wJ9RyRBkgPo>

- Hill climbing Algorithm:

[https://www.youtube.com/watch?v=\\_ThdIOA9Lbk](https://www.youtube.com/watch?v=_ThdIOA9Lbk)

- Best First Search algorithm:

[https://www.youtube.com/watch?v=i4MA\\_hFkKDg](https://www.youtube.com/watch?v=i4MA_hFkKDg)