INTRODUCTION

Intelligence

Intelligence is the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines.

Types of Intelligence

i) General intelligence: - Abilities that allow us to be flexible and adaptive thinkers, not necessarily tied to acquired knowledge
ii) Linguistic-verbal intelligence: - Use words and language in various forms / Ability to manipulate language to express oneself poetically. Linguistic intelligence is evident in poets, novelists, journalists, and effective public speakers.
iii) Logical-Mathematical intelligence: - Ability to detect patterns / Approach problems logically / Reason deductively. Logical intelligence is usually well developed in mathematicians, scientists, and detectives.
iv) Musical intelligence: - Recognize nonverbal sounds: pitch, rhythm, and tonal patterns. This intelligence enables us to recognize, create, reproduce, and reflect on music, as demonstrated by composers, conductors, musicians, vocalist, and sensitive listeners.
v) Spatial intelligence: - Typically thinks in images and pictures / Used in both arts and sciences. Core capacities include mental imagery, spatial reasoning, image manipulation, graphic and artistic skills, and an active imagination. Sailors, pilots, sculptors, painters, and architects all exhibit spatial intelligence.
vi) Intrapersonal intelligence: - Ability to understand oneself, including feelings and motivations / Can discipline themselves to accomplish a wide variety of tasks. Intra-personal intelligence involves not only an appreciation of the self, but also of the human condition. It is evident in psychologist, spiritual leaders, and philosophers.
vii) Interpersonal intelligence: - Ability to "read people"—discriminate among other individuals especially their moods, intentions, motivations. Teachers, social workers, actors, and politicians all exhibit interpersonal intelligence.
viii) Naturalist intelligence: - Ability to recognize and classify living things like plants, animals. Generally, naturalist intelligence designates the human ability to discriminate among living things (plants, animals) as well as sensitivity to other features of the natural world (clouds, rock configurations).
ix) Bodily-Kinesthetic intelligence: - Use one's mental abilities to coordinate one's own bodily movements. This intelligence also involves a sense of timing and the perfection of skills through mind–body union. Athletes, dancers, surgeons, and crafts people exhibit well-developed bodily kinesthetic intelligence

Artificial Intelligence (A I)

Artificial Intelligence is the development of systems that exhibit the characteristics we associate with intelligence in human behavior: perception, natural language processing, reasoning, planning and problem solving, learning and adaptation, etc.

Characteristics of AI Programs

a) Symbolic Processing

Symbolic processing is reasoning about objects represented by symbols, and their properties and relationships, rather than numerical calculations. Generally A.I. programs primarily use *symbolic* representations: collections of symbols that represent:

- Objects.
- Properties of objects.
- Relationships among objects.
- Rules about classes of objects.

b) Knowledge Representation

Knowledge is the general principles that are stored in the program and used for reasoning about novel situations. This knowledge is represented by some kind of data structures in the machine's memory that reflect

the complexity of the real world. Several kinds of knowledge need to be represented such as factual Data, general principles, hypothetical data etc.

c) **Search**

This is a method for finding a solution to a problem when no direct method exists. Search programs find a solution for a problem by trying different *sequences* of *actions* (*operators*) until a solution is found.
Search fits in nearly every area of A.I. in the following ways:

- **Natural Language:** A parser searches for the best ways of assigning structure and meaning to sentences that are ambiguous.
- **Planning:** A planner searches for a sequence of actions that will accomplish a goal.
- **Perception:** The raw input is often ambiguous; the perception program searches for a consistent set of interpretations of parts of the input.
- **Learning:** A learning program searches for a compact description of a set of training instances.
- **Expert Systems:** Search finds rules applicable to the current problem.

d) Heuristics

Are similar to rules of thumb where you need not rethink completely what to do every time a similar problem is encountered.

e) Inferencing

This is a form of reasoning with facts and rules using heuristics or some search strategies.

PROBLEM SOLVING TECHNIQUES IN ARTIFICIAL INTELLIGENCE

Problems are tackled in AI using two main broad approaches namely:

i). Symbolic AI

Problem solving by *searching*

- Ability to represent *knowledge*
- Ability to *reason* etc
- AI programming languages

ii). Sub Symbolic AI

- Artificial Neural Networks
- Connectionism

Searching technique

Searching is the process of looking for the solution of a problem through a set of possibilities (state space or search space). The *solution* is a path from the current state to the goal state.

Examples of Search Problems

- Route finding
  - Search routes for one that reaches destination
- Theorem proving
  - Search chains of reasoning for proof
- Machine learning
  - Search through concepts for one which achieves target categorisation

Search Terminology

- **States**: "places" the search *can* visit
- **Search space**: the set of possible states
- **Search path:** Sequence of states the agent *actually* visits
- **Solution:**
  - A state which solves the given problem, either known or has a checkable property
  - May be more than one solution
- **Strategy**: How to choose the next state in the path at any given state

Process of Searching
Searching proceeds as follows:

1. Check the current state;
2. Execute allowable actions to move to the next state;
3. Check if the new state is the solution state; if it is not then the new state becomes the current state and the process is repeated until a solution is found or the state space is exhausted.

Search problem
The search problem consists of finding a solution plan, which is a path from the current state to the goal state.
Specifying a Search Problem

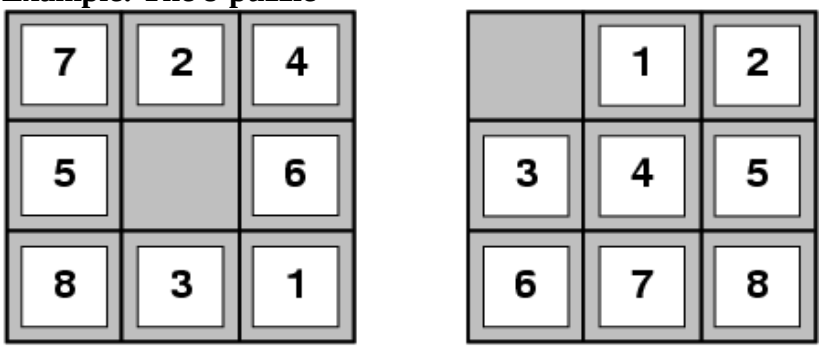
1. Initial state
   - Where the search starts
2. Operators
   - Function taking one state to another state
   - How the agent moves around search space
3. Goal test
   - How the agent knows if solution state found

Representing search problems
A search problem is represented using a directed graph. The states are represented as nodes while the allowed steps or actions are represented as arcs.

Example: Search Problem
On holiday in Kenya; currently in Kisumu.

- Formulate goal:
  - be in Nairobi
- Formulate problem:
  - states: various cities
  - actions: drive between cities
- Find solution:
  - sequence of cities, e.g., Kisumu, Kericho, Kakamega, Nakuru, Narok

Example: The 8-puzzle



Start State Goal State

- states? locations of tiles
- actions? move blank left, right, up, down
- goal test? = goal state (given)
- path cost? 1 per move

Example of a search Problem
Three blocks A, B, C on a table are considered. A block can be grasped when there is no other block on top of it. Only one block can be moved at a time.

Possible moves

- ✓ Put a block on table;
- ✓ Put a block on top of another block;
- ✓ Remove a block from the top of another and place on top of another block.

Search Problem

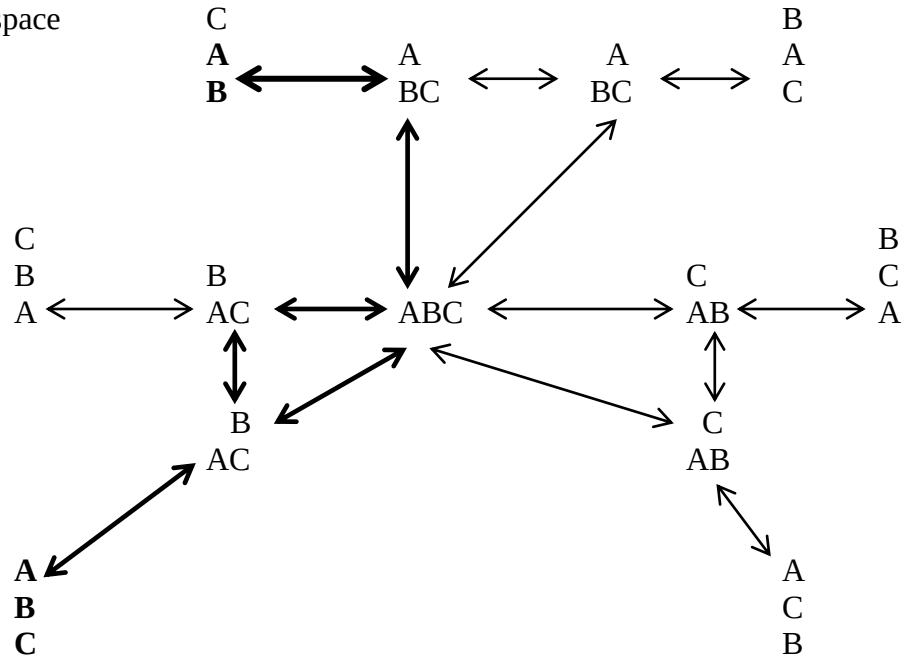
Initial state (current state)

C
A
B

Goal state (final state)

A
B
C

State space



The state space has 13 elements or nodes.

The solution to our problem is any member of the set of all paths from original to goal state such as the path indicated in bold.

Search Strategies

A search strategy is defined by picking the order of node expansion. Different search strategies are evaluated in terms of four criteria:

1. **Completeness**: is the strategy guaranteed to find a solution when there is one?
2. **Time complexity**: how long does it take to find a solution?
3. **Space complexity**: how much memory does it need to perform the search?
4. **Optimality:** does the strategy find the highest quality solution when there are several different solutions?

Time and space complexity are measured in terms of

- *b:* maximum branching factor of the search tree
- *d:* depth of the least-cost solution
- *m*: *maximum* depth of the state space (may be ∞)

Example Search Problem

- A genetics professor
  - Wants to name her new baby boy
  - Using only the letters D,N & A
- Search through possible strings (states)
  - D,DN,DNNA,NA,AND,DNAN, etc.
  - 3 operators: add D, N or A onto end of string
  - Initial state is an empty string
- Goal test
  - Look up state in a book of boys' names, e.g. DAN

Types of Searches

There are two broad classes of searches:

i) Uninformed search (blind / Exhaustive search)
ii) Informed search (Heuristic/ Guided search)

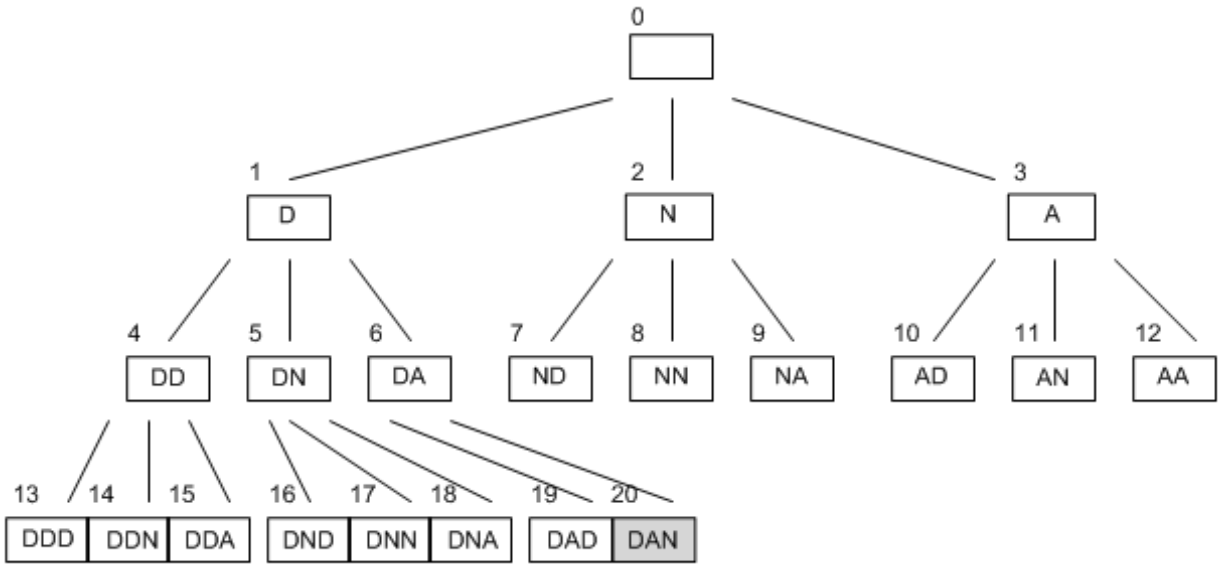
1) **Uninformed search:**

This is a search that has no information about its domain. The only thing that a blind search can do is distinguish a non-goal state from a goal state. The following are the various uniform search algorithms:

i). Breadth-first search
ii). Depth-first search
iii). Iterative deepening search
iv). Bidirectional search

i) Breadth-first search:- Expand shallowest unexpanded node

Implementation: It is a FIFO queue, i.e., new successors go at end

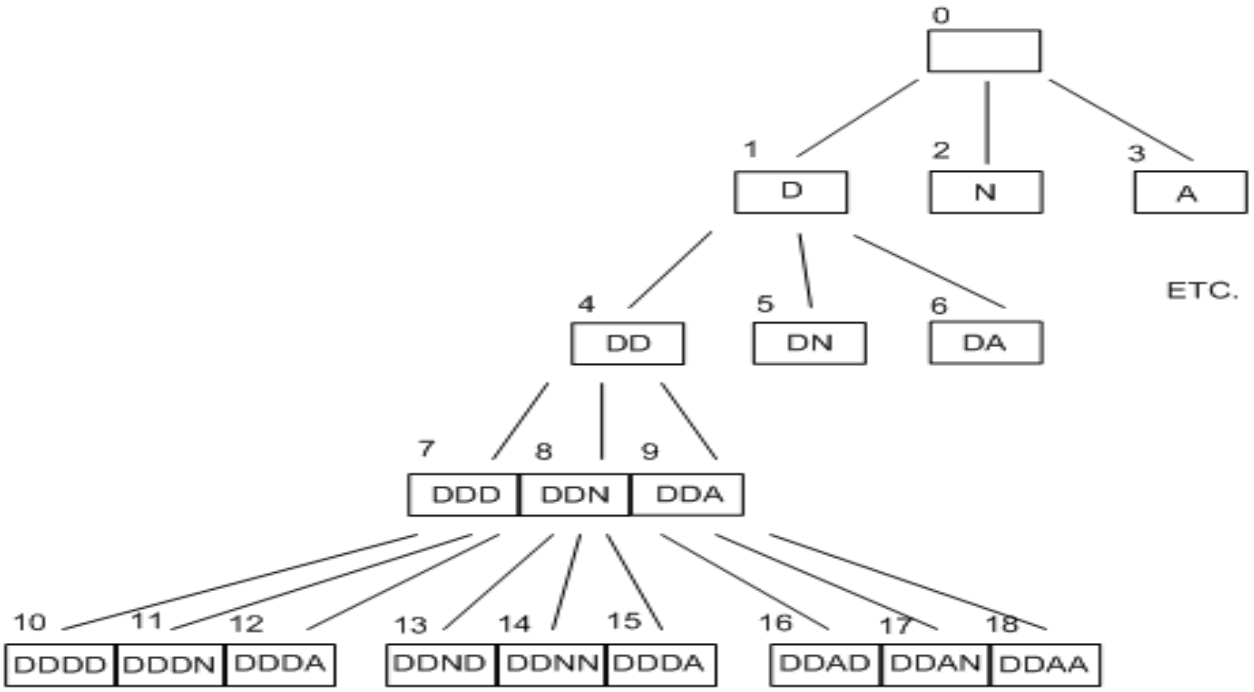


The following are characteristics of breadth-first searches:

- Very systematic and all paths of length l will be considered before paths of length l + 1.
- If there is a solution, breadth-first is guaranteed to find it.
- If there are several solutions, breadth-first will always find the shallowest goal state first.
- Not preferred in terms of time and space complexities. I.e. for a problem has a path length of d, the maximum number of nodes expanded is $1 + b + b^2 + b^3 + \ldots + b^d$, Where b is the branching factor.

Note: Breadth first search can only be used for small problems. If we have larger problems to solve then there are better blind search strategies that can be used (depth first search).

ii) **Depth-first search:-** Expand deepest unexpanded node
Implementation:
It is LIFO queue, i.e., put successors at front



The following are characteristics of breadth-first searches:

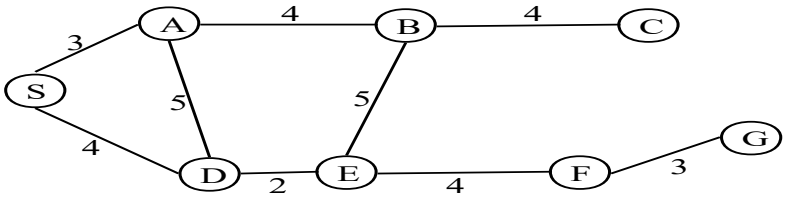
- It has modest memory requirements. For a state space with a branching factor of b and a maximum depth of m, depth first search requires storage of b^m nodes.
- Depth first search is neither complete nor optimal. i.e. If depth first search goes down an infinite branch it will not terminate if it does not find a goal state. If it does find a solution there may be a better solution at a higher level in the tree.

2) **Informed search**: -
Is a search that uses domain-dependent (heuristic) information in order to search the space more efficiently. The idea behind a ***heuristic*** search is to explore the node that is most *likely* to be nearest to a goal state. To do this, a heuristic function which tells us how close we are to a goal state is used. Heuristic is used as an estimate, based on domain-specific information that is computable from the current state description, of how close we are to a goal
The following are the various heuristic search algorithms:
i). Uniform-cost search
ii). Hill climbing
iii). Greedy search
iv). A* search
v). IDA* search

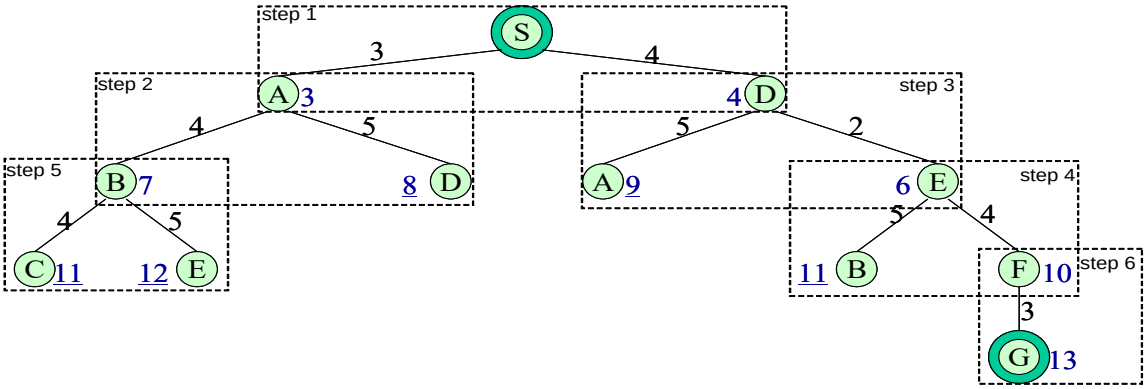
i). Uniform-cost search:-The search looks for the cheapest path from the start node to a goal node. i.e. this method always extends the cheapest path found so far. In this way it is guaranteed to find the cheapest path, if one exists.

Example
The figure below contains a representation of a map. The nodes represent cities, and the links represent direct road connections between cities. The number associated to a link represents the length of the corresponding road.

The search problem is to find a path from a city S to a city G

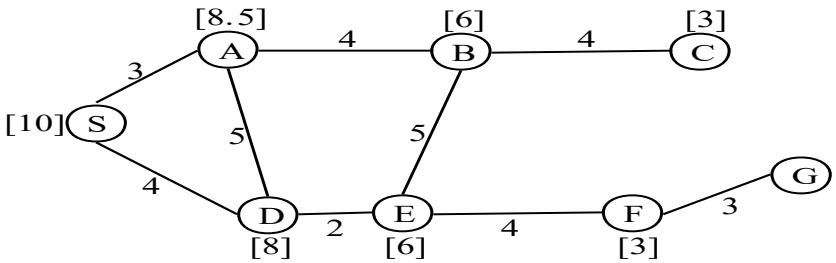


Using the problem defined above, find the shortest route from S to G; *that is S is the initial state and G is the goal state.*

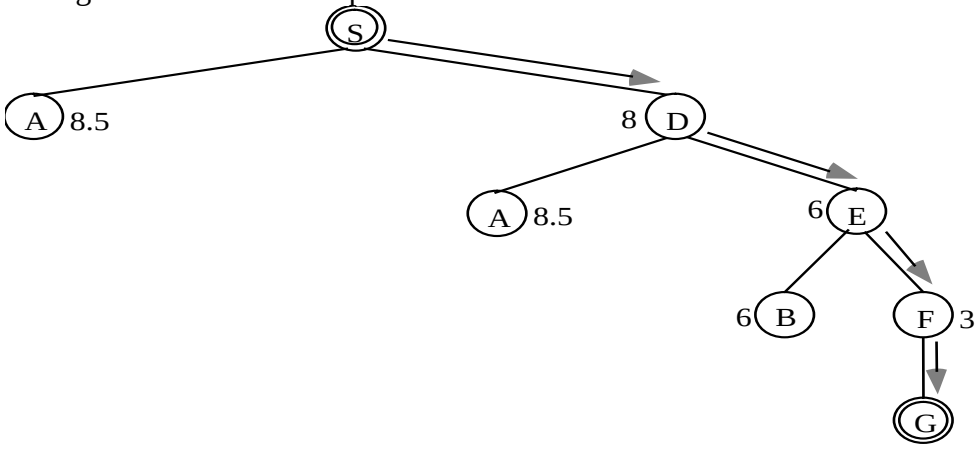


In summary, uniform cost search will find the cheapest solution provided that the cost of the path never decreases as we proceed along the path. If this requirement is not met then we never know when we will meet a negative cost. The result of this would be a need to carry out an exhaustive search of the entire tree.

ii). **Hill climbing: -** Hill climbing is depth-first search with a heuristic measurement that orders choices as nodes are expanded. It always selects the most promising successor of the node last expanded. For instance, one may consider that the most promising successor of a node is the one that has the shortest straight-line distance to the goal city G. In figure below, the straight line distances between each city and G is indicated in square brackets.



The hill climbing search from S to G proceeds as follows:



iii). **Greedy Search: -** The node closest to the goal state, as determined by an evaluation function *f(n) = h(n)*, is expanded first. Generally it selects node to expand that is believed to be closest (hence it's "greedy") to a goal node (i.e., smallest *f* value).

A Heuristic Function (h(n)) is a function that calculates the cost of reaching the goal state from the node n.

Issues:

- Finds solution quickly
- Doesn't always find the best solution, since it evaluates the immediate best choice, not the long term options.
- Can fall prey to false starts i.e expands a node that leads to a dead end (not complete).

iv). **The search algorithm A*: -** is a refinement of the greedy search due to the fact that it is neither optimal nor complete. The goal of the A* algorithm is to find a minimal cost-path joining the start node and a goal node. It uses the following evaluation function to estimate the promise of a node n:

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

where:

$g^*(n)$ estimates the minimum cost of a path from the start node to node n;

$h^*(n)$ estimates the minimum cost from node n to a goal.

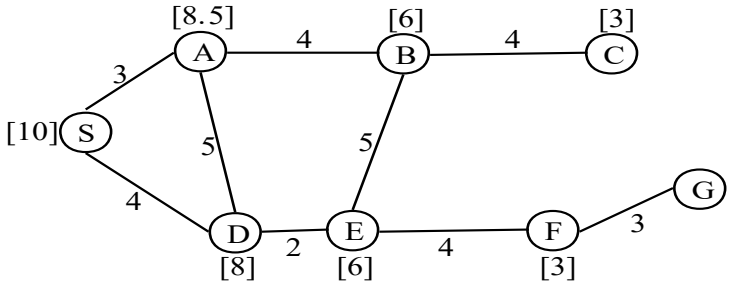
The node chosen for expansion is the one for which f* is minimum.

Notice that A* is very similar to the uniform-cost search, except that it uses a different function (which is a heuristic function) to estimate the promise of a node.

In fact, A* reduces to uniform-cost search if h*(n) is always considered to be 0.

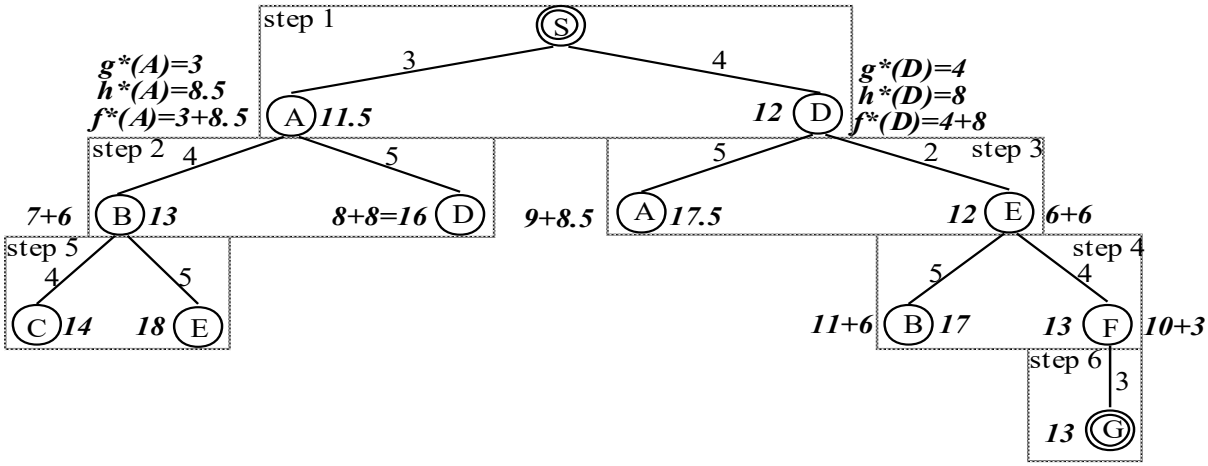
Example

Use A* algorithm to find the path from city S to city G by using the following functions:



In the above figure, the value of h* for each node is represented in square brackets. That is, h*(D) = 8.

Solution:



A* search of a path from S to G.