BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
WORK INTEGRATED LEARNING PROGRAMMES
Digital Learning
Part A: Content Design

Course Title	**Artificial and Computational Intelligence**
Course No(s)	**DSE**ZG557**
Credit Units	**5**
Credit Model	**1.25 - 1.5 - 2.25** **1.25 unit for class room hours, 1.5 unit for Tutorial, 2.25 units for student preparation.** **1 unit = 32 hours**
Course Authors	**S.P.Vimal, Rajavadhana P**
Version No	**4.0**
Date	**Aug 12, 2022**

Course Objectives

No	**Course Objective**
CO1	To provide a solid foundation for designing intelligent agents
CO2	Learn the representation and use of knowledge in inference-based problem solving approaches
CO3	Learn to apply probability theory to describe and model agents operating in uncertain environments
CO4	Learn the optimization models of computation and processing in real world application of intelligent agents

Text Book(s)

T1	Stuart Russell and Peter Norvig, “Artificial Intelligence – A Modern Approach”, Third Ed, Pearson Education, 2010

Reference Book(s) & other resources

R1	Ryszard S. Michalski, Jaime G. Carbonell and Tom M. Mitchell, “Machine Learning: An Artificial Intelligence Approach”, Elsevier, 2014
R2	Dan W Patterson, “Introduction to AI and Expert Systems”, Prentice Hall of India, New Delhi, 2010
R3	Elaine Rich and Kevin Knight, “Artificial Intelligence”, Tata McGraw Hill Publishing Company, New Delhi, 2003

Modular Content Structure

0. Introduction
    a. Artificial Intelligence: Foundations, Overview of Modern AI & Application Domains.
1. Introduction to Intelligent Agents: Notion of Agents and Environments, Rationality, Nature of Environments, Structure of Agents
2. Problem Solving Agent using Search:
    a. Problem Formulation, Uninformed Search Algorithms: Uniform cost Search, Depth Limited Search, Iterative Deepening Search – Informed Search Algorithms: Notion of Heuristics, Greedy best first search, A* search, Optimality of A*
    b. Heuristic Functions: Heuristic Accuracy & Algorithm performance, Admissible heuristics from relaxed problems, pattern databases & Experience
    c. Local Search Algorithms & Optimization Problems: Hill Climbing Search, Simulated Annealing, Local Beam Search, Evolutionary Algorithms - Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization
3. Game Playing:
    a. Searching to play games: Minimax Algorithm, Alpha-Beta Pruning
    b. Making imperfect real time decisions
4. Knowledge Representation using Logics:
    (**Pre-Reading:** Logics- Propositional, Predicate, TT-Entail, Theorem Proving)
    a. Logic Representation of a sample agent, Proof by resolution, DPLL Algorithm, Agents based on Propositional logic
    b. Overview of First Order Logic semantics, Example representation, Unification & Lifting, forward chaining, Backward Chaining, Resolution
5. Probabilistic Representation and Reasoning
    a. Inference using full joint distribution & Example, Knowledge representation using Bayesian Networks, semantics of Bayesian Networks, Representation of Conditional Independence using Bayesian Networks
    b. Exact Inference - by enumeration and variable elimination, Need for Approximate Inference - Direct Sampling
6. Reasoning over time
    a. Time and Uncertainty, Inference in temporal models
    b. Hidden Morkov Models, Algorithms: Filtering, Smoothing, Finding the most likely sequence, EM algorithms for Learning the parameters of HMM
7. Ethics in AI
    a. Explainable AI- Logically Explained Network, Explainable Bayesian Network

Learning Outcomes:

No	Learning Outcomes
LO1	Understand the environment and process of development to build intelligent agents
LO2	Identify heuristics to pursue goals in exponentially large search spaces.
LO3	Represent problem and derive reasoning using logical inferences
LO4	Apply probability theory to describe and model agents operating in uncertain environments
LO5	Analyze ways for agents to learn and improve their behavior.

Part B: Course Handout

Academic Term	**First Semester, 2022-23**
Course Title	**Artificial and Computational Intelligence**
Course No	**DSE**ZG557**
Lead Instructor	

Session #	List of Topic Title (from content structure in Part A)	References
1	(1) What is Artificial Intelligence: Acting Humanly, Thinking humanly, Thinking rationally, Acting Rationally (2) Foundations of AI (3) Brief Overview of Modern AI & Application Domains.	T1: 1.1 T1: 1.2, 1.4
2	(1) Intelligent Agents: Notion of Agents and Environments, Rationality (2) Nature of Environments, Structure of Agents	T1: Chapter 2
3	Problem Solving Agent: (1) Problem Formulation - Examples (2) Uninformed Search Algorithms: Uniform cost Search, Depth Limited Search, Iterative Deepening Search. (3) Notion of Heuristics (4) Informed Search Algorithms : Greedy best first search, A* search	T1: Chapter 3.1-3.4, 3.5.1, 3.5.2

4	Problem Solving Agent using Informed Search: (1) Optimality of A* Heuristic Functions: (1) Heuristic Accuracy & Algorithm performance (2) Admissible heuristics from relaxed problems, pattern databases. & Experience	T1: Chapter 3.5.2, 3.6
5	Local Search Algorithms & Optimization Problems (1) Hill Climbing Search (2) Simulated Annealing, (3) Local Beam Search	T1: Chapter 4.1
6	Local Search Algorithms & Optimization Problems (1) Genetic Algorithm	T1: Chapter 4.1, Research papers & web resources
7	Local Search Algorithms & Optimization Problems (1) Ant Colony Optimization (2) Particle Swarm Optimization	Research papers & web resources
8	Searching to play games: (1) Minimax Algorithm (2) Alpha-Beta Pruning (3) Making imperfect real time decisions	T1: Chapter 5.1 to 5.4
9	Logical Agent: (1) Logic Representation of a sample agent (2) DPLL Algorithm, Agents based on Propositional logic (3) Overview of First Order Logic semantics, Example representation	T1: Chapter 7.1, 7.2, 7.5.2, 7.5.3, 7.6.1, 8.1, 8.3.4
10	Inference in First Order Logic (1) Unification & Lifting (2) Forward chaining (3) Backward Chaining	T1: Chapter 9
11	Inference in First Order Logic (1) Resolution Probabilistic Representation and Reasoning (1) Inference using full joint distribution & Example (2) Knowledge representation using Bayesian Networks	T1: Chapter 9, 13, 14.1

12	Probabilistic Representation and Reasoning (1) Semantics of Bayesian Networks (2) Representation of Conditional Independence using BN	T1: 14.2, 14.3
13	Probabilistic Representation and Reasoning (1) Exact Inference - by enumeration and variable elimination (2) Need for Approximate Inference - Direct Sampling	T1: 14.4, 14.5
14	Reasoning over time (1) Time and Uncertainty (2) Inference in temporal models	T1: Chapter 15.1, 15.2
15	Reasoning over time (1) Hidden Markov Models (2) Learning HMM Parameters using EM Algorithm (3) Applications of HMM	T1: Chapter , 15.3, 20.3-20.3.3
16	Ethics in AI: (1) Explainable AI- Logically Explained Network, Explainable Bayesian Network	Research papers & web resources

Detailed Plan for Lab work/Design work

Lab No	Lab Objective	Lab Sheet Access URL	Content Reference
1	Implementing Uninformed Search Algorithms	NA	Module #2
2	Implementing Informed Search Techniques	NA	Module #2
3	Implementing Local Search Techniques	NA	Module #2
4	Implementing adversarial search techniques	NA	Module #3
5	Representing knowledge using logics and performing reasoning	NA	Module #4
6	Experimenting with Bayesian Networks and Inferencing	NA	Module #5
7	Experimenting with HMM	NA	Module #6

Case studies: Detailed Plan

Case study No	Case study Objective	Case study Sheet Access URL
-	-	-

Evaluation Scheme:
Legend: EC = Evaluation Component; AN = After Noon Session; FN = Fore Noon Session

No	Name	Type	Duration	Weight	Day, Date, Session, Time
EC-1	Quiz-I	Online	-	5%	TBA
	Assignment -1	Take-Home	-	13%	TBA
	Assignment -2	Take-Home	-	12%	TBA
EC-2	Mid-Semester Test	Open/Closed Book	1.5 hours	30%	TBA
EC-3	Comprehensive Exam	Open/Closed Book	2.5 hours	40%	TBA

Note:
Syllabus for Mid-Semester Test (Open/Closed Book): Topics in Contact Sessions: 1 to 8
Syllabus for Comprehensive Exam (Open/Closed Book): All topics

Important links and information:
Elearn portal: https://elearn.bits-pilani.ac.in
Students are expected to visit the Elearn portal on a regular basis and stay up to date with the latest announcements and deadlines.
Contact sessions: Students should attend the online lectures as per the schedule provided on the Elearn portal.
Evaluation Guidelines:

1. EC-1 consists of two Assignments and one Quiz. Students will attempt them through the course pages on the Elearn portal. Announcements will be made on the portal, in a timely manner.
2. For Closed Book tests: No books or reference material of any kind will be permitted.
3. For Open Book exams: Use of books and any printed / written reference material (filed or bound) is permitted. However, loose sheets of paper will not be allowed. Use of calculators is permitted in all exams. Laptops/Mobiles of any kind are not allowed. Exchange of any material is not allowed.
4. If a student is unable to appear for the Regular Test/Exam due to genuine exigencies, the student should follow the procedure to apply for the Make-Up Test/Exam which will be made available on the Elearn portal. The Make-Up Test/Exam will be conducted only at selected exam centres on the dates to be announced later.

It shall be the responsibility of the individual student to be regular in maintaining the self-study schedule as given in the course handout, attend the online lectures, and take all the prescribed evaluation components such as Assignment/Quiz, Mid-Semester Test and Comprehensive Exam according to the evaluation scheme provided in the handout.

Artificial & Computational Intelligence

AIML CLZG557

M1 : Introduction

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Agenda

- Course Administration
- Getting Started (with some definitions)
- Course Overview with example

Course Administration

About the course

- Focus on
  - principles of artificial intelligence
  - concepts, algorithms involved in building rational agents
  - topics covered like
    - (informed and uninformed) search & applications
    - (logical & probabilistic) knowledge representation
    - (logical & probabilistic) Reasoning & applications
  - topics not-covered like
    - Formal introduction to machine learning algorithms, neural networks etc., are covered as a ML course is running in parallel, Deep neural networks, which are part of AI as well.
    - Hardware aspects of the Design

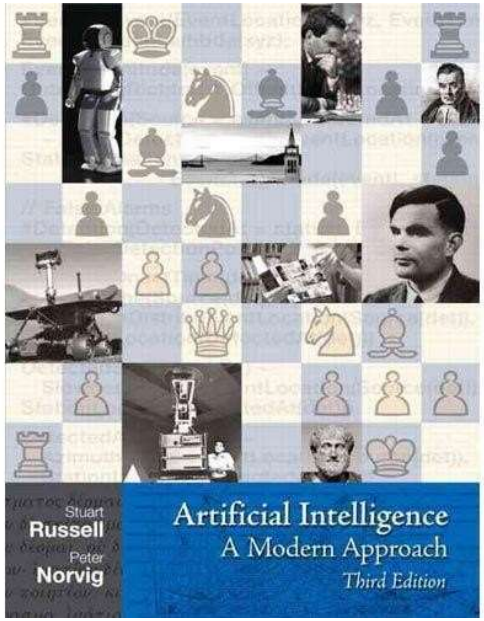
Course Outline

- **Pedagogy**
  - Weekly online live sessions
  - Webinars on lab implementation
  - Assignment:
    - 1 Quiz-5%,
    - 2 Assignments- 25%

- **Lab Modules**
  - Supported by 6 lab capsules for practical implementation and better understanding of the concepts learned in the live lecture sessions.

About the course

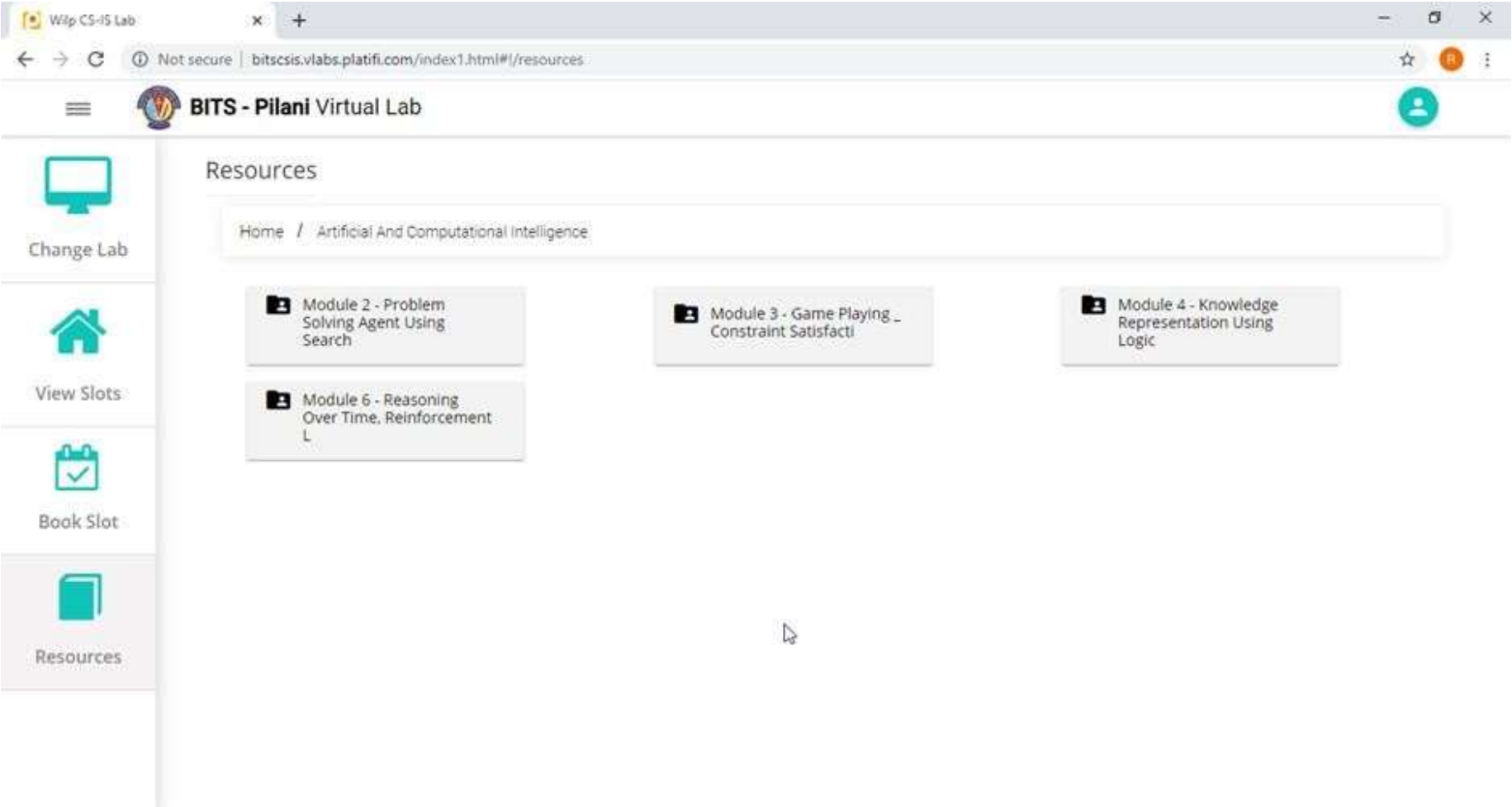
Text Book



Exercises : In Python & its libraries

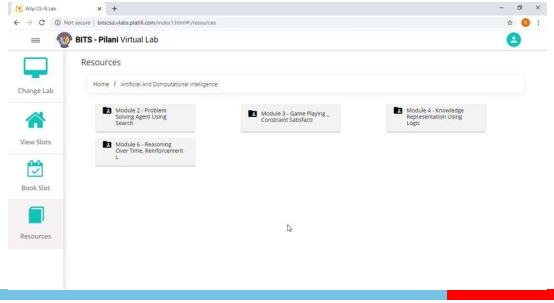
Evaluation : 25% Assignment + 5% Quiz + 30% Mid Semester + 40% End Semester

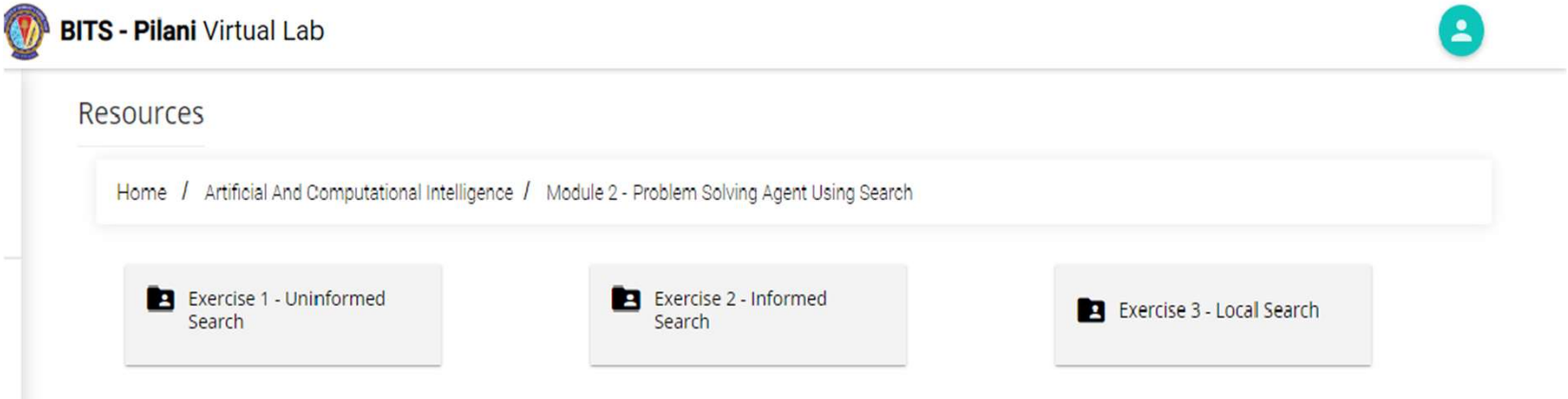
About Labs

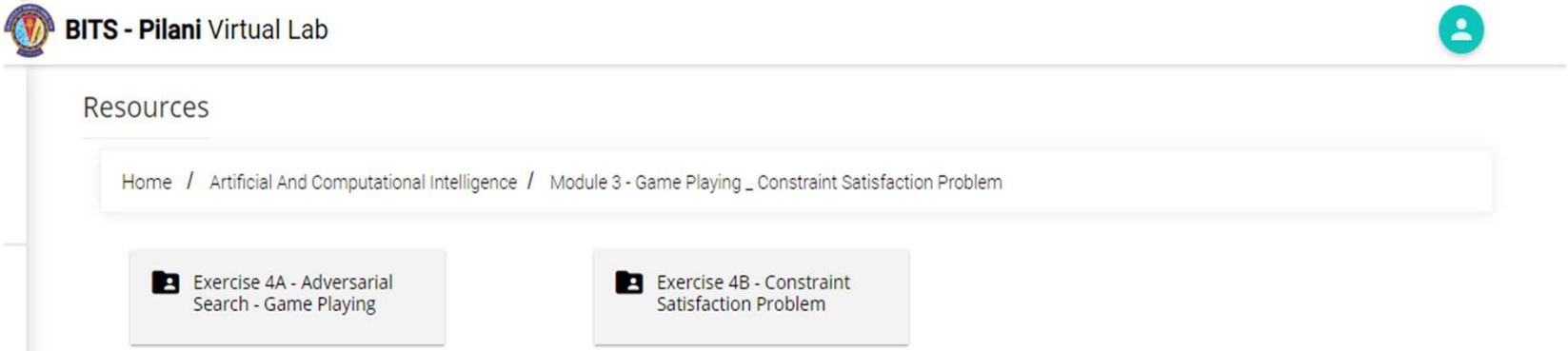


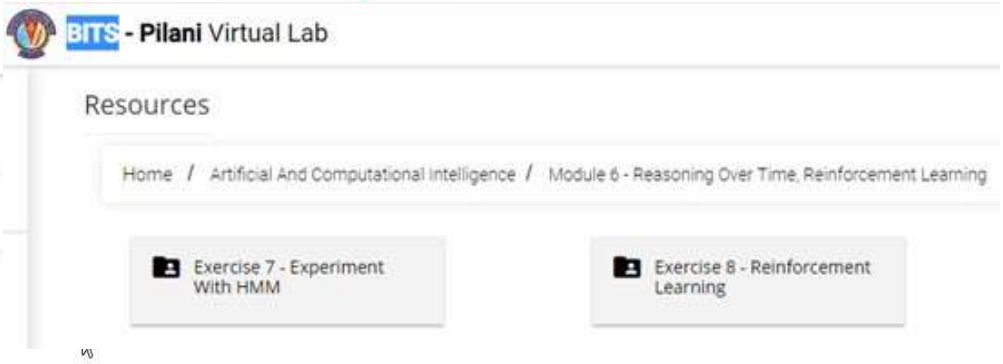
Exercises : In Python & its libraries

About Labs



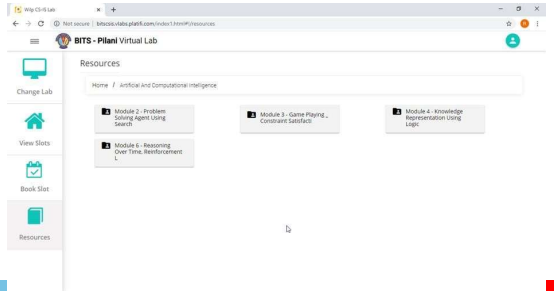


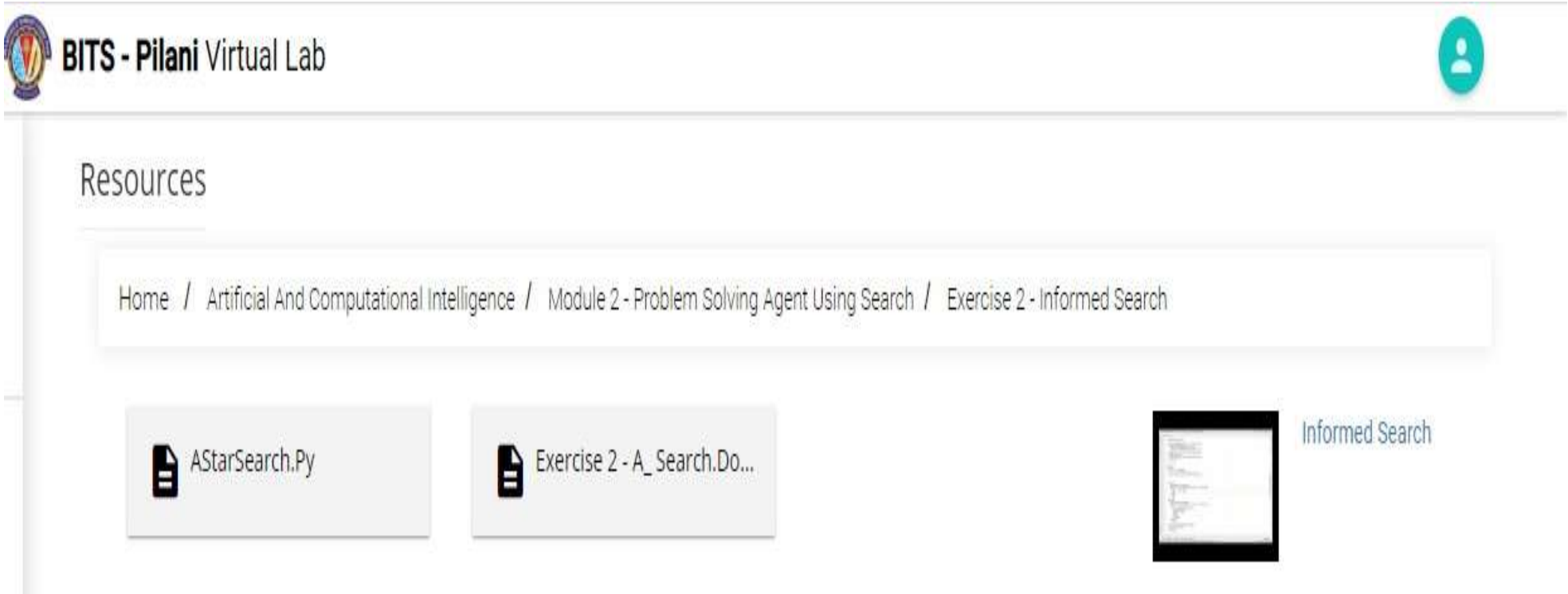




Exercises : In Python & its libraries

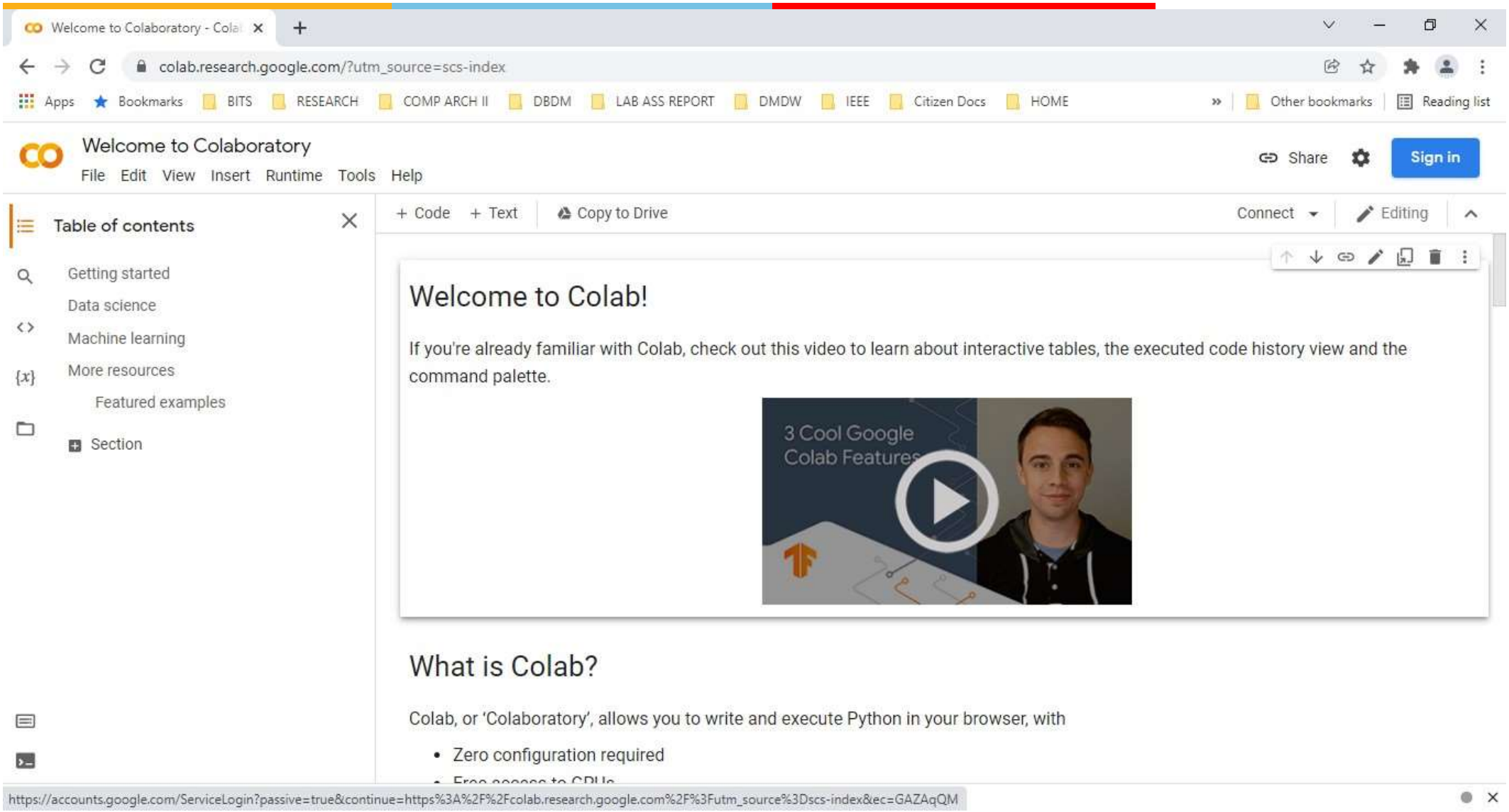
About Labs





Exercises : In Python & its libraries

About Labs



Exercises : In Python & its libraries

Artificial Intelligence

- Term coined by, *John McCarthy* (1955) & *Dartmouth Summer Research Project on Artificial Intelligence* (1956)

On September 2, 1955, the project was formally proposed by McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon. The proposal is credited with introducing the term 'artificial intelligence'.

The Proposal states[7]

> "We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer."

Artificial Intelligence

- Term coined by, *John McCarthy* (1955) & *Dartmouth Summer Research Project on Artificial Intelligence* (1956)

On September 2, 1955, the project was formally proposed by McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon. The proposal is credited with introducing the term 'artificial intelligence'.

The Proposal states[7]

> We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

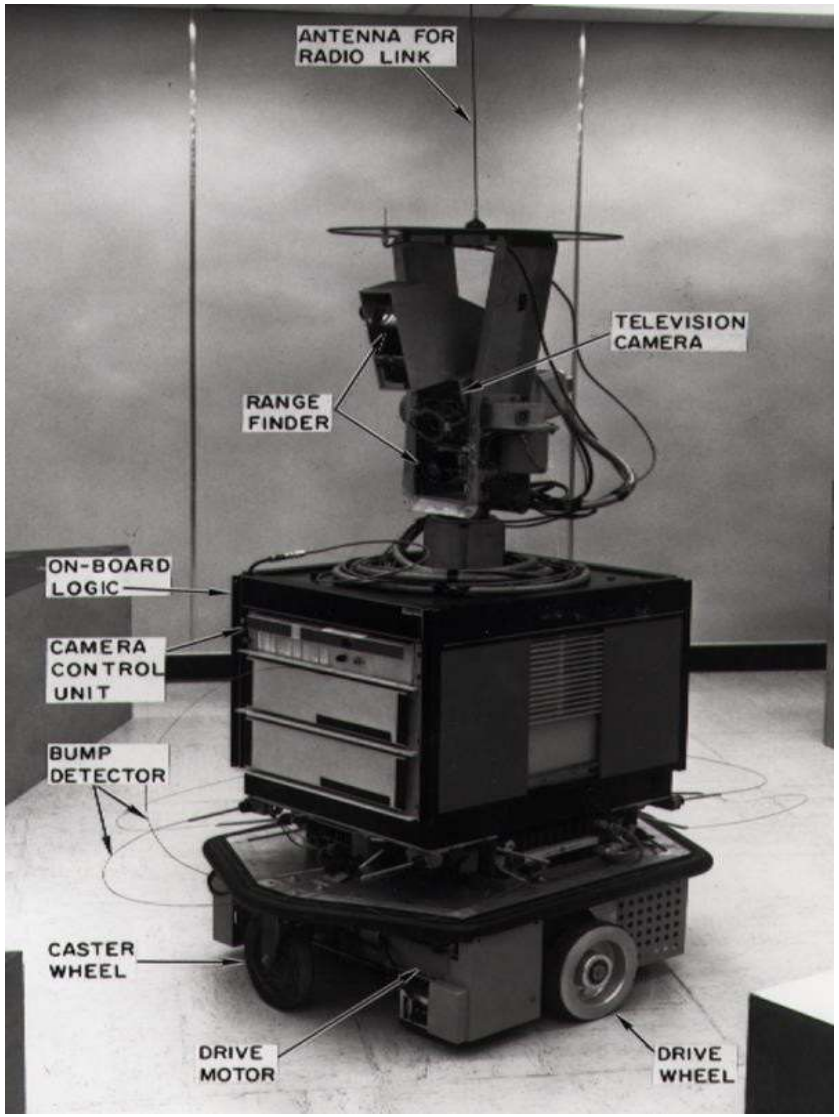
https://en.wikipedia.org/wiki/Dartmouth_workshop [01 June, 2019]

Larger Intent, Dream, Overconfidence ...

Some Early successes of Dartmouth

Many key projects were initiated after Dartmouth summer project.

Shakey robot - First mobile robot to perceive environment & reason about surroundings, actions - Introduced A* algorithm to find paths - Hough Transform for image analysis - Used Lisp for programming - visibility graph used for finding shortest paths in the presence of obstacles...



A brief history of AI

1950's

John McCarthy, Newell, Simon, Alan

GPS

LISP

Advice Taker – 1st complete AI system

1960's & 70's

Minsky

MYCIN

PROLOG

1980's

John Hopfield, McClelland

DEC- Digital Equipment Corp

Neural Nets, HMM

Mining, Bayesian Nets

Parallel Distributed Proc

1990's ....

Intelligent Agent based Architectures

How is AI unique or in other words different from Applied Math?

Some Early successes of Dartmouth

DENDRAL -

Attempted to encode the domain expertise in molecular biology as an expert system

Led to the creation of expert systems for various other domain, including medical.

A milestone worship in the history of AI !!!

Perspectives of AI

Definitions

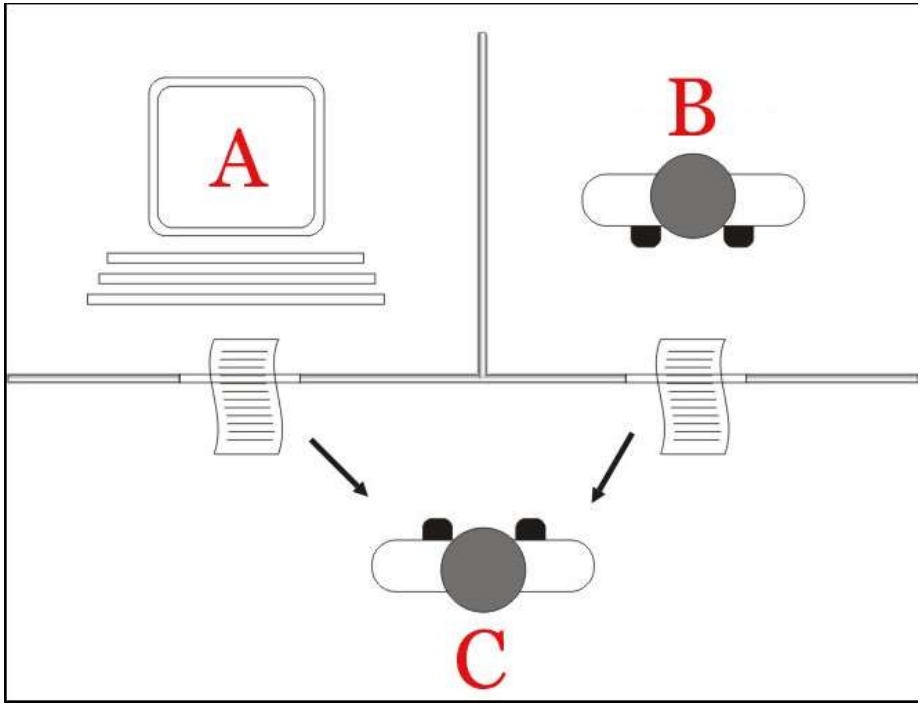
	Thought / Reasoning	Acting
Human Performance	THINKING HUMANLY "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning, ... " (Bellman, 1978)	ACTING HUMANLY "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)
Rational Performance	THINKING RATIONALLY "The study of computations that make it possible to perceive, reason, and act" (Winston, 1992)	ACTING RATIONALLY "Computational intelligence is the study of the design of intelligent agents" (Poole et al., 1998)

Acting Humanly

Turing Test Approach

- *Turing Test* & *Total Turing test* (operational test to determine an entity is intelligent / not) [50's]

- Skills necessary to pass these tests
  - NLP, Knowledge Representation, Automated Reasoning, ML + Computer Vision & Robotics(for total turing test)



Pictorial Representation of Turing Test from https://en.wikipedia.org/wiki/Turing_test

Acting Humanly

Turing Test Approach

Some Definitions of AI:

"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)

"The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)

Thinking Humanly

Cognitive Modelling Approach

- How do we capture human thinking to implement?
  - Introspection
  - Psychological Experiments
  - Brain Imaging

- System : *"General Problem Solver"* (*Newell and Simon, 1961*)
  - Designed to work as a universal problem solver
  - Problems represented by horn clauses
  - First AI Machine which has KB + Inference separation
  - Authors focus on this is on comparing the trace of its reasoning steps to traces of human subjects solving the same problems

- Growth of Cognitive science and AI supports each other

Passing the Turing Test

Transcript of a chat

EUGINE - a thirteen-year-old Ukrainian boy, chats

JUDGE: Hello.
EUGENE: Hello, I'm really glad to have the chance to chat with you! My guinea pig Bill sends his regards too!
JUDGE: Is Bill a male or a female?
EUGENE: Ask Bill personally, please.
JUDGE: Well I'd rather talk to you. What is your name?
EUGENE: Call me Eugene. I am glad to talk to you!
JUDGE: My name is Jane and I am female. How about you? What's your gender?
EUGENE: I'm a male. A "guy' I'd say.
JUDGE: Pleased to meet you Eugene. What's the weather like where you are?
EUGENE: Let's get on with our conversation!
JUDGE: Don't you like talking about the weather?
EUGENE: All these talks about weather is a waste of time.
JUDGE: What would you like to discuss?
EUGENE: I don't know. Better tell me more about yourself!

Passing the Turing Test

- 2014 - Royal Society (London) - Sixteenth Anniversary of Alan Turing -
- Chabot - Eugene Goostman - Pretended to be a thirteen-year-old Ukrainian boy
  - Passed the turing test for the first time
  - 10/30 Judges believed the response is from human

- *Turing predicted in 50 years time, computers can be programmed to play imitation game in which an average interrogator fails to identify the machine 70% time in a 5 mins questioning*

Thinking Humanly

Cognitive Modelling Approach

Some Definitions of AI:

"The exciting new effort to make computers think . . . machines with minds, in the full and literal sense." (Haugeland, 1985)

"[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)

Thinking Rationally

"Laws of Thought" Approach

- Invention of Formal Logic, Greek Philosopher **Aristotle**, Third century BC.
- Introduced syllogisms, providing argument structures

In all boring classes, students sleep

It is a boring class

Students sleep in this class [Are you ?]

- Field of Logics gave rise to codifying rational thinking
    - When elements are '*things*', we reason about things

Hurdles to the idea : (1) Not everything can be logically coded (2) no provably correct action at a moment (3) Exhaustive computational resources

Acting Rationally

The Rational Agent Approach

- An agent is an entity that perceives and acts

  This course is about designing rational agents

- Abstractly, an agent is a function from percept histories to actions:

  [f: P* → A]

- For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance
- Computational limitations make perfect rationality unachievable
- Design best program for given machine resources

Acting Rationally

The Rational Agent Approach

- Rational behaviour: doing the *right thing*
- The *right thing:* that which is expected to maximize goal achievement, given the available information
- Rational behaviour is not just about correct inference / thinking, skills needed to pass turing test etc.

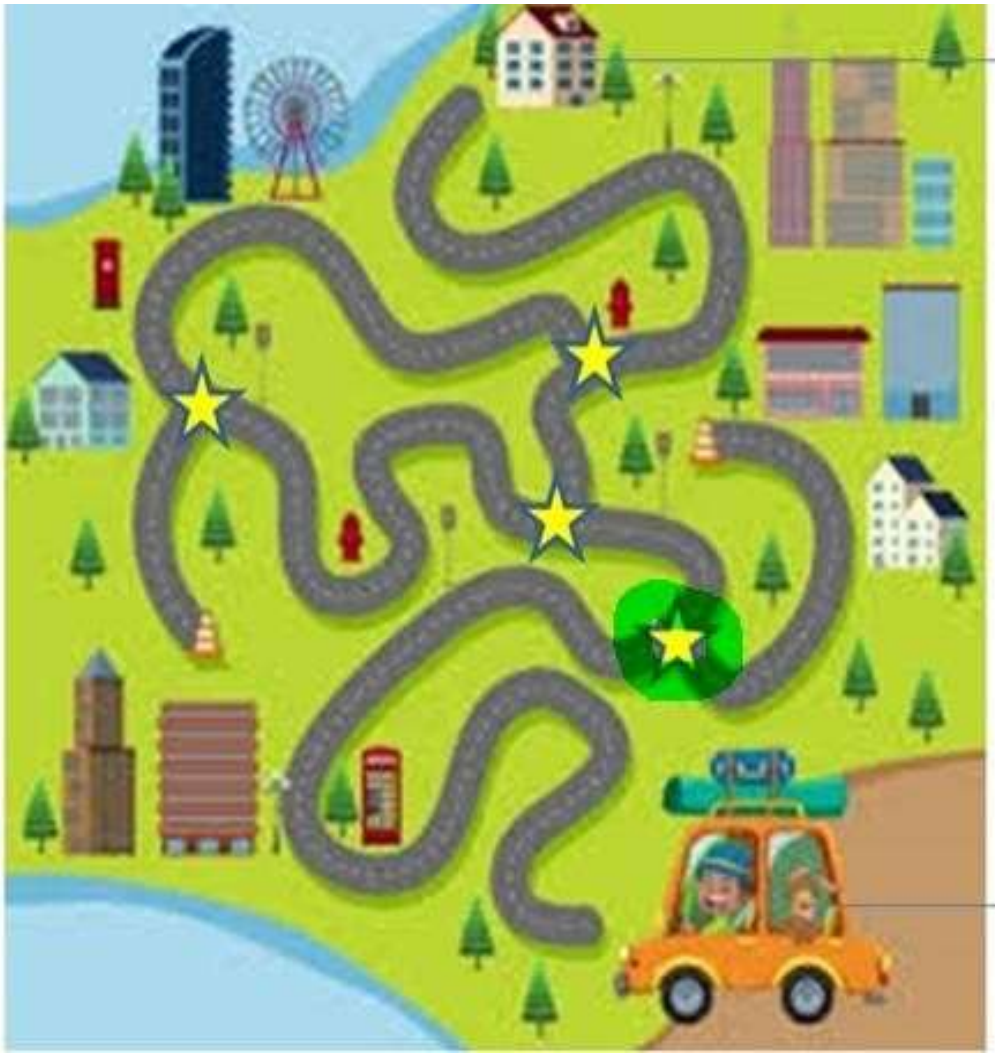
(adv) : More General - Correct inference is just a thing

(adv) : More amenable for scientific developments, as the rational behaviour is better defined than human thinking and behaviour

Definitions

Thinking Humanly	Thinking Rationally
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Traveller's Problem



Destination - Fixed **Goal**

Source

Traveller's Problem

AI in HealthCare

Lyrebird's Project Re-Voice

AI in Culinary Field

Spyce

AI in NLS
IBM Watson

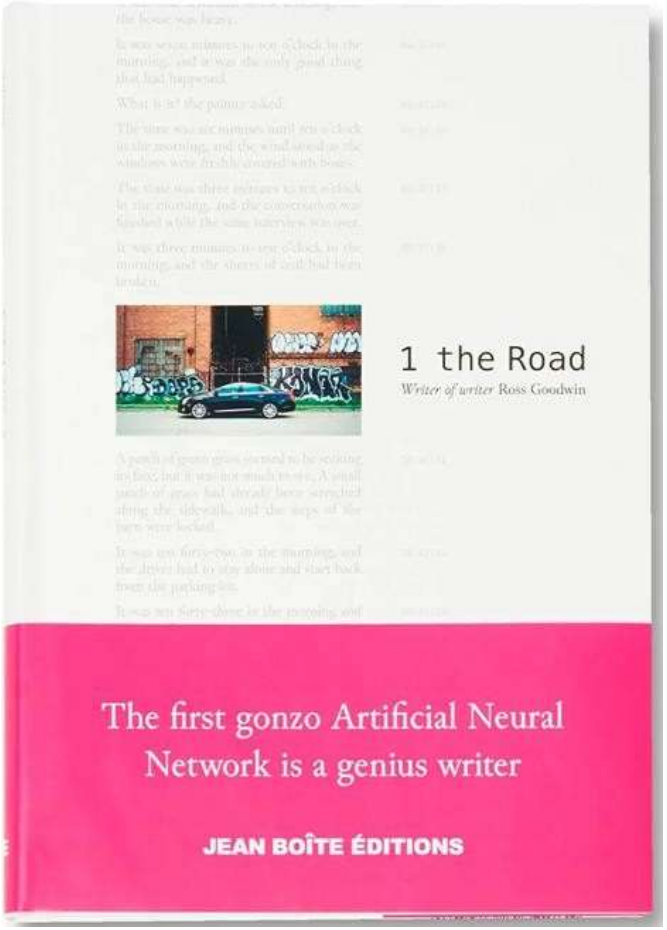
Computer Vision
NLP
ML
Speech Recognition
Automation

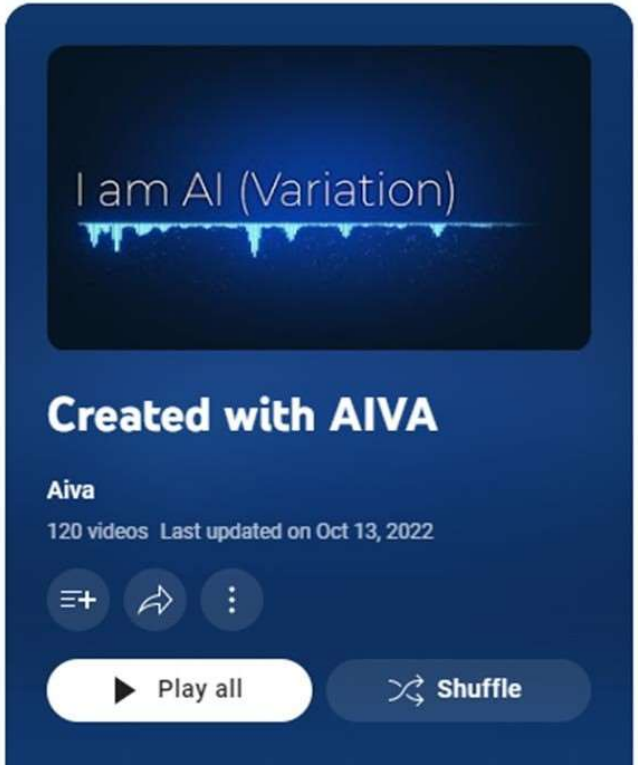
AI in Transportation

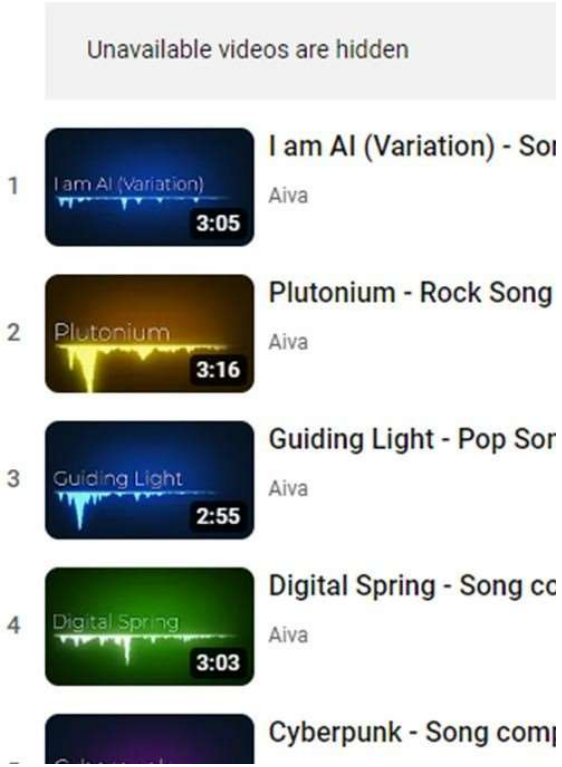
AI in HCI

Google Map Navigation Assistant

AI in Literacy & Music







Application Domain

(Additional Notes added from the textbook for self read)

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

- Can formal rules be used to draw valid conclusions?
- How does the mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Aristotle (384–322 B . C .) : first to formulate precise set of laws to govern rational part of brain

Ramon Lull (d. 1315) : useful reasoning could actually be carried out by a mechanical artifact

Hobbes (1588–1679) : "we add

and subtract in our silent thoughts."

Leibniz (1646–1716) : Built a mechanical device intended to carry out operations on concepts rather than numbers

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Some *'isms* on the working of minds :

Rationalism - Correct Reasonings (Aristotle, Descartes …)

Dualism - A part of the human mind (or soul or spirit) that is outside of nature

Materialism - Alternative to dualism - holds that the brain's operation according to the laws of physics constitutes the mind

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Obtaining Knowledge

David Hume's (1711–1776) : First principles of induction

Logical positivism- Rudolf Carnap : Every knowledge obtained has a logical connection

Carnap (1905–1997) : A book "*The Logical Structure of the World*" (1928) defined an explicit computational procedure for extracting knowledge from elementary experiences

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Connection between knowledge and action:

Aristotle - (in *De Motu Animalium*) that actions are justified by a logical connection between goals and knowledge of the action's outcome

```
I need covering;
a cloak is a covering.
I need a cloak.
What I need, I have to make;
I need a cloak.
I have to make a cloak.
And the conclusion, "I have to make a cloak" is an action
```

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

- What are the formal rules to draw valid conclusions?

George Boole (1815–1864) : Propositional Logic

Gottlob Frege (1848–1925): First order logic

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

- What can be computed?

Kurt Gödel (1906–1978) : In any formal theory as strong as Peano arithmetic #(the elementary theory of natural numbers), there are true statements that are undecidable in the sense that they have no proof within the theory

Computability, tractability, NP-completeness

Probability theory & inference mechanisms

- https://en.wikipedia.org/wiki/Peano_axioms

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

- How should we make decisions so as to maximize payoff?

Utility / preferred outcomes

Decision theory -Probability & utility theory

Game theory

- How to make decisions when payoffs are not immediate?
  - MDP

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

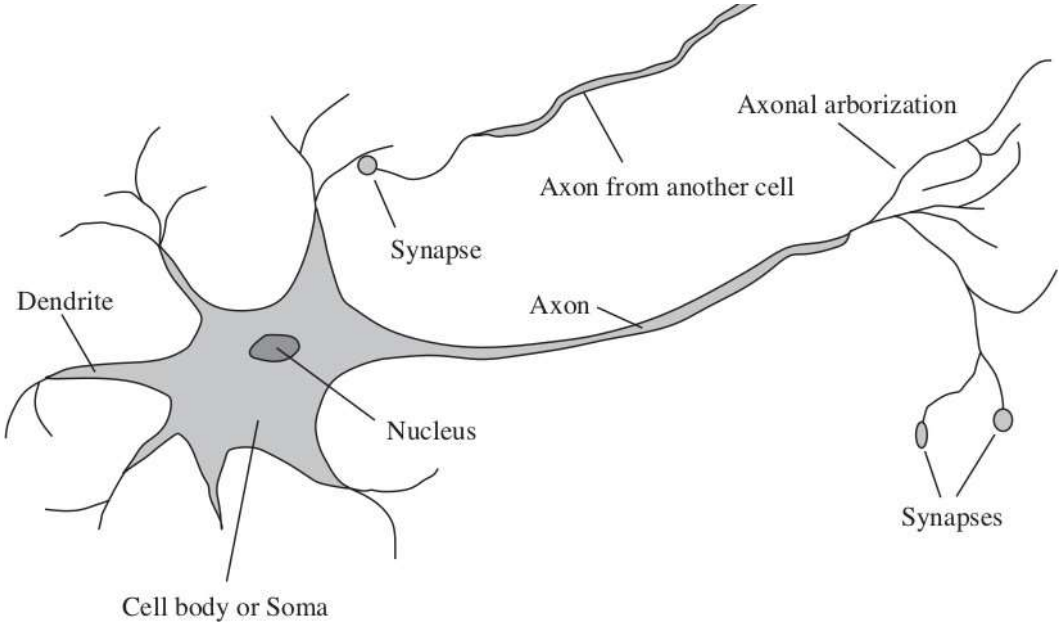
Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

How do brains process information?

- Study of the nervous system / brain
- How does brain enables thoughts - Mystery Still

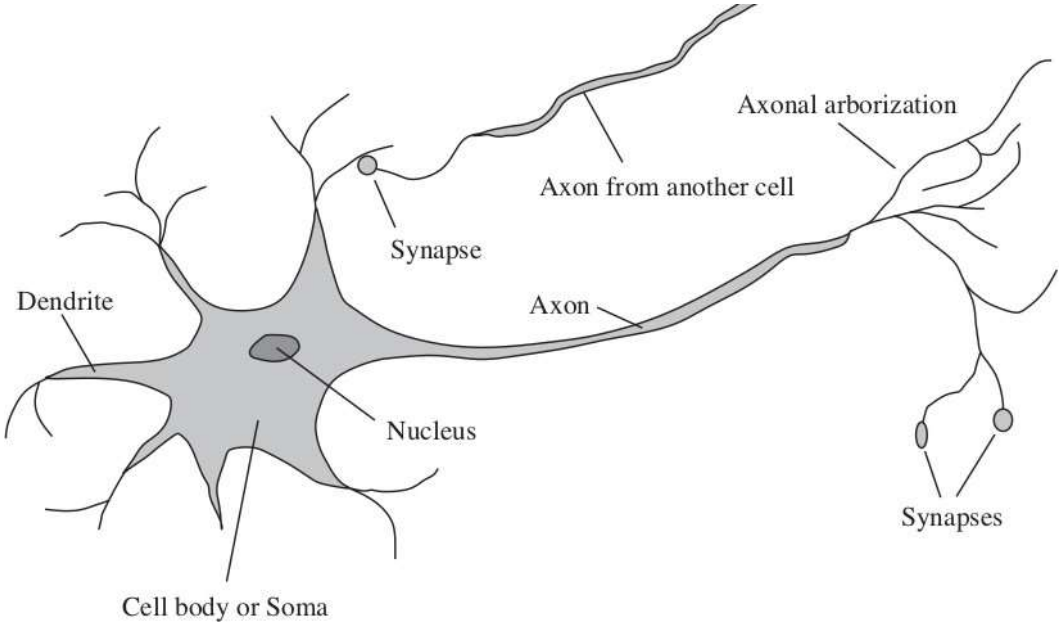
Aristotle , "*Of all the animals, man has the largest brain in proportion to his size*"



Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

	Supercomputer	Personal Computer	Human Brain
Computational units	10^4 CPUs, 10^{12} transistors	4 CPUs, 10^9 transistors	10^{11} neurons
Storage units	10^{14} bits RAM	10^{11} bits RAM	10^{11} neurons
	10^{15} bits disk	10^{13} bits disk	10^{14} synapses
Cycle time	10^{-9} sec	10^{-9} sec	10^{-3} sec
Operations/sec	10^{15}	10^{10}	10^{17}
Memory updates/sec	10^{14}	10^{10}	10^{14}



Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

How do humans and animals think and act?

- *Cognitive Psychology* - Brain as an information-processing device
- Two months after the dartmouth workshop, a workshop in MIT gave birth to *Cognitive Science*
  - George Miller, Noam Chomsky, Allen Newell and Herbert Simon - roles of computer models to address the psychology of memory, language, and logical thinking, issues..

"*a cognitive theory should be like a computer program*" (Anderson, 1980);

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Computers & Programming Languages

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

Control theory

- Deals with the behaviour of dynamic systems
  - behaviour must ensure the error between the current state and goal state is minimized
- Cybernetics - Book by Wiener
  - (Norbert Wiener, 1948) : Scientific study of control and communication in the animal and the machine
- Ashby's Design for a Brain (1948, 1952):
  - Intelligence could be created by the use of homeostatic devices containing appropriate feedback loops to achieve stable adaptive behavior
  - Led to the idea of *design of systems that maximize an objective function over time*

Areas Contributing to AI

Philosophy
Mathematics
Economics
Neuroscience
Psychology
Computer Engineering
Control theory, Cybernetics
Linguistics

How does language relate to thought?

Verbal Behavior (1957, B. F. Skinner) :

- Behaviorist approach to language learning
- Reviewed by Noam Chomsky
  - criticised lack of notion of creativity in language

Syntactic Structures (1957, Noam Chomsky)

- Computational linguistics / natural language processing as a part of AI
  - Understanding a language is realized as more complex than ever
  - Context, subject matter knowledge complicated it further
  - Representing language consumed volume of work done in NLP, in early times

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Course Outline

- **In this course, you will learn :**
  - a solid foundation for designing intelligent agents
  - to represent and use the knowledge learnt for inferencing
  - to model agents operating in uncertain environments
  - optimization models of computation and processing in real world application
- **Modules :**
  - Problem Solving Agent using Search
  - Game Playing
  - Probabilistic Representation and Reasoning
  - Reasoning over time

Required Reading: AIMA - Chapter #1

AIMA is the first prescribed text book

Thank You for your active participation

Note : Some of the slides are adopted from AIMA TB materials

Next Class Plan

- Agent Design
- Environment
- Agent Architecture
- Problem Solving Agent Formulation

Artificial & Computational Intelligence

AIML CZG557

M1 : Introduction
&
M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

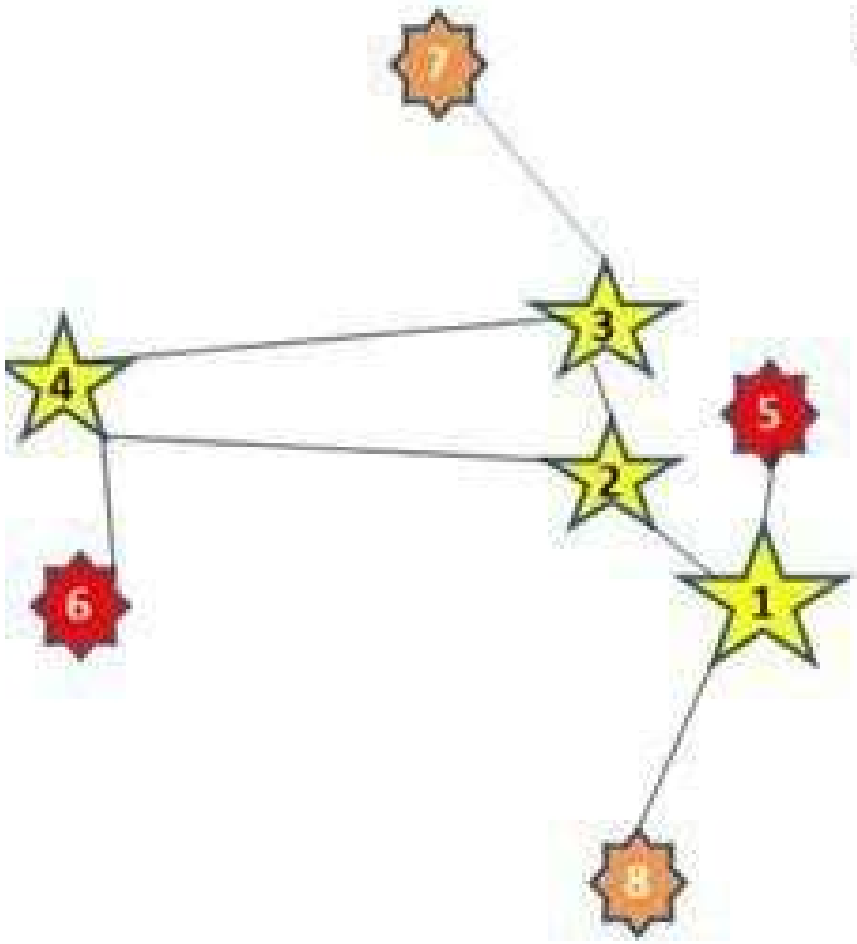
BITS Pilani
Pilani Campus

Course Plan

M1	Introduction to AI
M2	Problem Solving Agent using Search
M3	Game Playing
M4	Knowledge Representation using Logics
M5	Probabilistic Representation and Reasoning
M6	Reasoning over time
M7	Ethics in AI

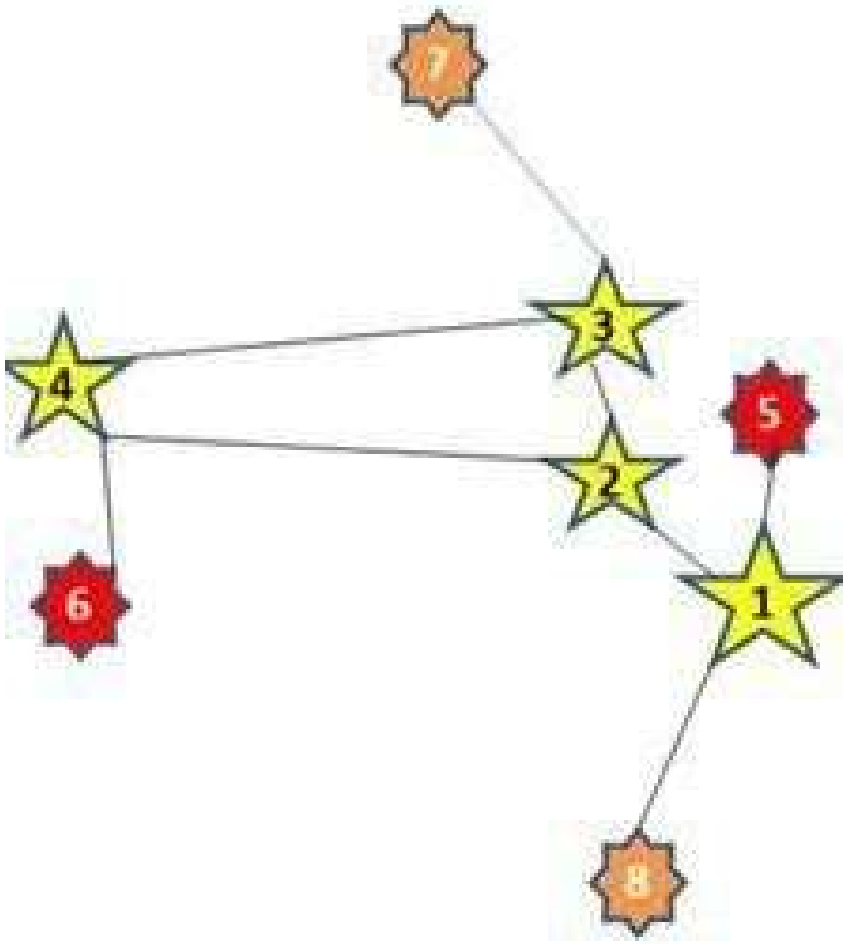
Traveller's Problem

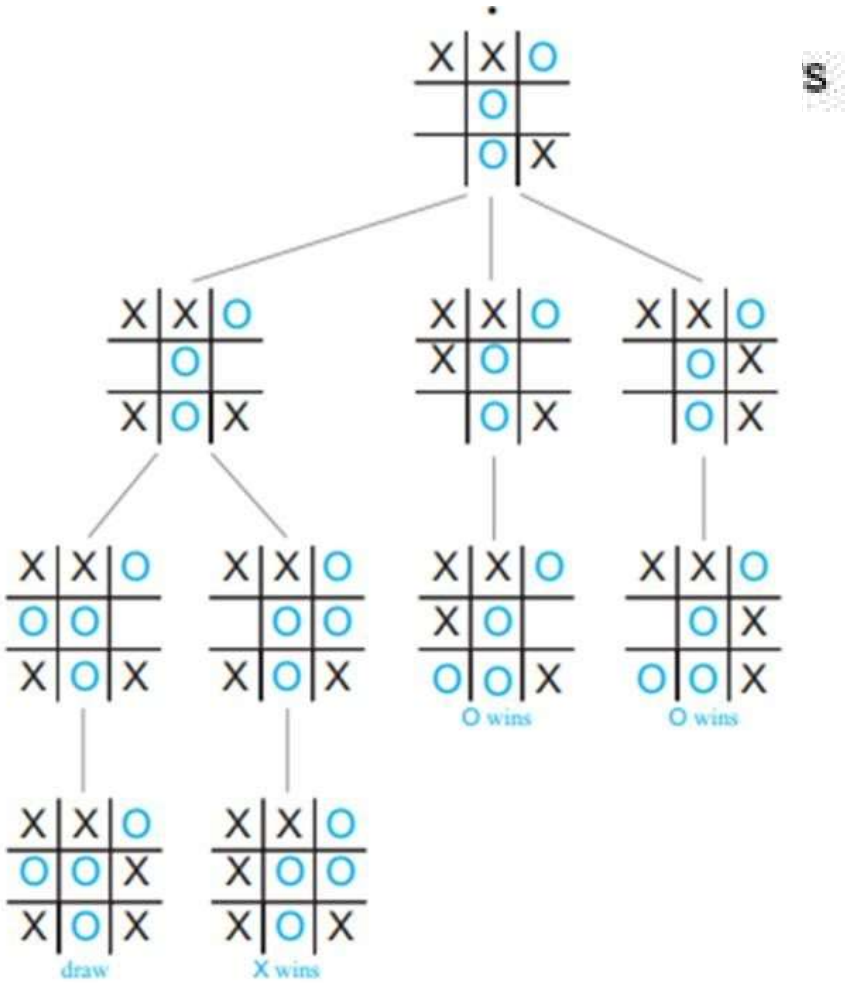
Traveller's Problem



Sensors → Environment → Actuators

Traveller's Problem





Rational Agents

Rational Agent

Design Principles & Techniques

	Thought / Reasoning	Acting
Human Performance	THINKING HUMANLY "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning, ... " (Bellman, 1978)	ACTING HUMANLY "The art of creating machines that perform functions that require intelligence when performed by people" (Kurzweil, 1990)
Rational Performance	THINKING RATIONALLY "The study of computations that make it possible to perceive, reason, and act" (Winston, 1992)	ACTING RATIONALLY "Computational intelligence is the study of the design of intelligent agents" (Poole et al., 1998)

Acting Rationally

The Rational Agent Approach

- An agent is an entity that perceives and acts

This course is about designing rational agents

- Abstractly, an agent is a function from percept histories to actions: [f: P* → A]
- For any given class of environments and tasks, we seek the agent (or class of agents) with the best performance
- Computational limitations make perfect rationality unachievable
- Design best program for given machine resources

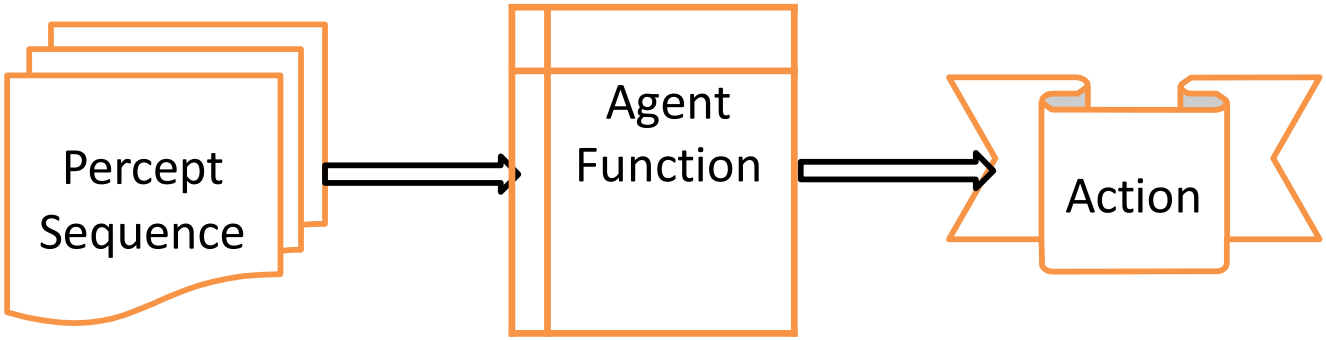
Properties of Rational Agent

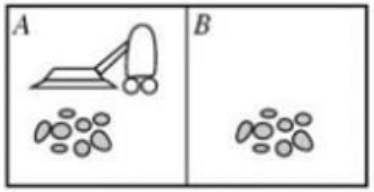
- Omniscience : Expected Vs Actual Performance
- Learning Capability : Apriori Knowledge
- Autonomous in decision making: An agent is autonomous if its behaviour is determined by its own experience (with ability to learn and adapt)

Intelligent Agent

Rational Agent is one that acts to achieve the best outcome or the best expected outcome even under uncertainty

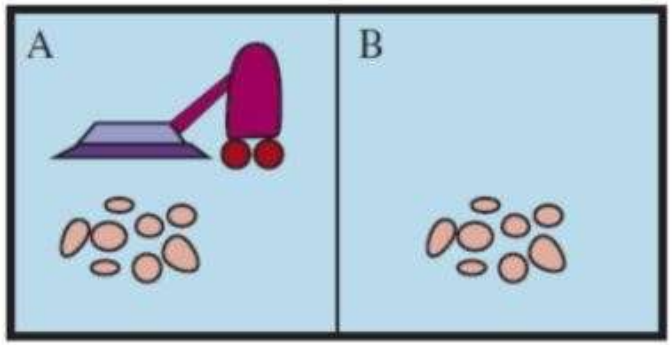
Maps / Tabulated / Programmed





Percept sequence	Action
[A,Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean],[A, Clean]	Right
[A, Clean],[A, Dirty]	Suck
...	...

Intelligent Agent



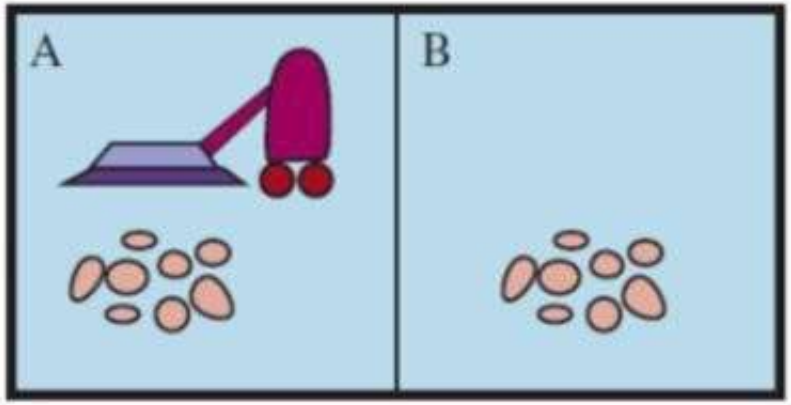
- Percepts: location and contents, e.g., [A , Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

Performance measure: An objective criterion for success of an agent's behaviour

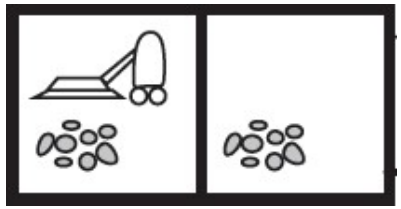
E.g., performance measure of a vacuum-cleaner agent
» amount of dirt cleaned up
» amount of time taken
» amount of electricity consumed
» amount of noise generated, etc.

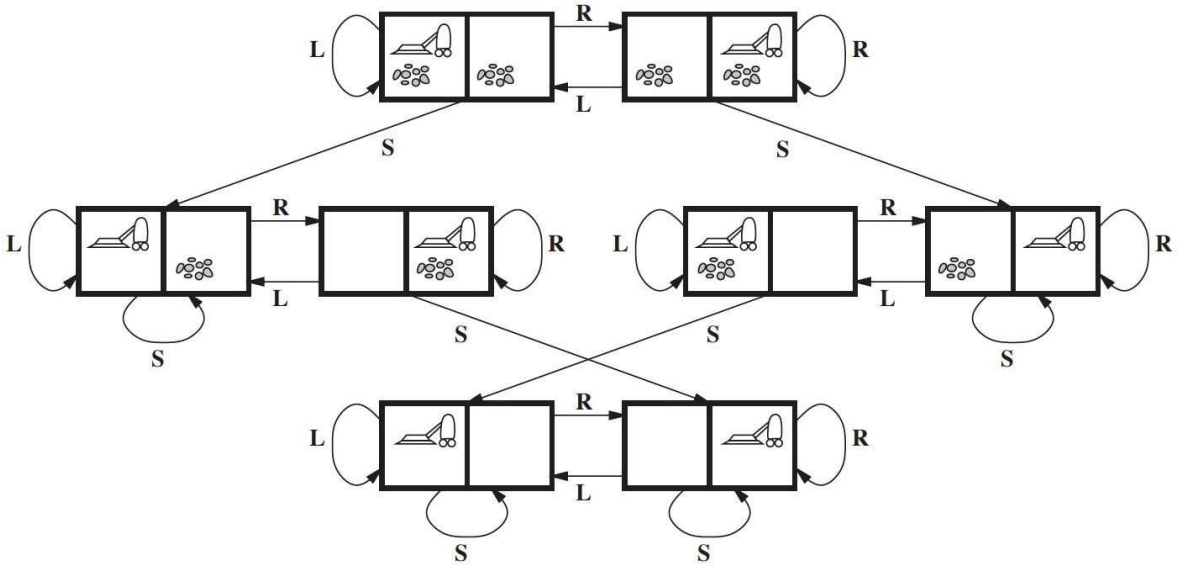
Intelligent Agent

Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Clean], [A, Clean]	Right
[A, Clean], [A, Dirty]	Suck
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮



Vacuum World Problem



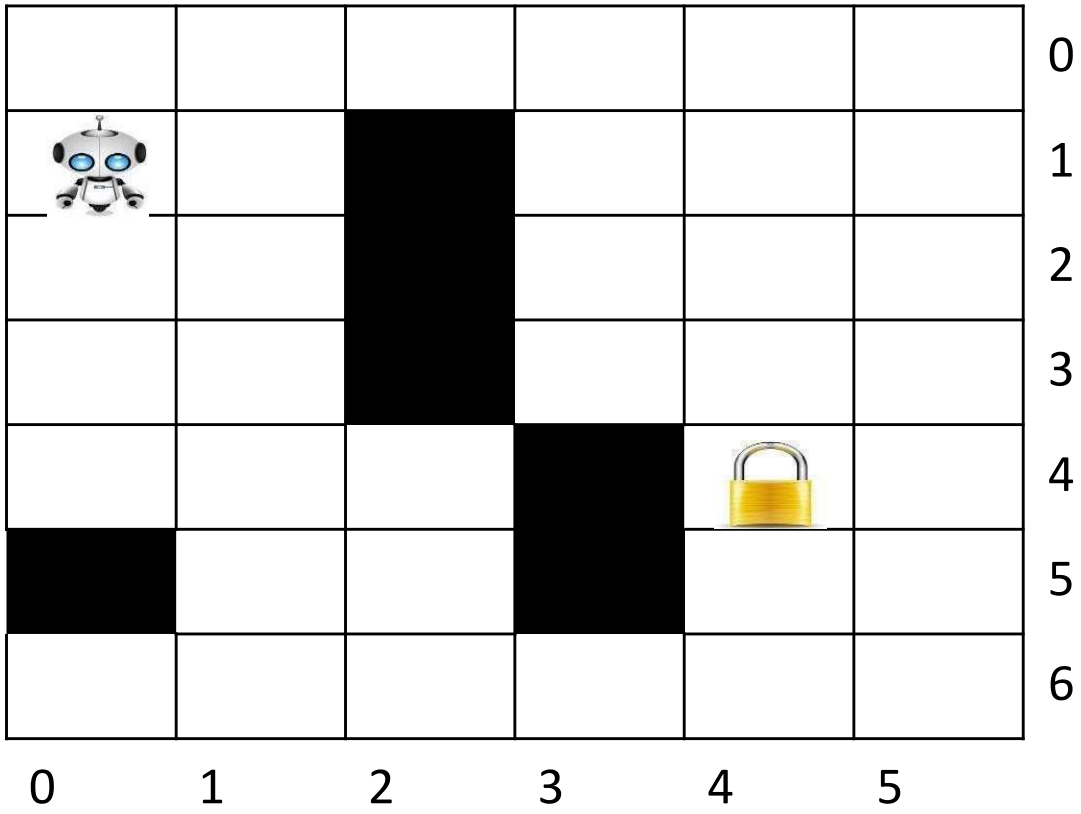


PEAS Environment

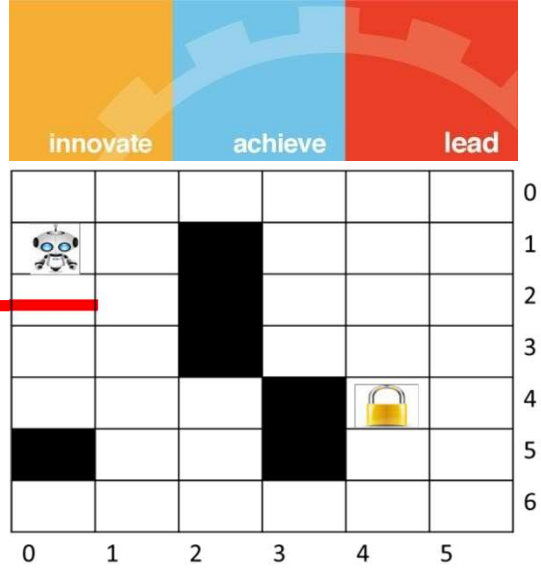
Design on what an application wants the agent to do in the environment

Agent	Performance	Environment	Sensors	Actuators
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Keyboard entry of symptoms, findings, patient's answers	Display of questions, tests, diagnosis, treatments, referrals
Satellite Image analysis system	Correct image categorization	Downlink from orbiting satellite	Color pixel analysis	Display of scene categorization
Interactive English tutor	Student's score on test	Set of students, testing agency	Keyboard entry	Display of exercises, suggestions, corrections

Path finding Robot - Lab Example



PEAS Environment



Agent

Performance

Environment

Sensors

Actuators

Dimensions of Task Environment

Sensor Based:

- Observability : Full Vs Partial

Action Based:

- Dependency : Episodic Vs Sequential

State Based:

- No.ofState : Discrete Vs Continuous

Agent Based:

> Cardinality : Single Vs MultiAgent

Action & State Based:

- State Determinism : Deterministic Vs Stochastic | Strategic
- Change in Time : Static Vs Dynamic

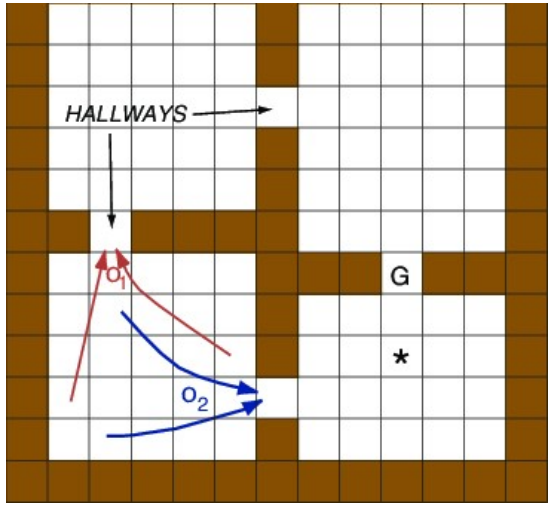
Task Environment

A rational agent is built to solve a specific task. Each such task would then have a different environment which we refer to as Task Environment

Based on the applicability of each technique for agent implementation its task environment design is determined by multiple dimension

Sensor Based:

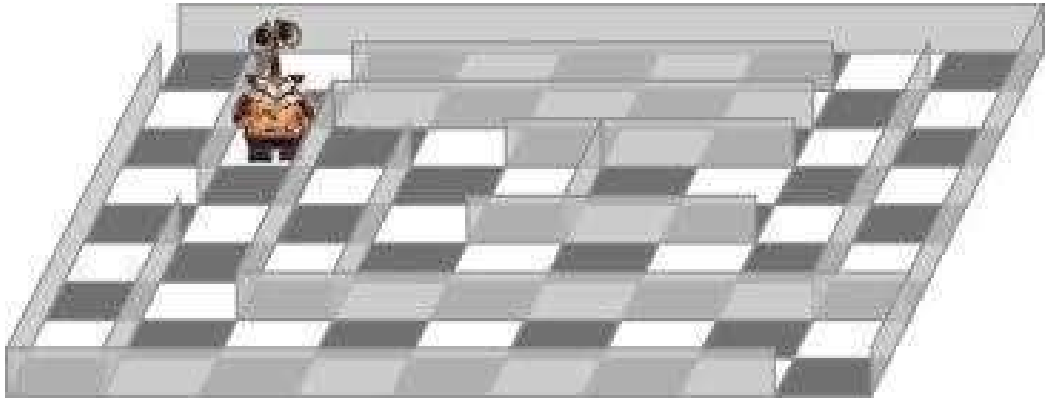
- Observability : Full Vs Partial

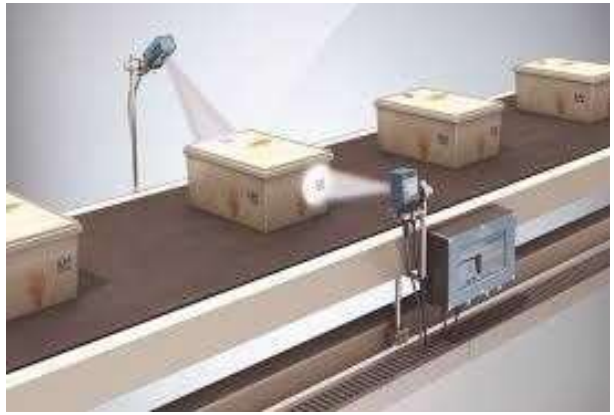


Task Environment

Action Based:

- Dependency : Episodic Vs Sequential

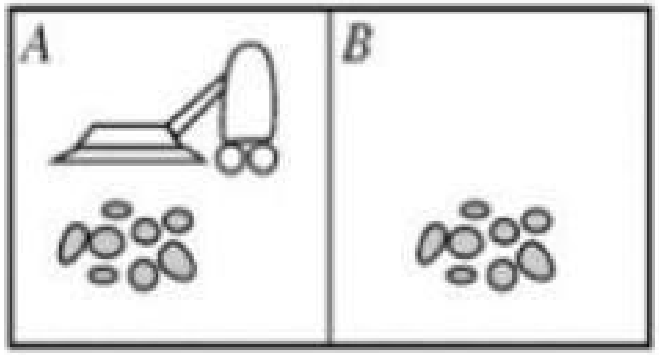


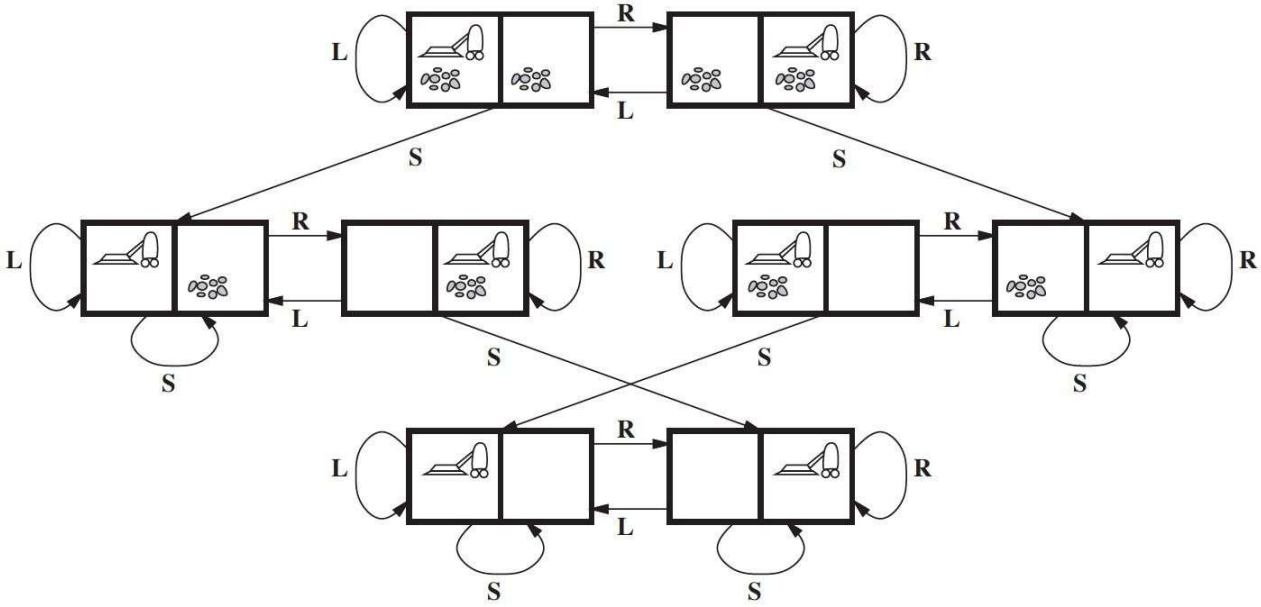


Task Environment

State Based:

- No.of.State : **Discrete** Vs Continuous





Task Environment

State Based:

- No.of.State : Discrete Vs Continuous

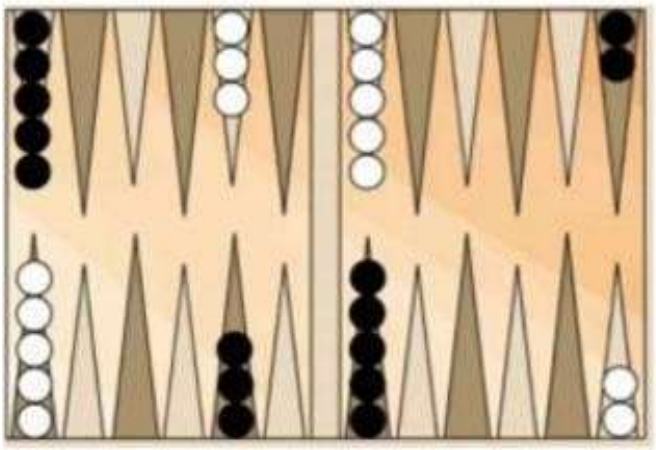
VS.

Task Environment

Action & State Based:

- State Determinism : Deterministic Vs Stochastic | Strategic

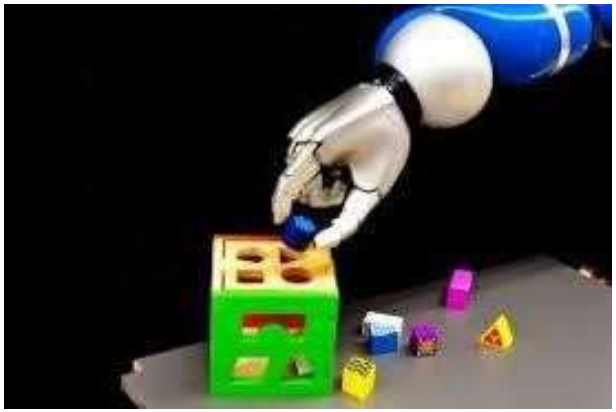
(If the environment is deterministic except for the actions of other agents, then the environment is strategic)



Task Environment

Agent Based:

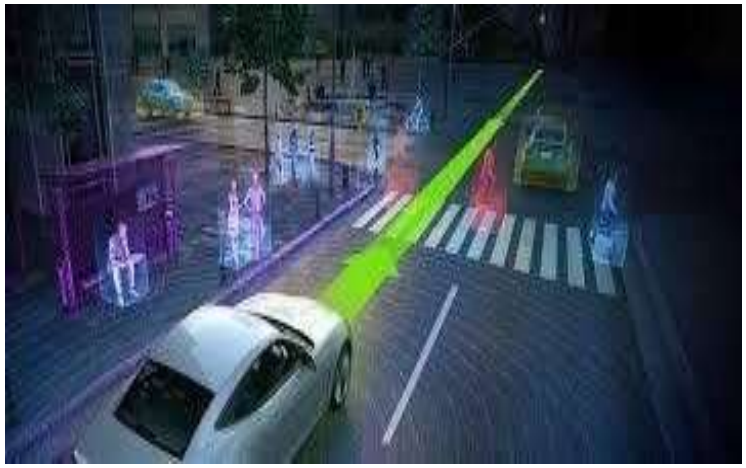
> Cardinality : Single Vs MultiAgent

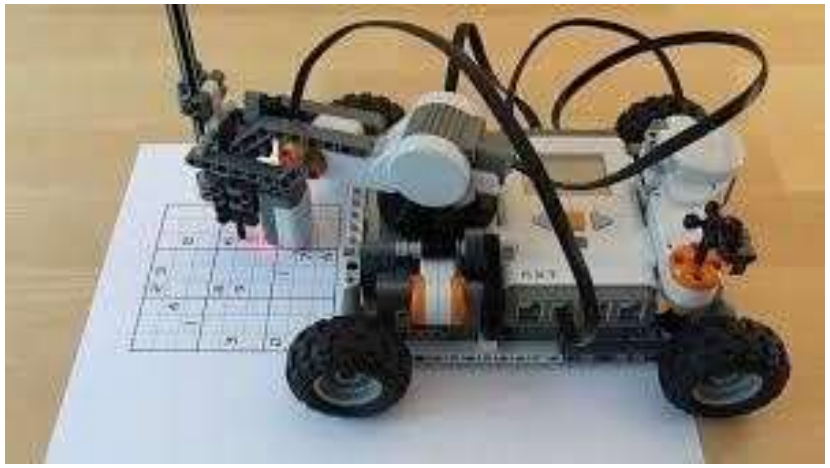


Task Environment

Action & State Based:

- Change in Time : Static Vs Dynamic
- (The environment is semi dynamic if the environment itself does not change with the passage of time but the agent's performance score does)

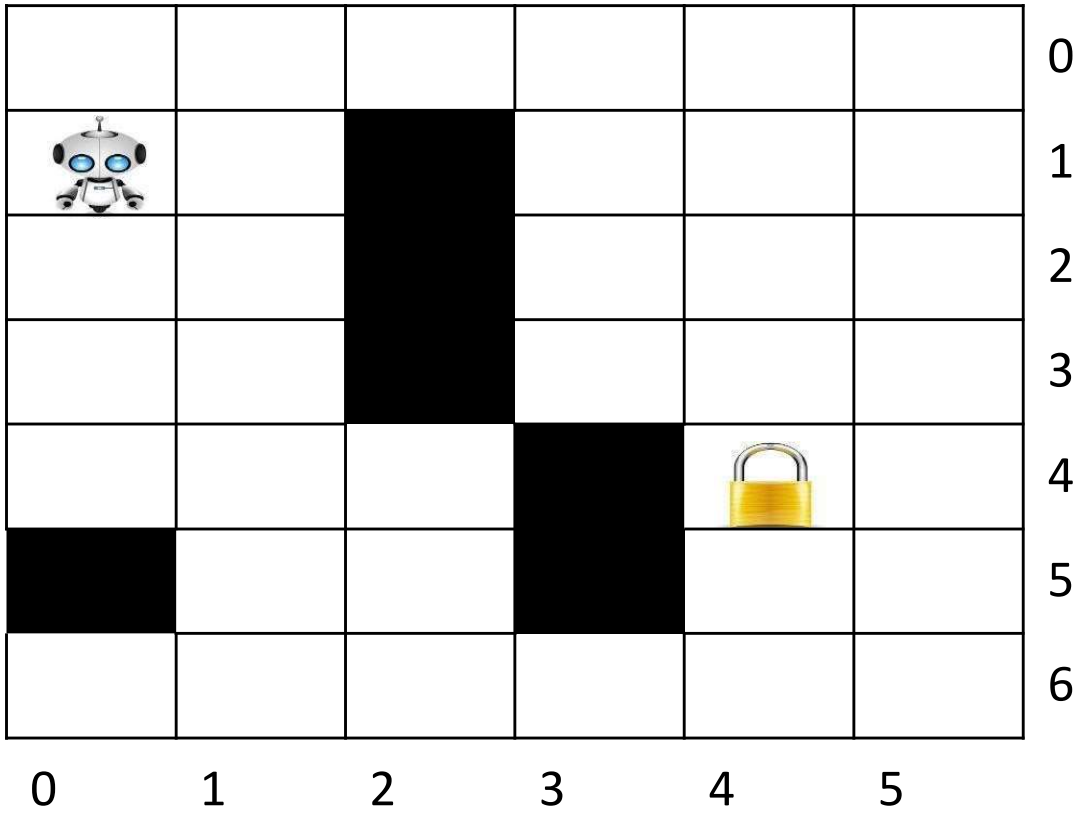




Task Environment

Task Environment	Fully vs Partially Observable	Single vs Multi-Agent	Deterministic vs Stochastic	Episodic vs Sequential	Static vs Dynamic	Discrete vs Continuous
Medical diagnosis system	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Satellite Image Analysis System	Fully	Single	Deterministic	Episodic	Static	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Path finding Robot - Lab Example



Agent
Observability
No.of.Agents
No.of.States
Determinism
Dynamicity
Output Dependency

Learning Objective Achieved

At the end of this class , students Should be able to:

1. Identify the requirement for AI solutions for given problem
2. Understand the significance of State based representations
3. Design the PEAS (Performance, Environment, Actuators, Sensors) for given problem
4. Identify dimensions of TASK environment

Next Class Plan

Structure of Agents-Architectures

Problem Solving Agents

Problem Formulation

Uninformed Search Algorithms

Required Reading: AIMA - Chapter #2

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CZG557

M1 : Introduction
&
M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1	Introduction to AI
M2	Problem Solving Agent using Search
M3	Game Playing
M4	Knowledge Representation using Logics
M5	Probabilistic Representation and Reasoning
M6	Reasoning over time
M7	Ethics in AI

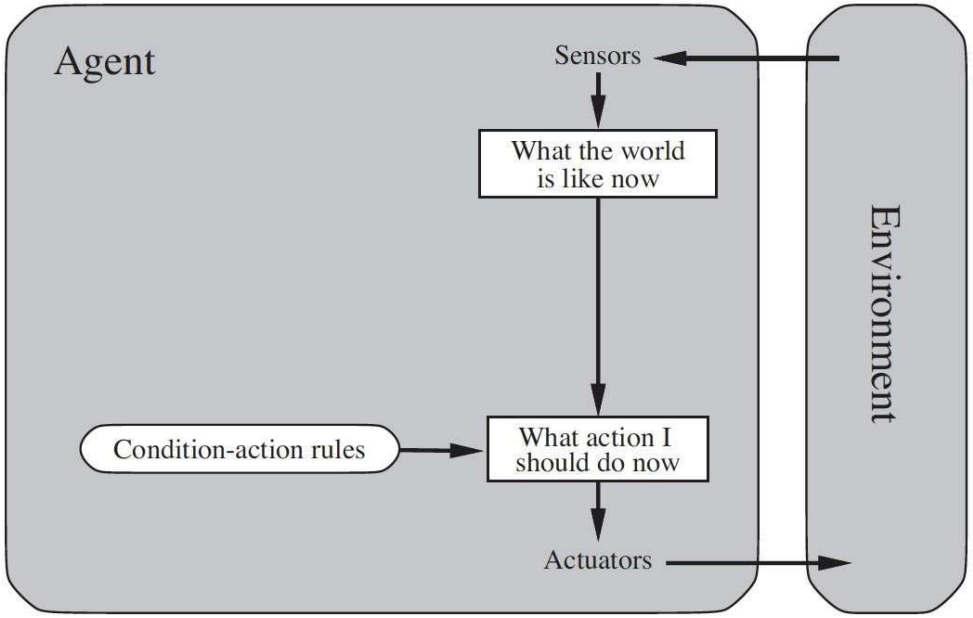
Agents Architectures

Agent Architectures

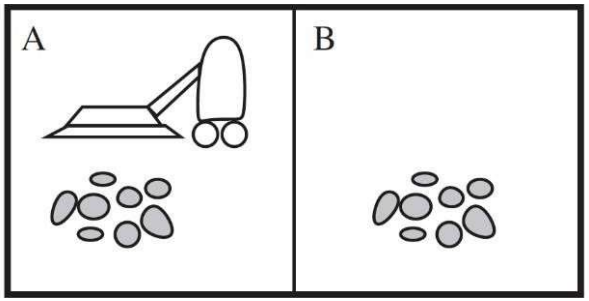
Reflex Agent

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
    persistent: rules, a set of condition–action rules
    state ← INTERPRET-INPUT(percept)
    rule ← RULE-MATCH(state, rules)
    action ← rule.ACTION
    return action
```

```
function REFLEX-VACUUM-AGENT( [location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```



Simple Reflex Agents

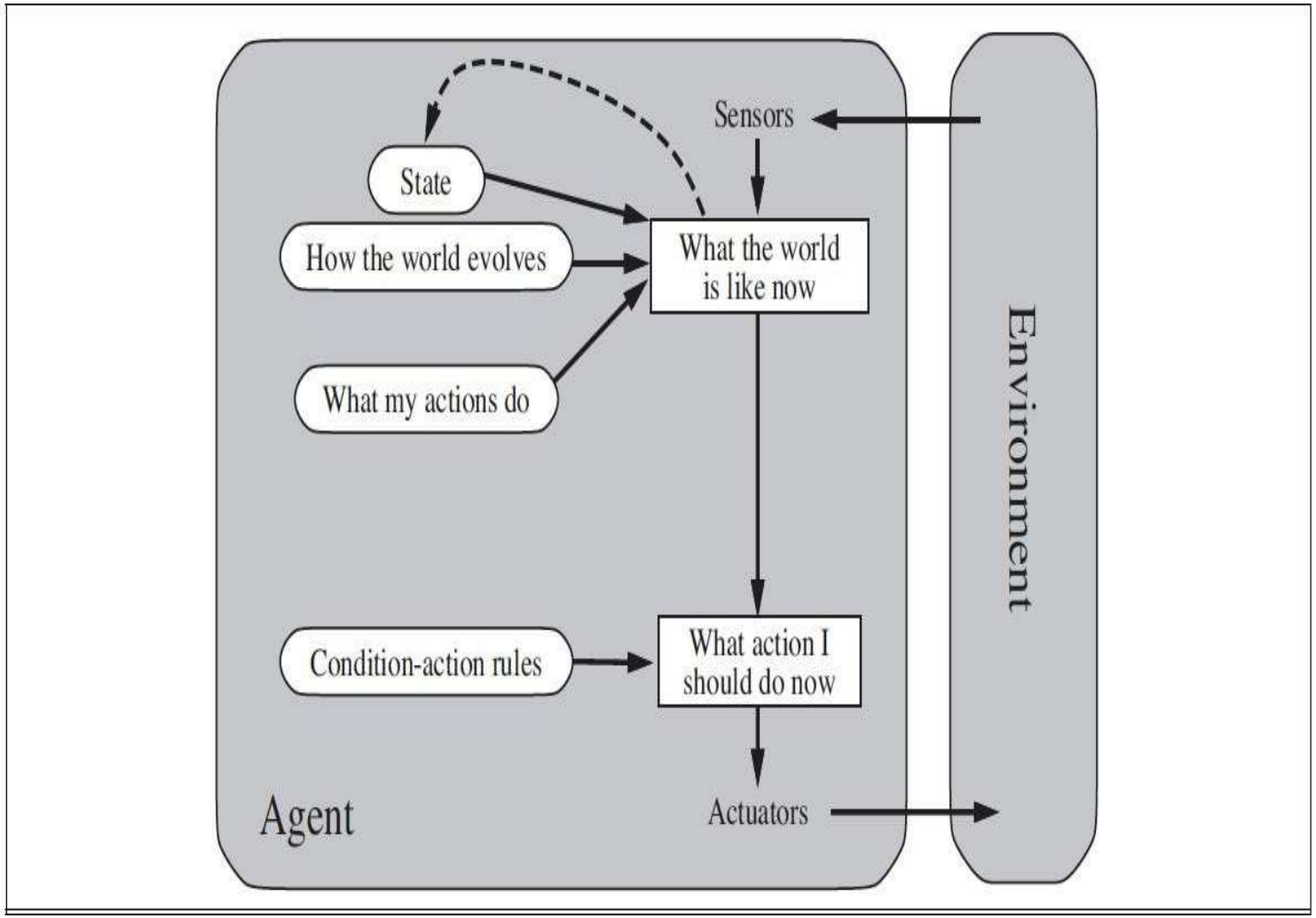


Agent Architectures

Model based Agent

Simple Reflex Agents

Model Based Agents



Model based Agent

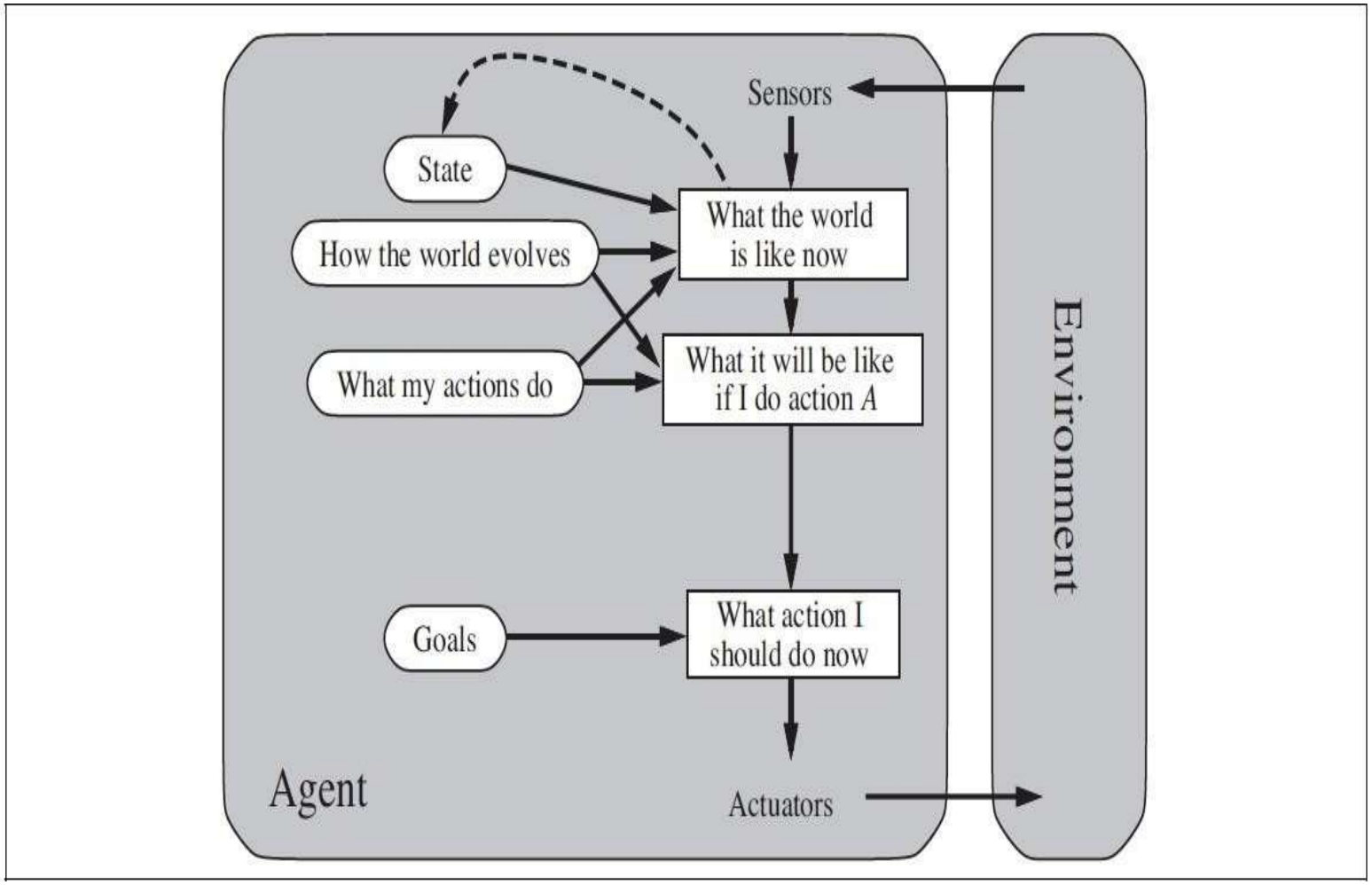
```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
persistent: state, the agent's current conception of the world state
            transition model, a description of how the next state depends on the current state and action
            sensor model , a description of how the current world state is reflected in the agent's percepts
            rules, a set of condition–action rules
            action, the most recent action, initially none
state ← UPDATE-STATE(state, action, percept, transition model, sensor model )
rule ← RULE-MATCH(state, rules)
action ← rule.ACTION
return action
```

Agent Architectures

Simple Reflex Agents

Model Based Agents

Goal Based Agents



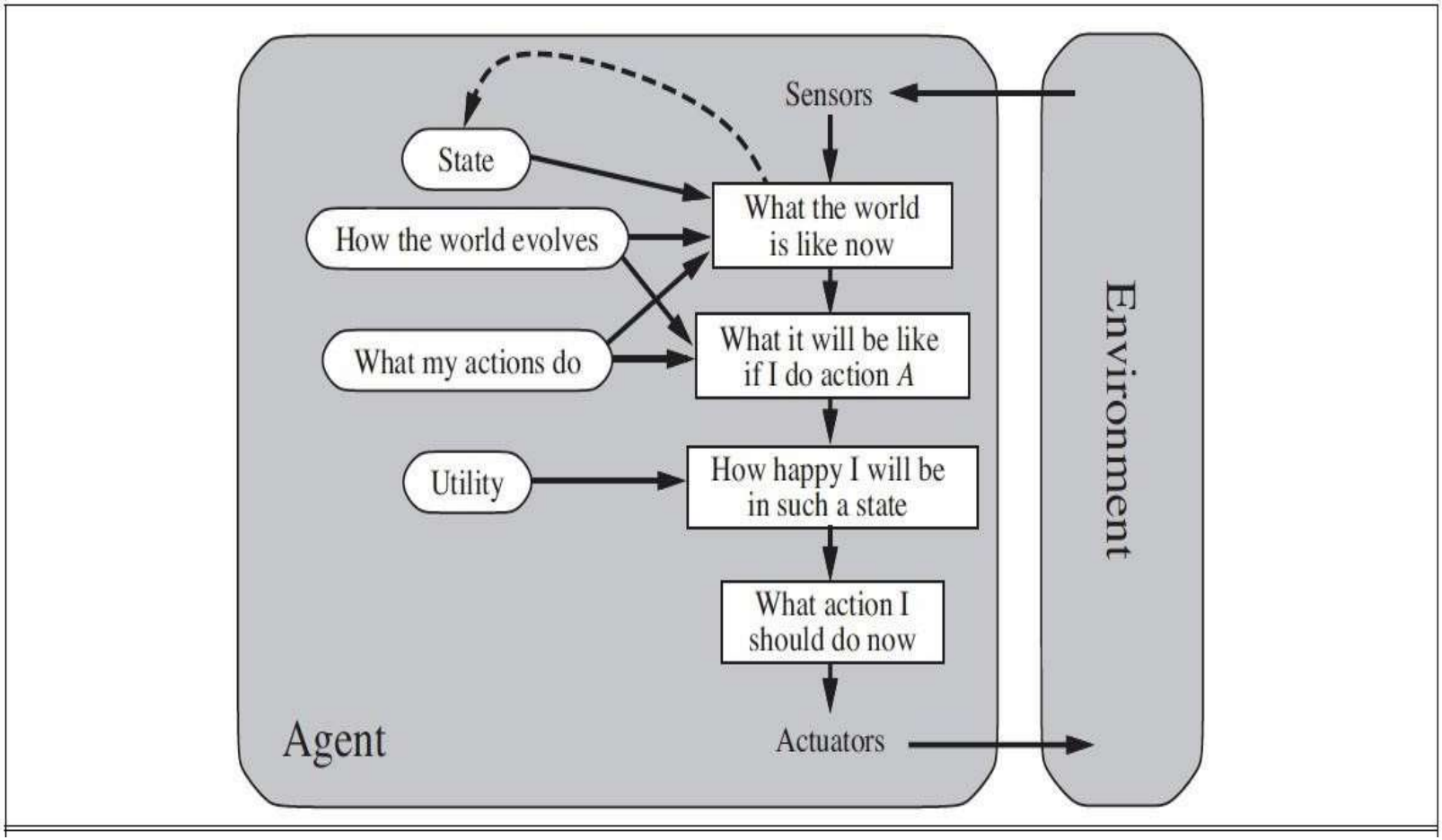
Agent Architectures

Simple Reflex Agents

Model Based Agents

Goal Based Agents

Utility Based Agents



Agent Architectures

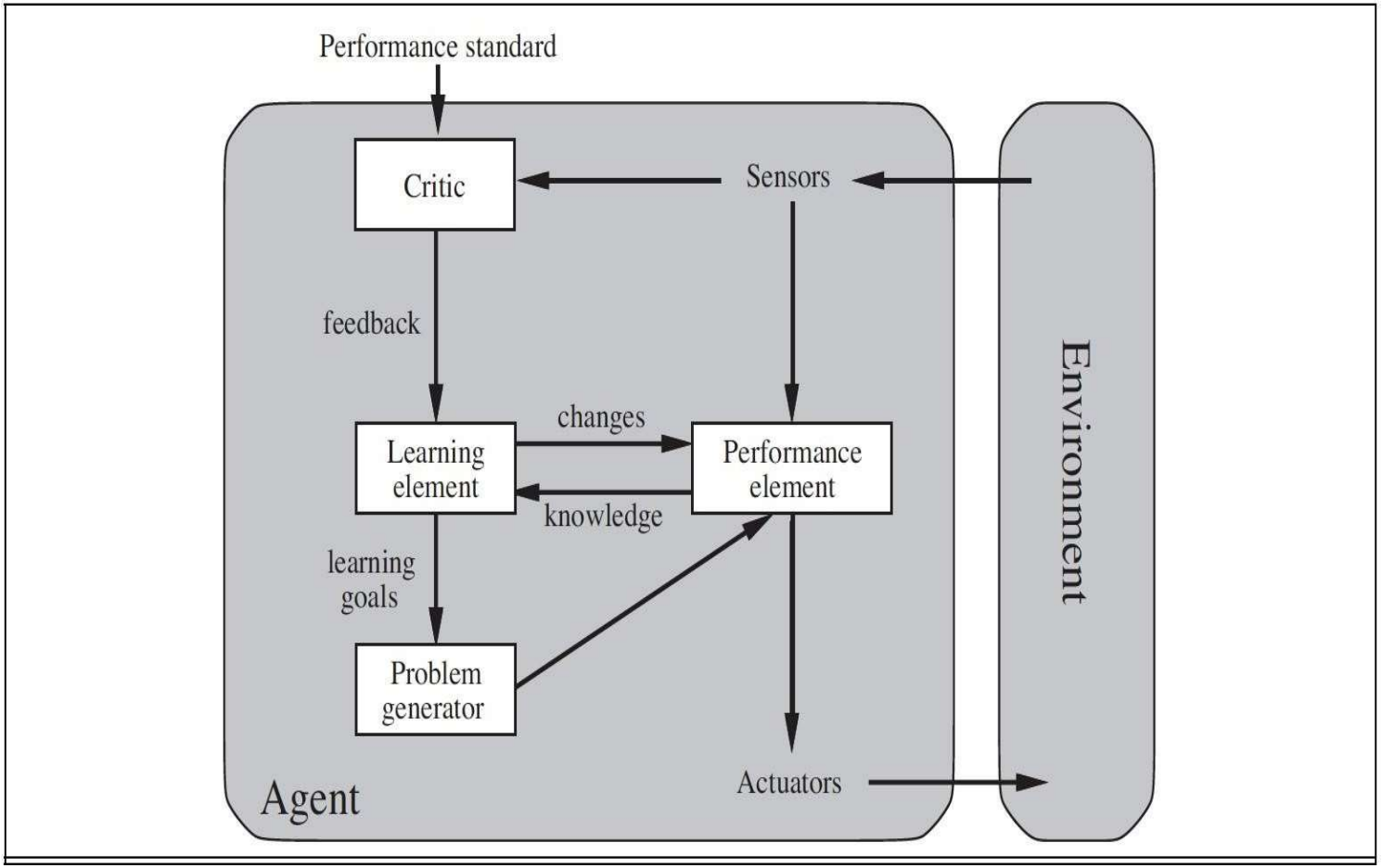
Simple Reflex Agents

Model Based Agents

Goal Based Agents

Utility Based Agents

Learning Agents



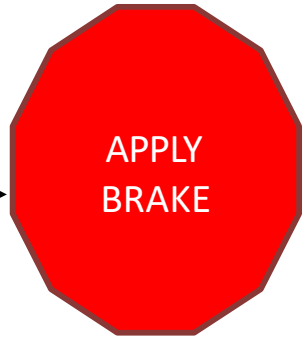
Role of Learning

Agents that improve their performance by learning from their own experiences

Input Percept

Performance Element

Output Action



Resulting State

Critic

Feedback

-2

Role of Learning

Input Percept

Possible Actions

Brake
Change Gear to Lower
Change Gear to Higher
Accelerate
Steer left
Steer right

Random

Selected Action

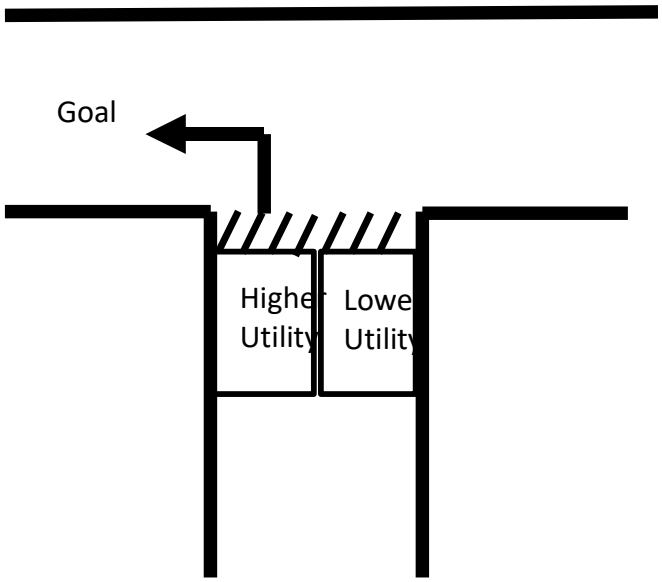
Change Gear to Lower

Role of Learning

<u>Performance Element</u> – Takes decision on action based on percept

$$f(red\ signal,\quad distance) = 15k\ N\ brake$$
$$distance = f^{'}(percept\ sequence)$$
$$f(percepts, distance, raining)$$

- $f(state_0, actionA) = 0.83,$
- $f(state_0, actionB) = 0.45$

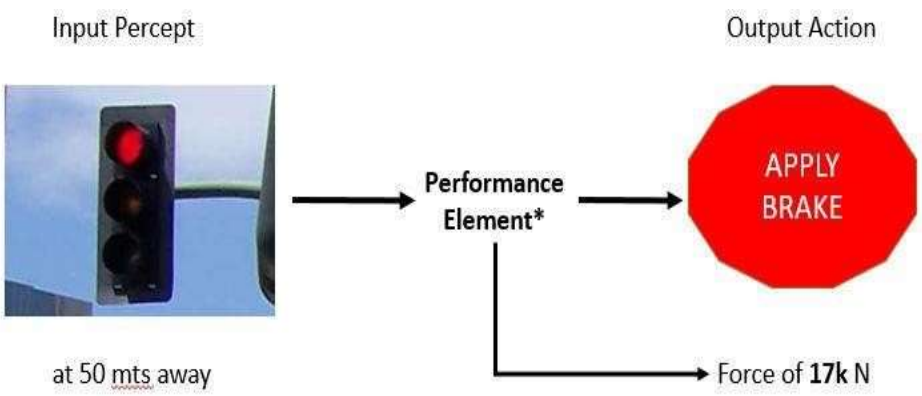


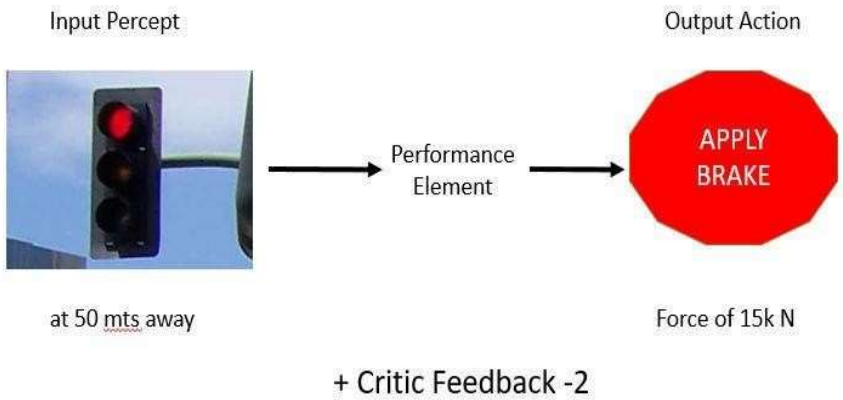
Role of Learning

Critic – Provides feedback on the actions taken

Learning :

Supervised Vs Unsupervised Vs Reinforcement

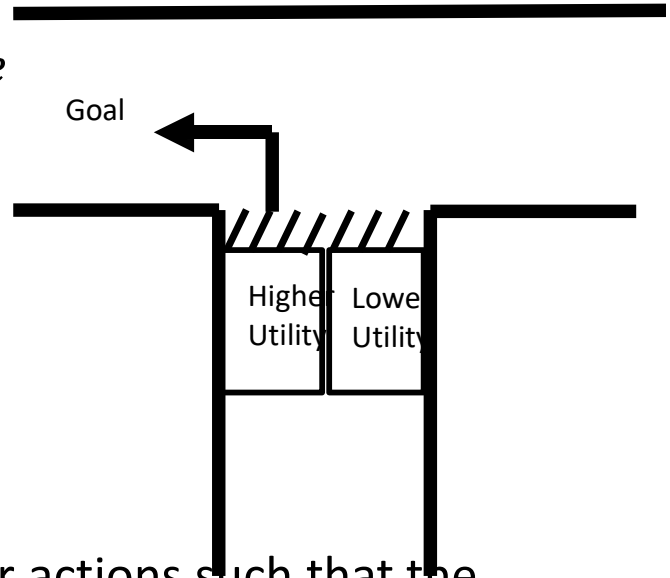




Role of Learning

Performance Element – Takes decision on action based on percept

$$f(red\ signal,\quad distance) = 15k\ N\ brake$$
$$distance = f'(percept\ sequence)$$
$$f(percepts, distance, raining)$$



- $f(state_0, actionA) = 0.83,$
- $f(state_0, actionB) = 0.45$

Learning Element – Make the performance element select better actions such that the utility function is optimized

Critic – Provides feedback on the actions taken

Problem Generator – Make the Performance Element select sub-optimal actions such that you would learn from unseen actions

Next Class Plan

- Problem Solving Agents
- Uninformed Search Algorithms
  - BFS vs DFS – An overview
  - Uniform Cost Search
  - Iterative Depth First Search
  - Notion of Bi-Directional Search

Required Reading: AIMA - Chapter #1, 2

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CZG557

M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Learning Objective

At the end of this class , students Should be able to:

1. Design problem solving agents
2. Create search tree for given problem
3. Apply uninformed search algorithms to the given problem
4. Compare performance of given algorithms in terms of completeness, optimality, time and space complexity
5. Differentiate for which scenario appropriate uninformed search technique is suitable and justify

Problem Formulation

Problem Solving Agents

Goal based decision making agents finds sequence of actions that leads to the desirable state.

Phases of Solution Search by PSA

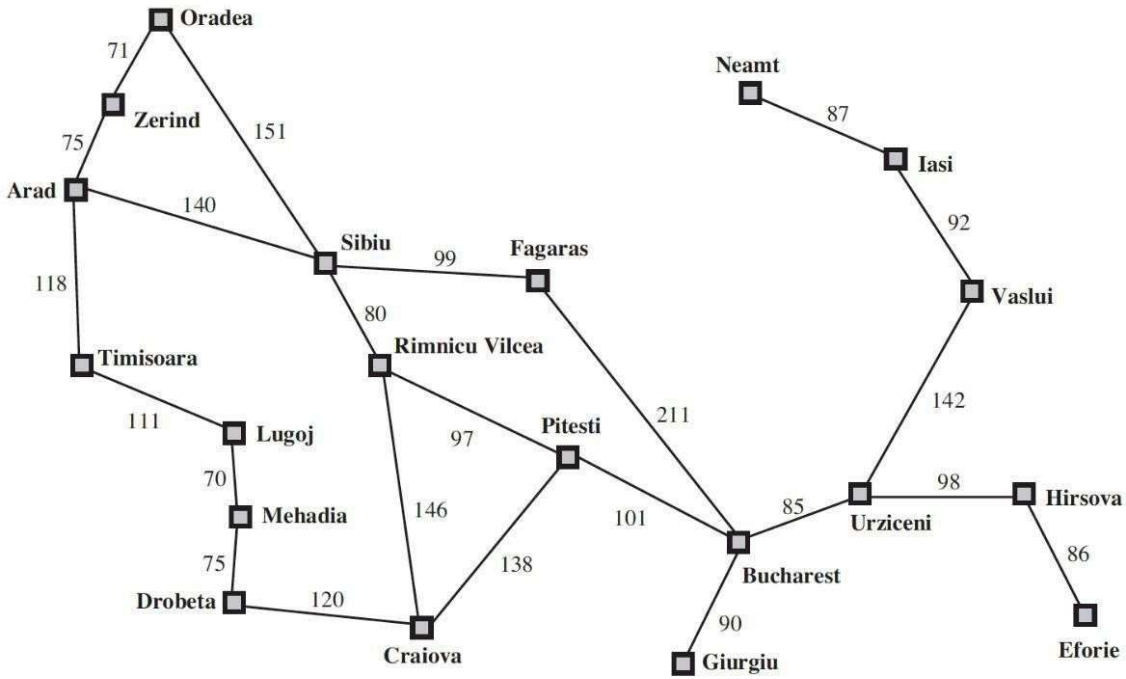
Goal Formulation

Problem Formulation

Search Phase

Execution Phase

Optimizes the Objective (Local | Global) Limits the Actions



Problem Solving Agents

Phases of Solution Search by PSA

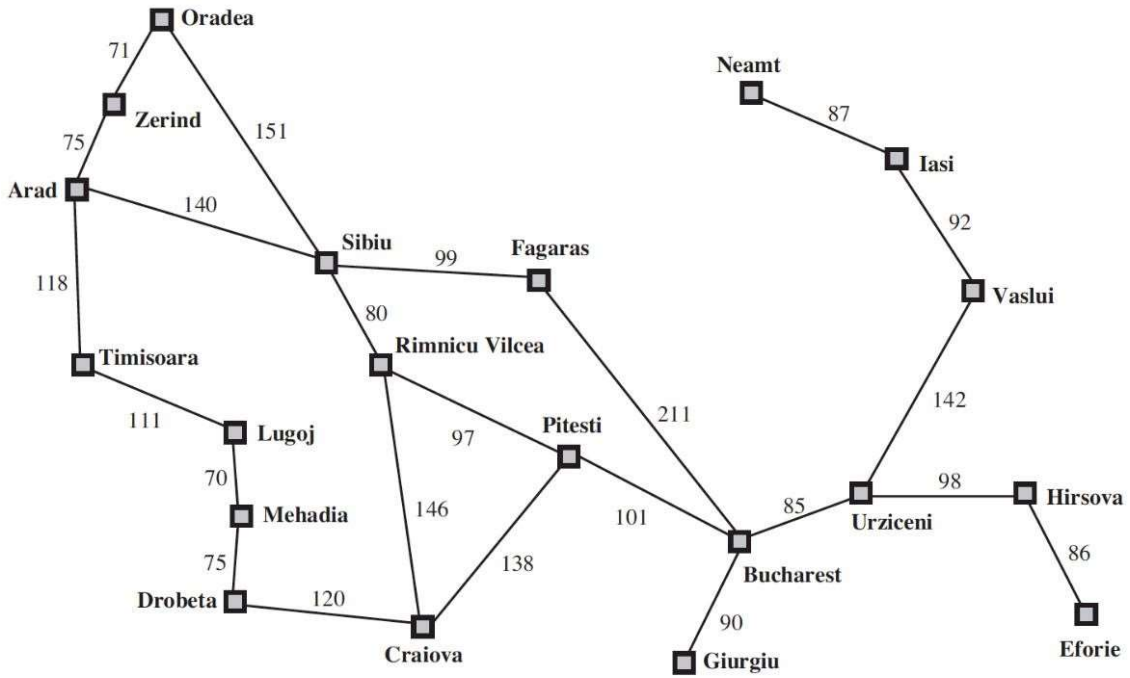
Goal Formulation

Problem Formulation

Search Phase

Execution Phase

State Space Creations [in the path of Goal]
Lists the Actions



Problem Solving Agents

Goal Formulation

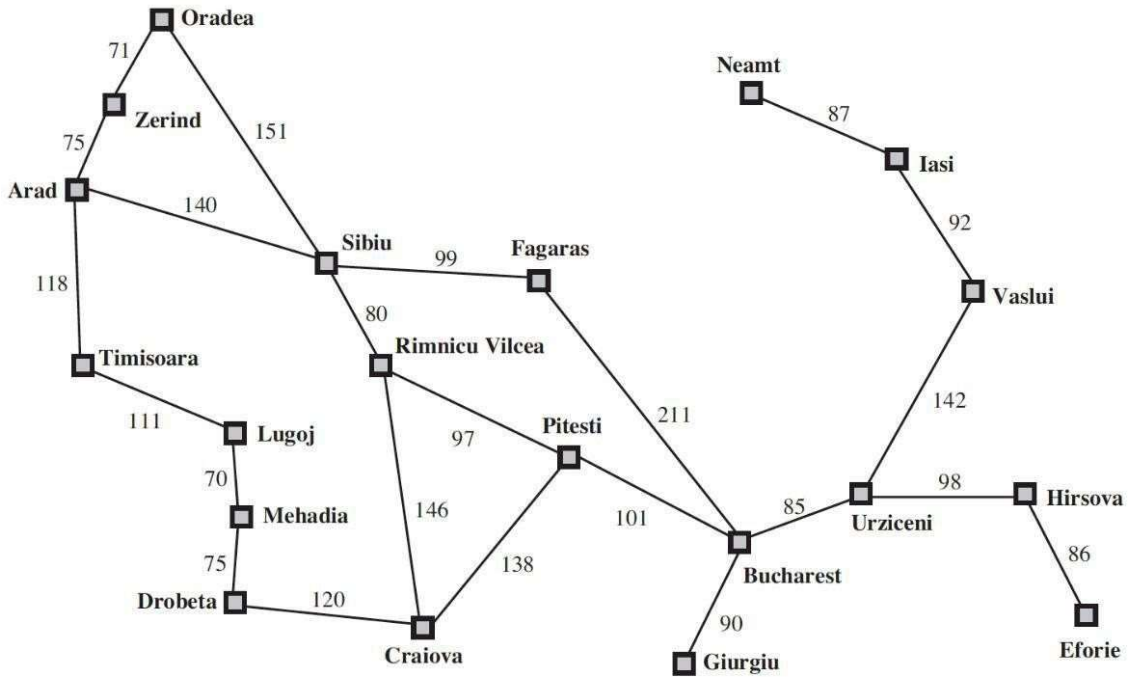
Problem Formulation

Search Phase

Execution Phase

Phases of Solution Search by PSA

Assumptions – Environment :
Static
Observable Discrete
Deterministic



Problem Solving Agents

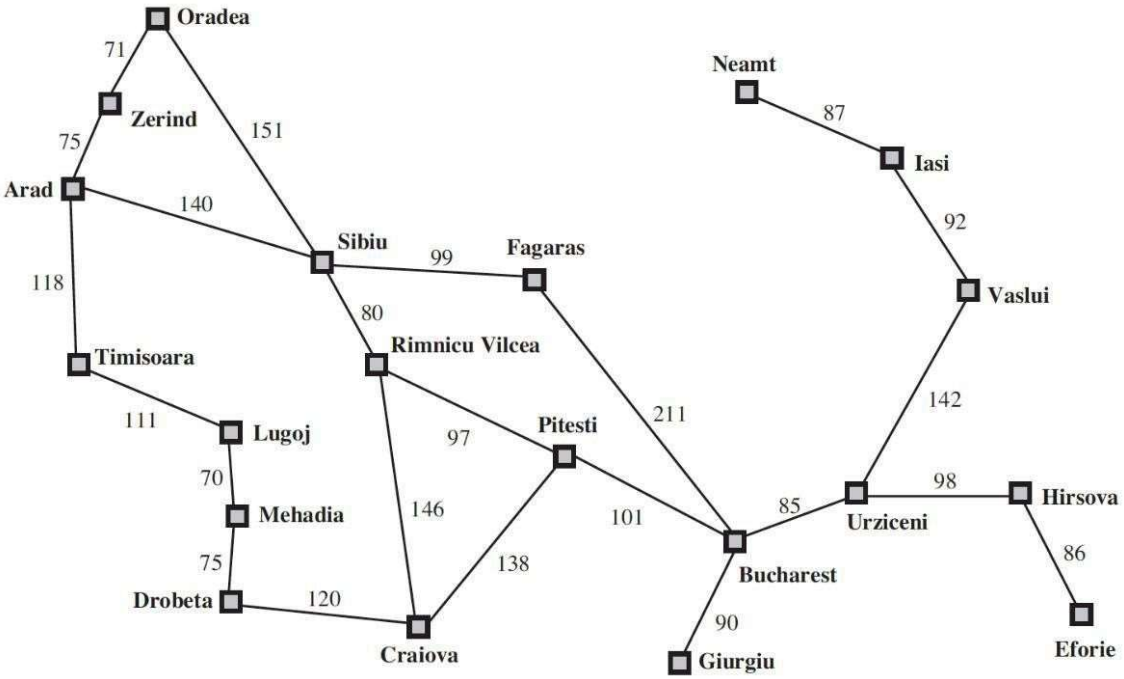
Phases of Solution Search

Goal Formulation

Problem Formulation

Search Phase

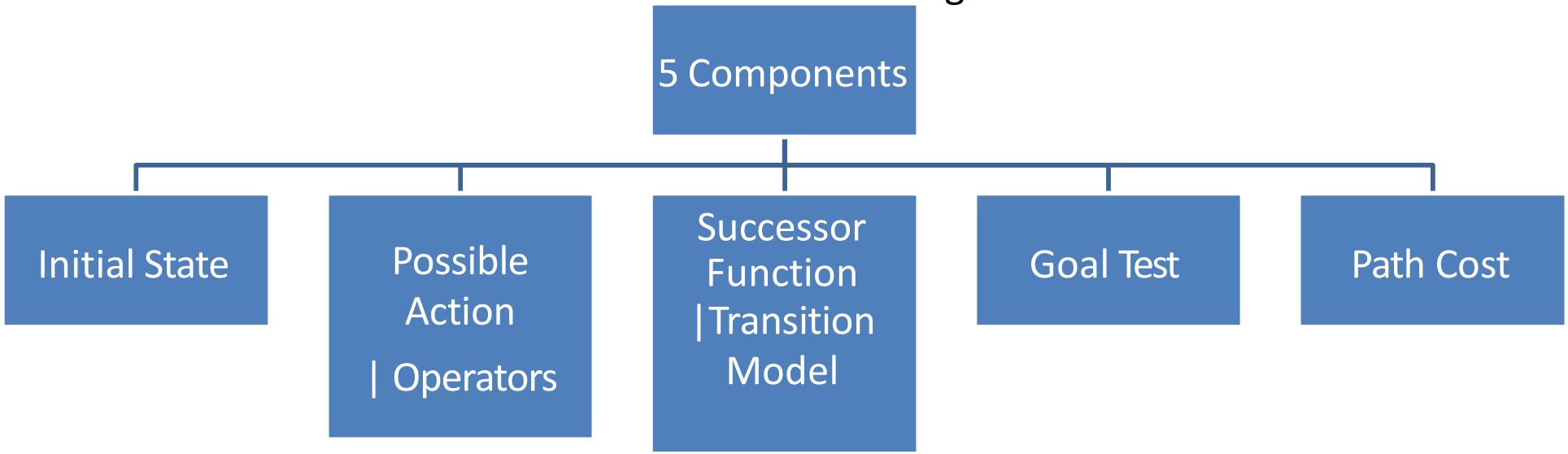
Execution Phase



Examine all sequence
Choose best |
Optimal

Problem Solving Agents – Problem Formulation

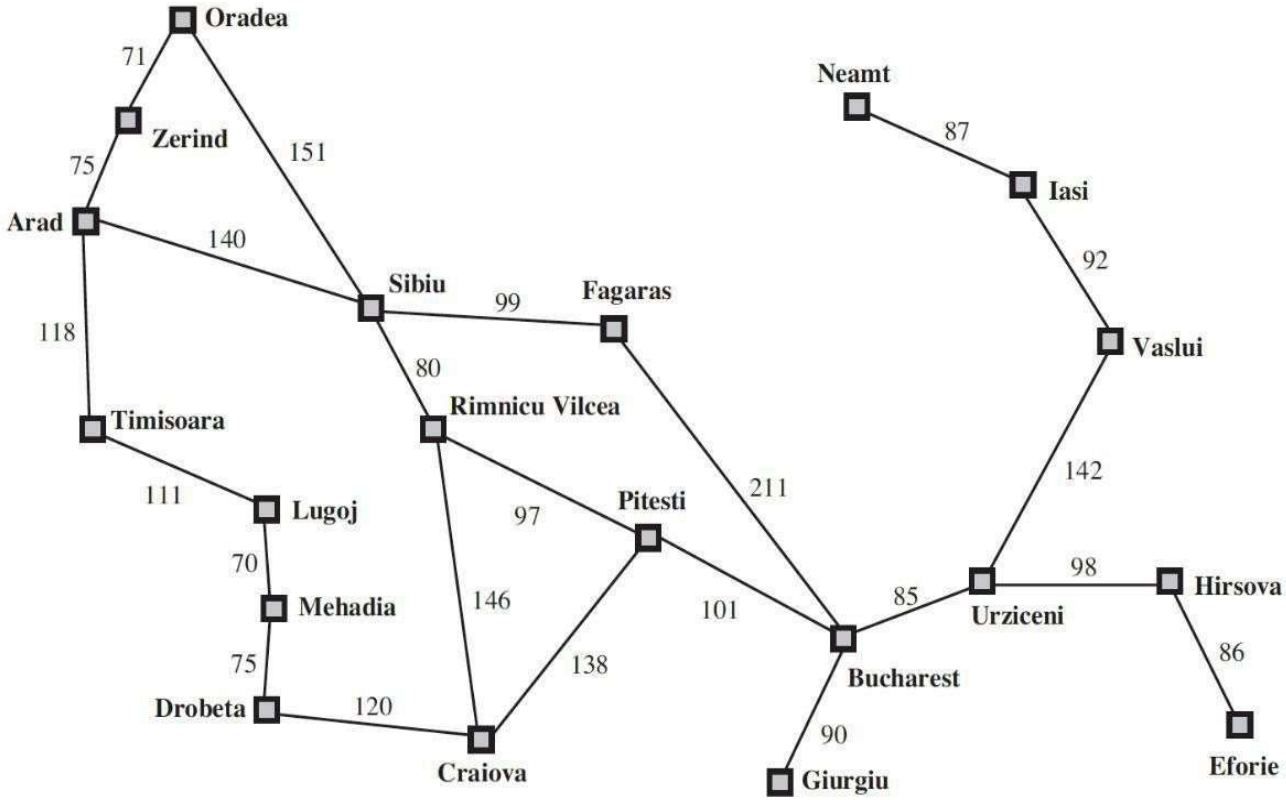
Abstraction Representation
Decide what actions under states to take to achieve a goal



A function that assigns a numeric cost to each path. A path is a series of actions. Each action is given a cost depending on the problem.

Solution = Path Cost Function + Optimal Solution

Problem Solving Agents – Problem Formulation: BookExample



Initial State –E.g., *In(Arad)*

Possible Actions – *ACTIONS(s)* □ *{Go(Sibiu), Go(Timisoara), Go(Zerind)}*

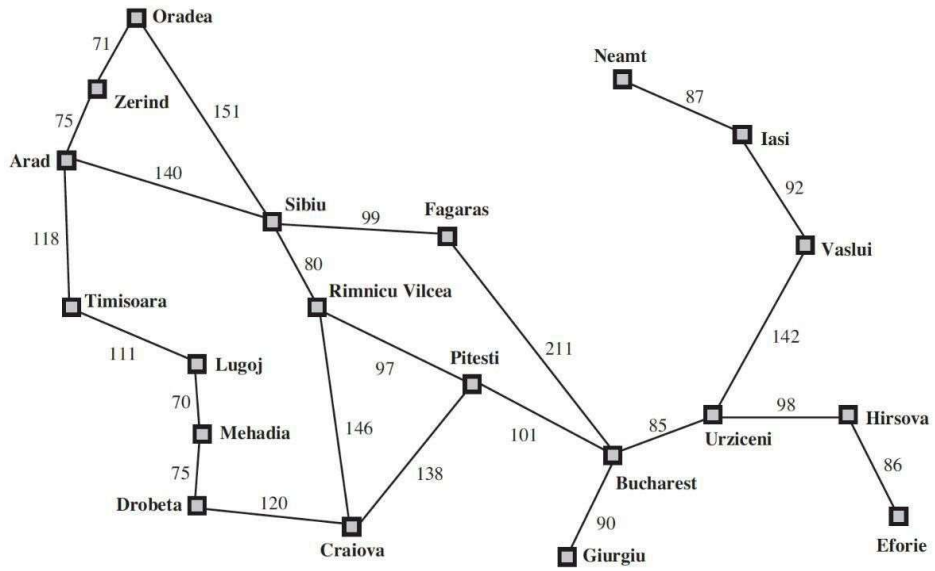
Transition Model – *RESULT(In(Arad), Go(Sibiu)) = In(Sibiu)*

Goal Test – *IsGoal(In(Bucharest)) = Yes*

Path Cost – *cost(In(Arad), go(Sibiu)) = 140 kms*

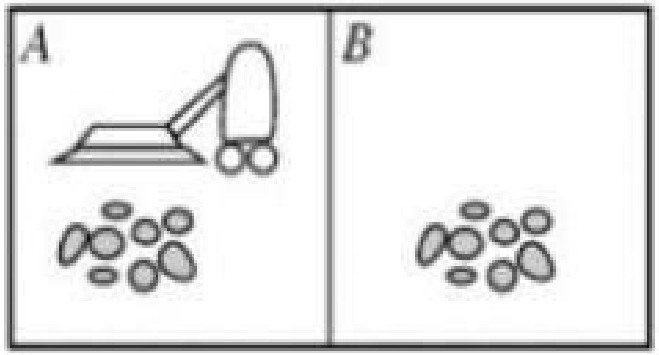
Example Problem Formulation

	Travelling Problem
Initial State	Based on the problem
Possible Actions	Take a flight | Train | Shop
Transition Model/ Successor Function	[A, Go(A->S)] = [S]
Goal Test	Is current = B (destination)
Path Cost	Cost + Time + Quality



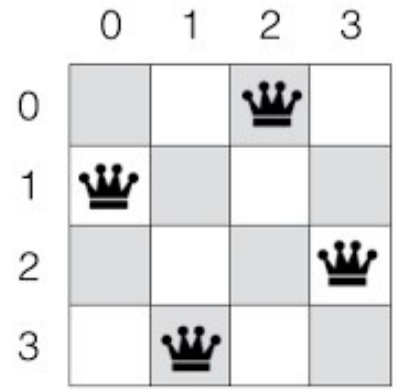
Example Problem Formulation

	Vacuum World
Initial State	Any
Possible Actions	[Move Left, Move Right, Suck, NoOps]
Transition Model/ Successor Function	[A, ML] = [B , Dirty] [A, ML] = [B, Clean]
Goal Test	Is all room clean? [A, Clean] [B, Clean]
Path Cost	No of steps in path



Example Problem Formulation

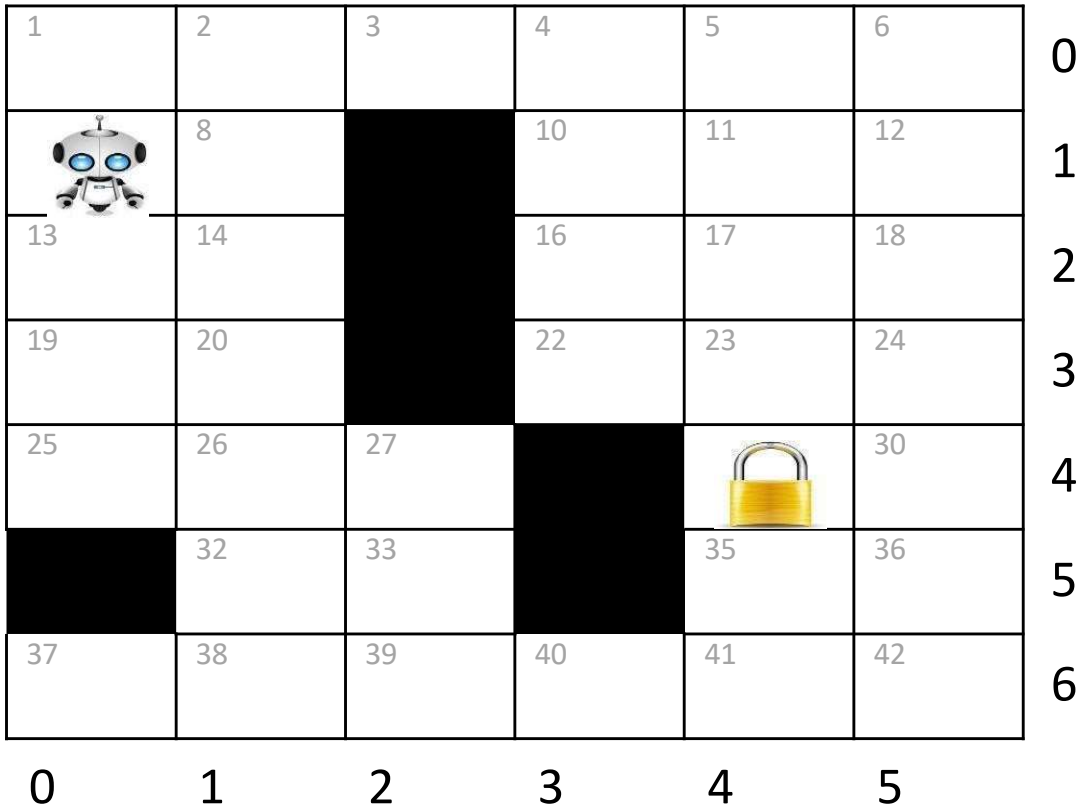
	N-Queen
Initial State	Empty | Partial | Full
Possible Actions	
Transition Model/ Successor Function	
Goal Test	
Path Cost	



board[r][c]

Path finding Robot

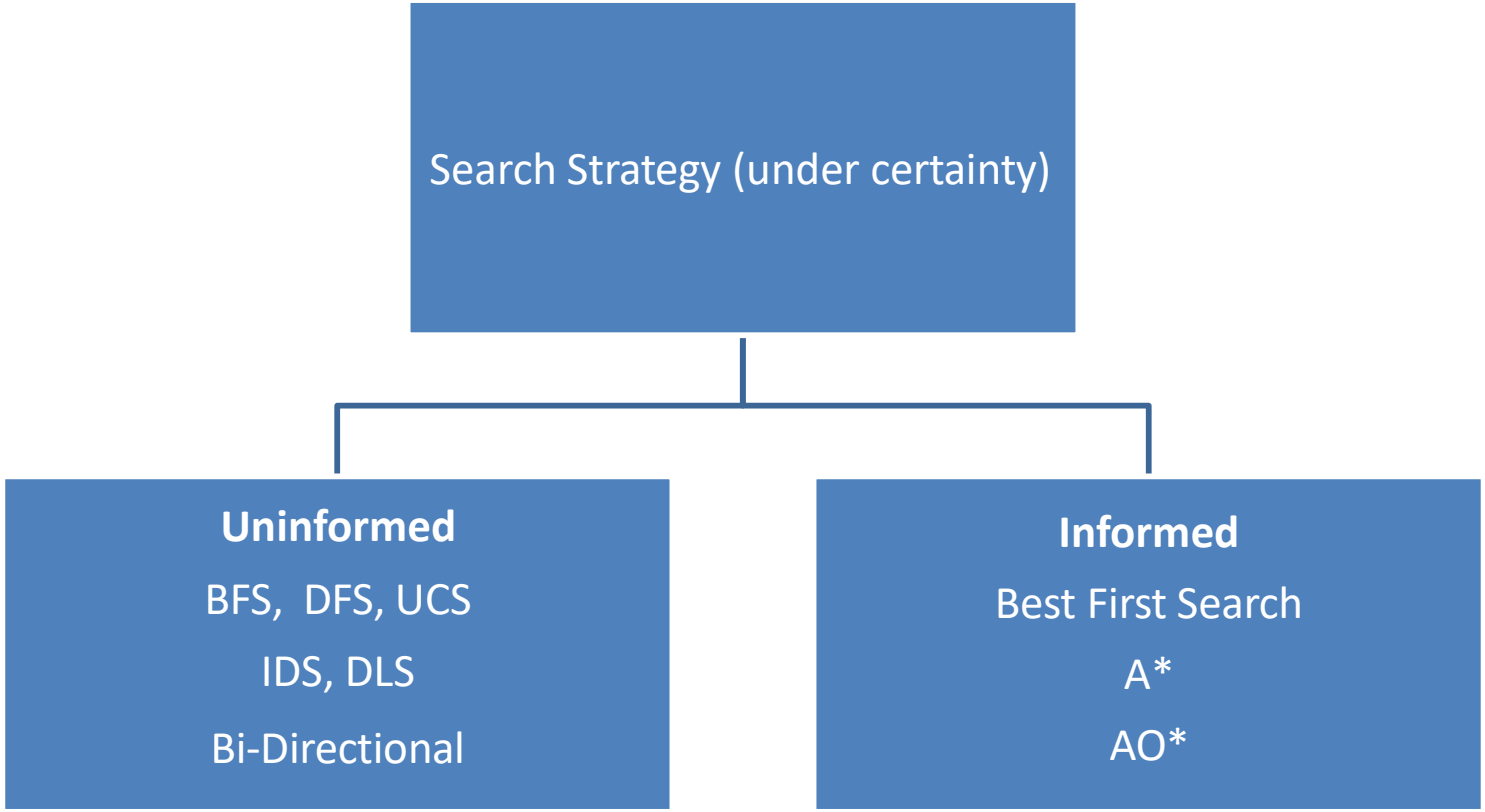
Successor Function Design

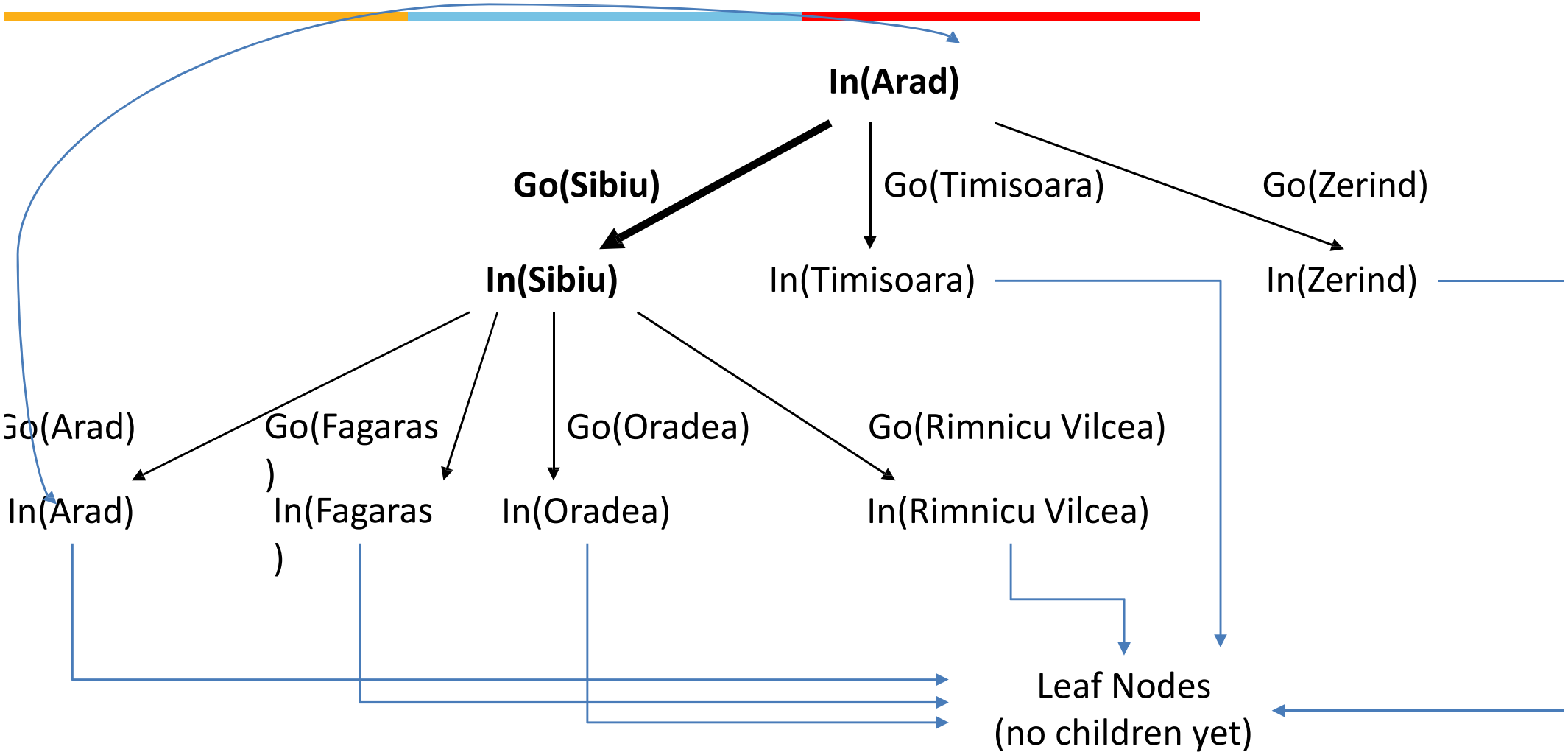


N-W-E-S

Searching for Solutions

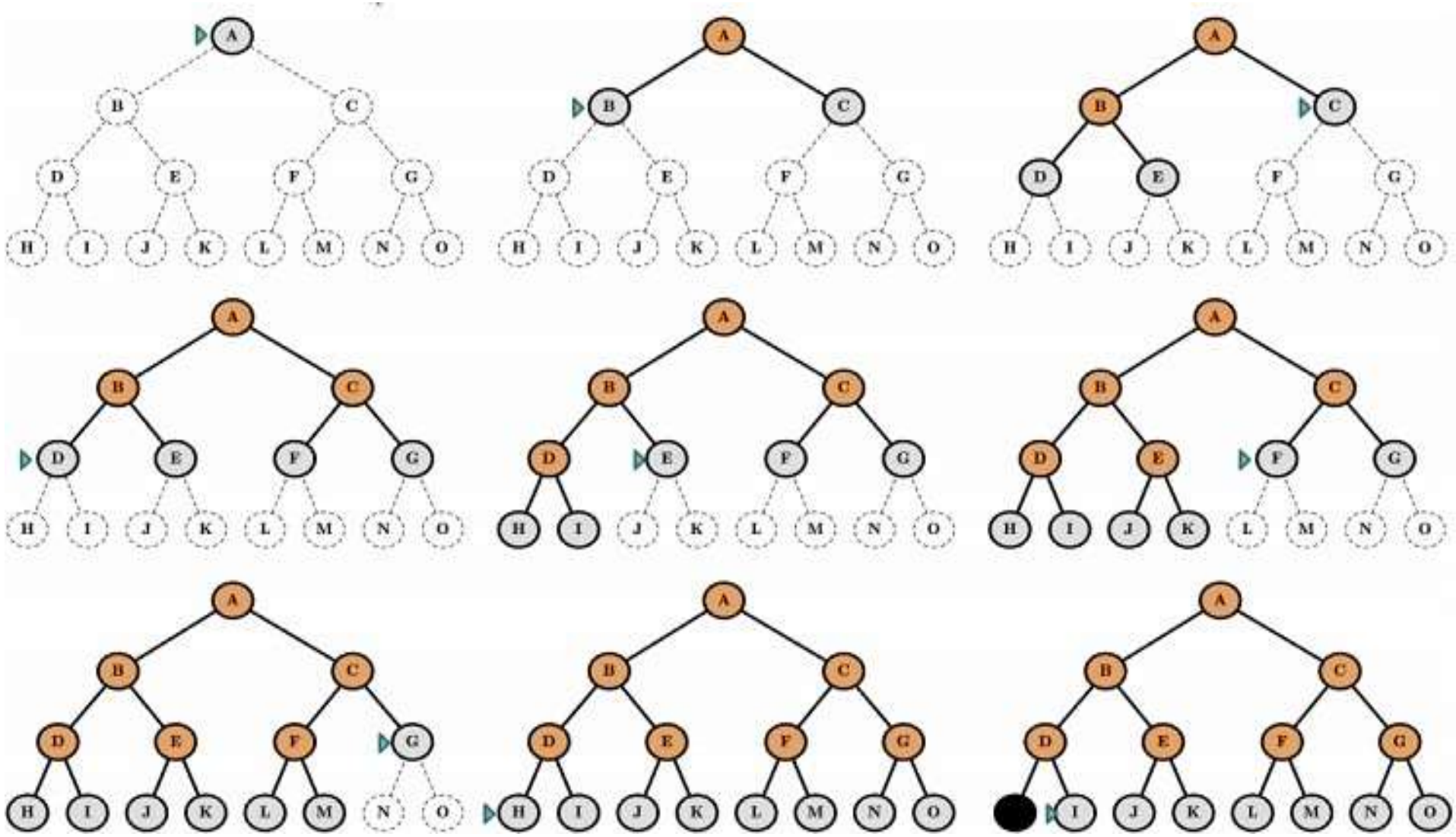
Choosing the current state, testing possible successor function, expanding current state to generate new state is called Traversal. Choice of which state to expand – Search Strategy



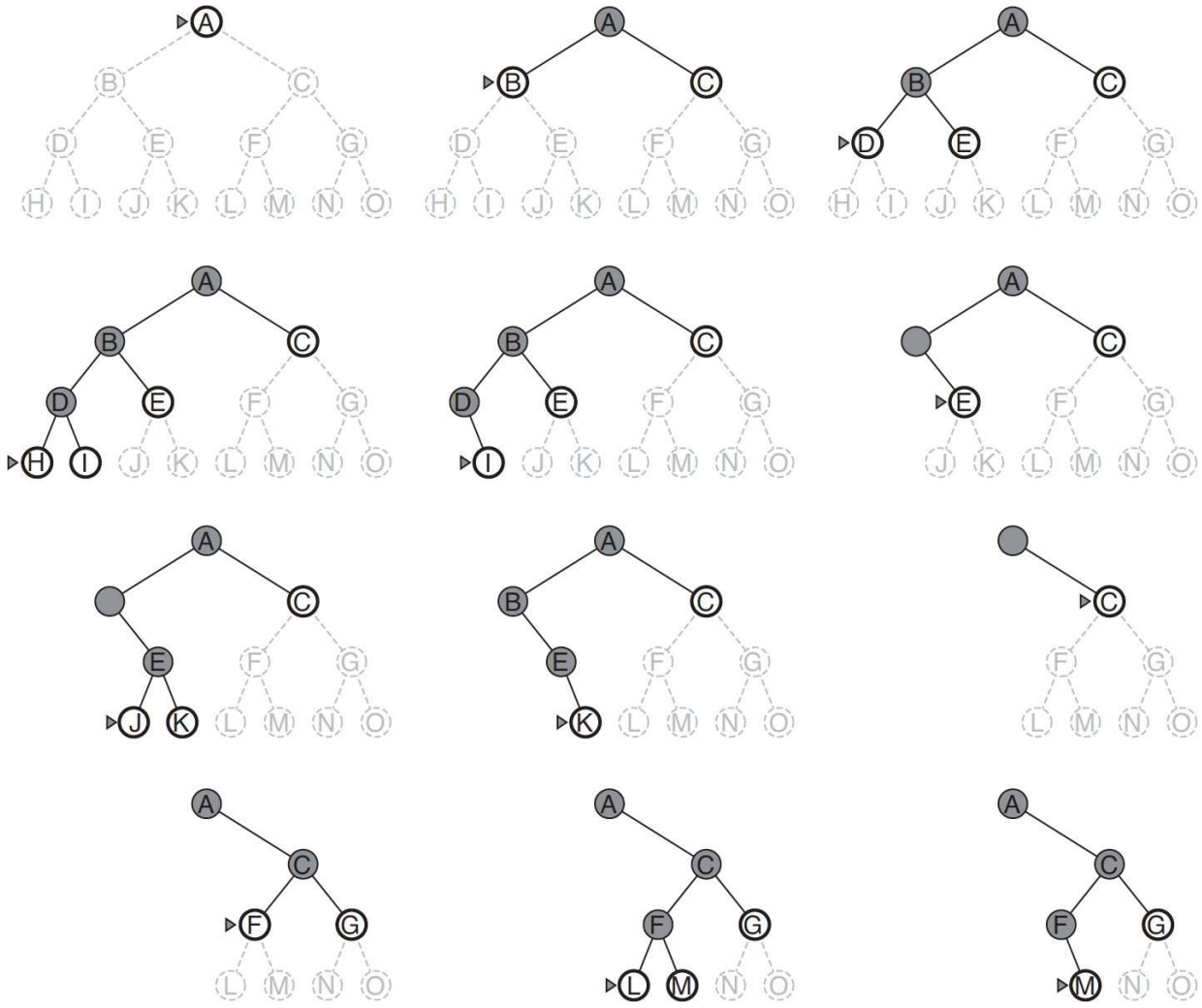

In(Arad)
Go(Sibiu)
Go(Timisoara)
Go(Zerind)
In(Sibiu)
In(Timisoara)
In(Zerind)
Go(Arad)
Go(Fagaras)
Go(Oradea)
Go(Rimnicu Vilcea)
In(Arad)
In(Fagaras)
In(Oradea)
In(Rimnicu Vilcea)
Leaf Nodes
(no children yet)

Uninformed Search Overview

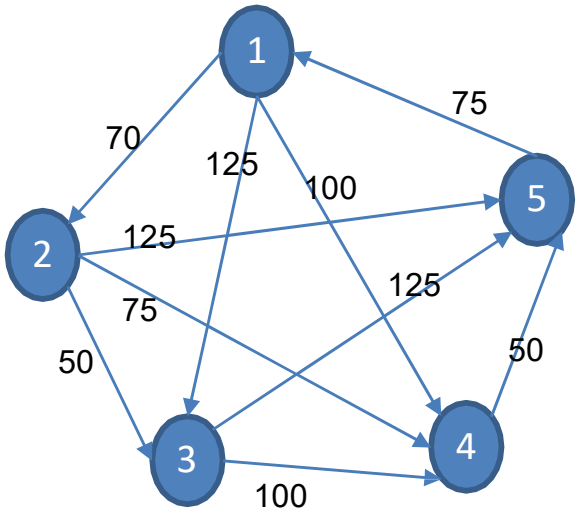
Breadth First Search (BFS)

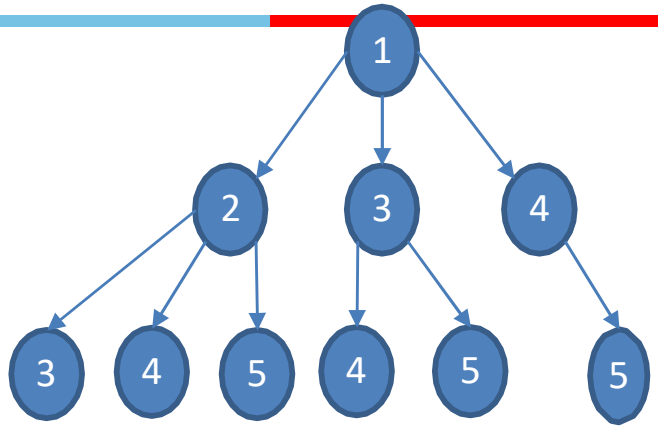


Depth First Search (DFS)



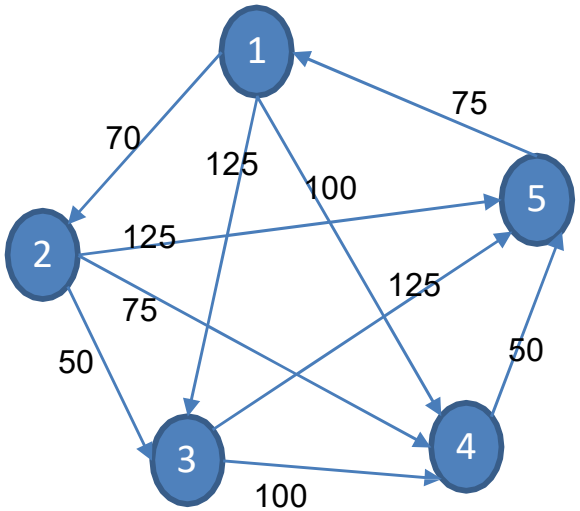
Search Tree – Sample Generation

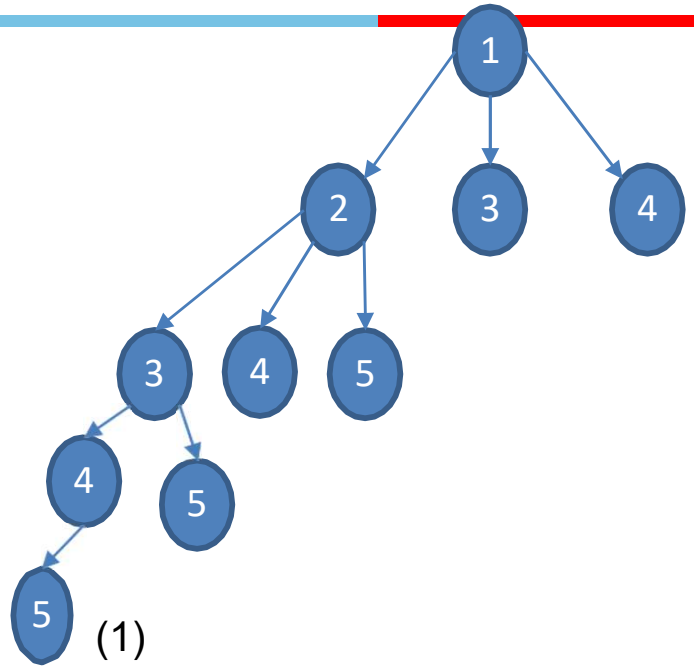




Each NODE in in the search tree denotes an entire PATH through the state space graph.

Search Algorithm – Uninformed Example - 1

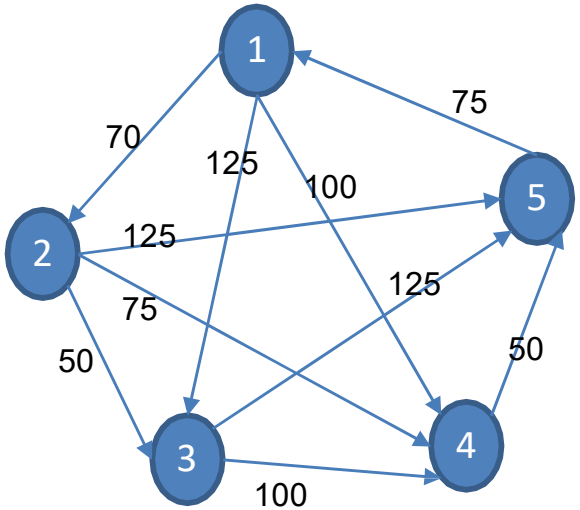


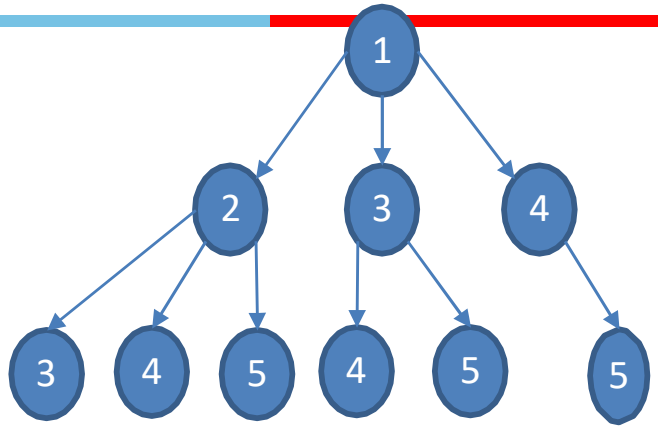


(1)
(1 2) (1 3) (1 4)
(1 2 3) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4 5) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)

C(1-2-3-4-5) = 70 + 50 + 100 + 50 =270
Expanded : 4
Generated : 10
Max Queue Length : 6

Search Algorithm – Uninformed Example - 2



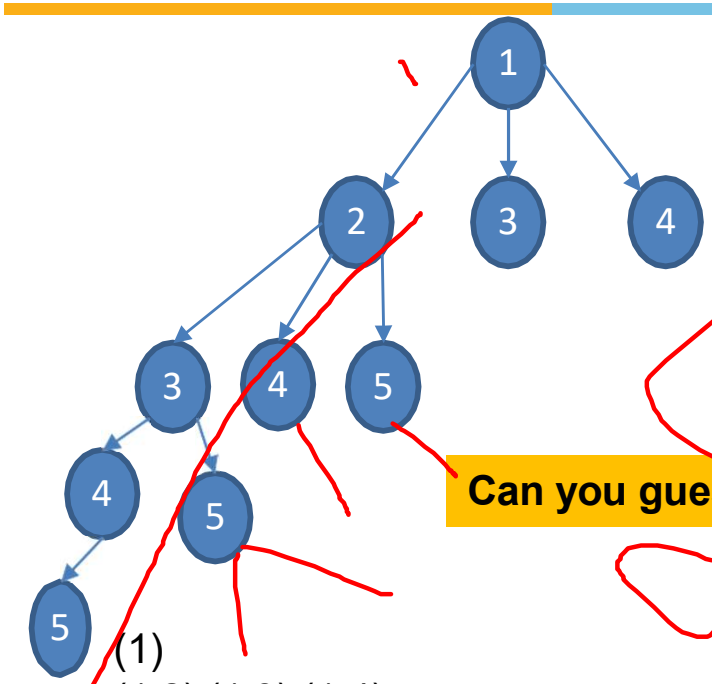


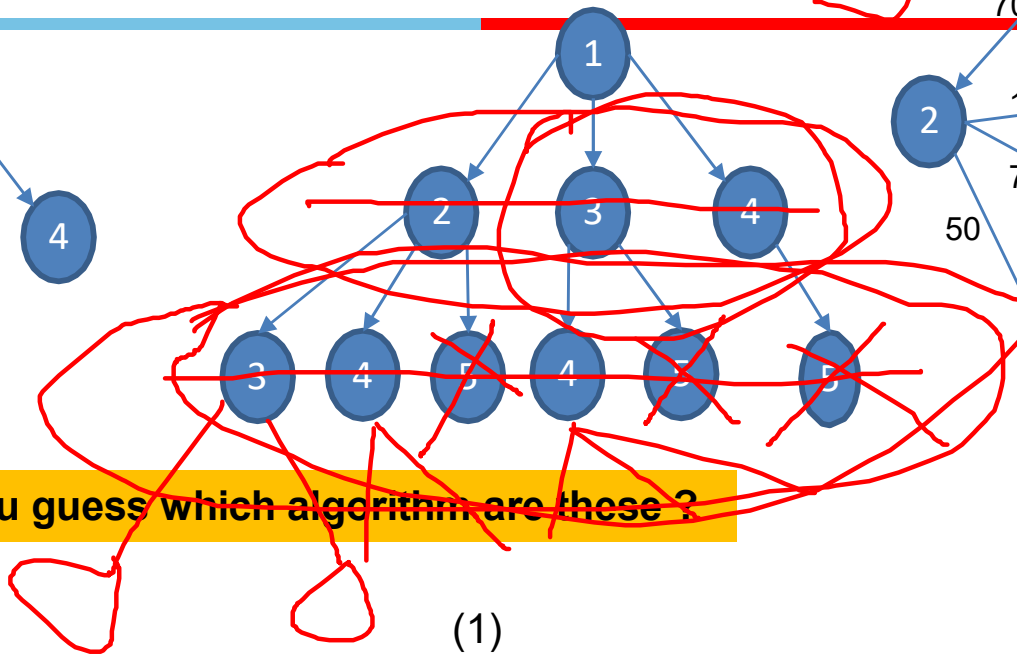
(1)
(1 2) (1 3) (1 4)
TEST FAILED

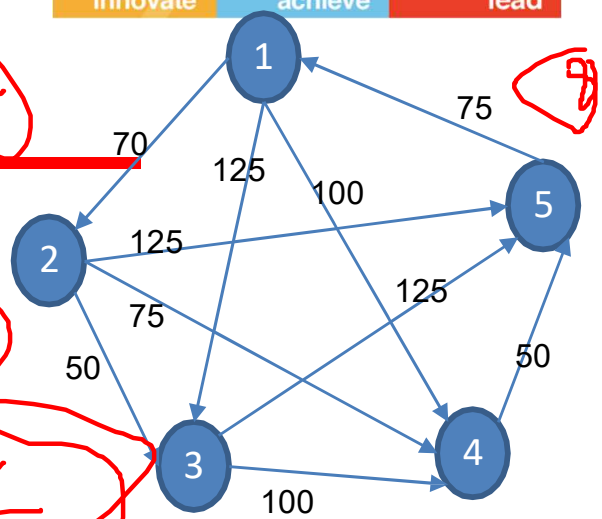
(1 3) (1 4) (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) (1 4 5)
TEST PASSED

C(1-2-5) = 70 + 125 =195
Expanded : 4
Generated : 10
Max Queue Length : 6

Search Algorithm – Uninformed Example - 2







Can you guess which algorithm are these ?

(1)
(1 2) (1 3) (1 4)
(1 2 3) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4 5) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)

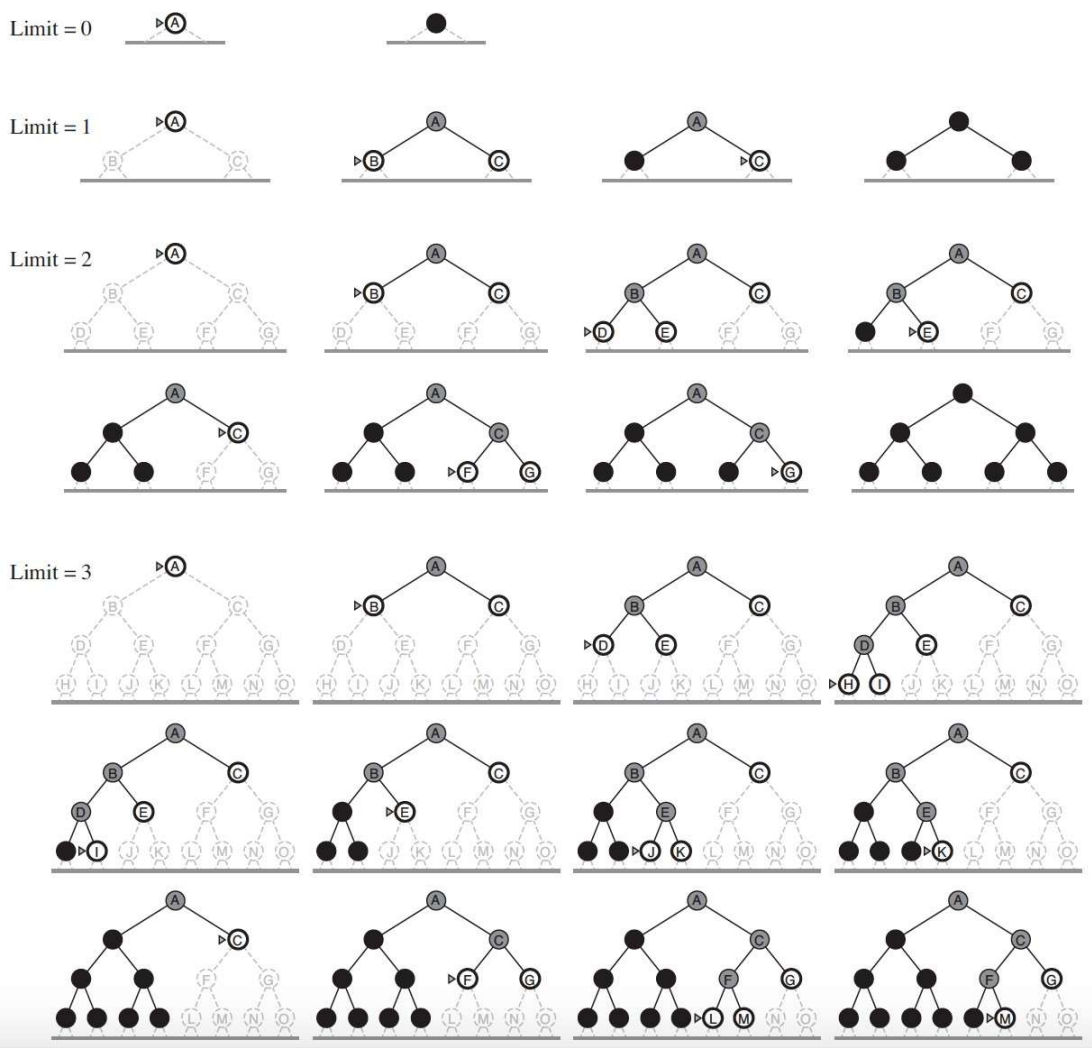
C(1-2-3-4-5) = 70 + 50 + 100 + 50 =270
Expanded : 4
Generated : 10
Max Queue Length : 6

(1)
(1 2) (1 3) (1 4)
TEST FAILED

~~(1 3)~~ ~~(1 4)~~ (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) ~~(1 4 5)~~
TEST PASSED

C(1-2-5) = 70 + 125 =195
Expanded : 4
Generated : 10
Max Queue Length : 6

Iterative Deepening Depth First Search (IDS)

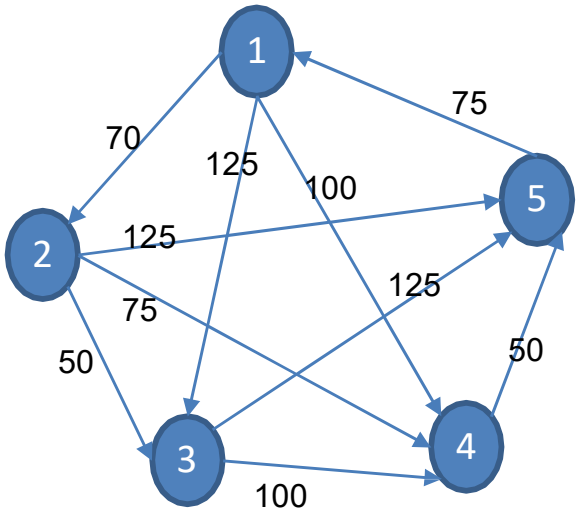


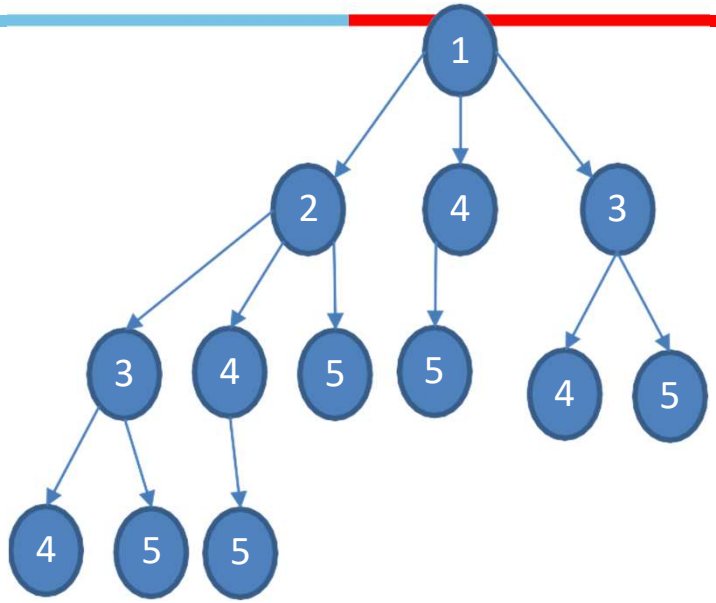
Algorithm Tracing

Students must follow this in the exams for all the search algorithms in addition to the search tree constructions. The ordering of the Open Lists must be in consistent with the algorithm with a note on the justification of the order expected!

Iter	Open List / Frontiers / Fringes	Closed List	Goal Test
1.	(1)		Fail on (1)
2.	(1 3), (1 4), (1 2)	(1)	Fail on (1 3)

UCS





(1)
(1 2 : 70) (1 4 : 100) (1 3 : 125)
TEST-F
(1 4 : 100) (1 2 3 :120)(1 3 : 125) (1 2 4 : 145) (1 2 5 : 195)
TEST-F
(1 2 3 :120)(1 3 : 125) (1 2 4 : 145) (1 4 5: 150)(1 2 5 : 195)
TEST-F
(1 3 : 125) (1 2 4 : 145) (1 4 5: 150) (1 2 3 4:170) (1 2 5 : 195) (1 2 3 5 : 245)
TEST-F
(1 2 4 : 145) (1 4 5: 150) (1 2 3 4:170) (1 2 5 : 195) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST-F
(1 4 5: 150) (1 2 3 4:170) (1 2 4 5 : 195) (1 2 5 : 195) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST - P

Uniform Cost Search

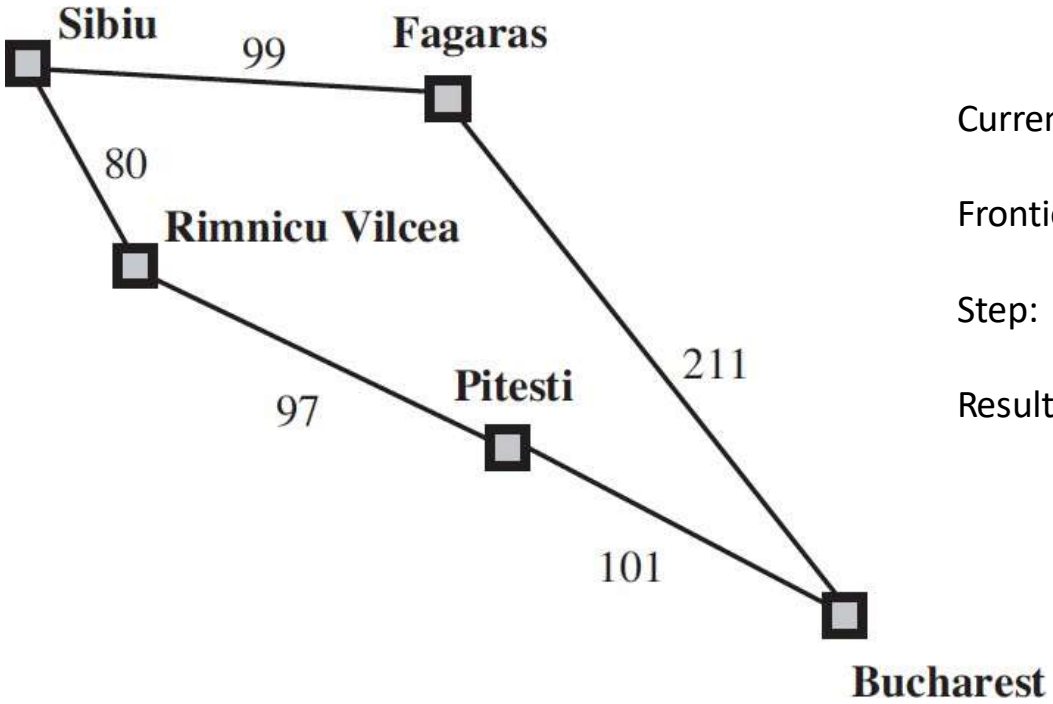
Instead of expanding the shallowest node, Uniform-Cost search expands the node n with the lowest path cost g(n)

Sorting the Frontier as a priority queue ordered by g(n)

Goal test is applied during expansion

- The goal node if generated may not be on the optimal path
- Find a better path to a node on the Frontier

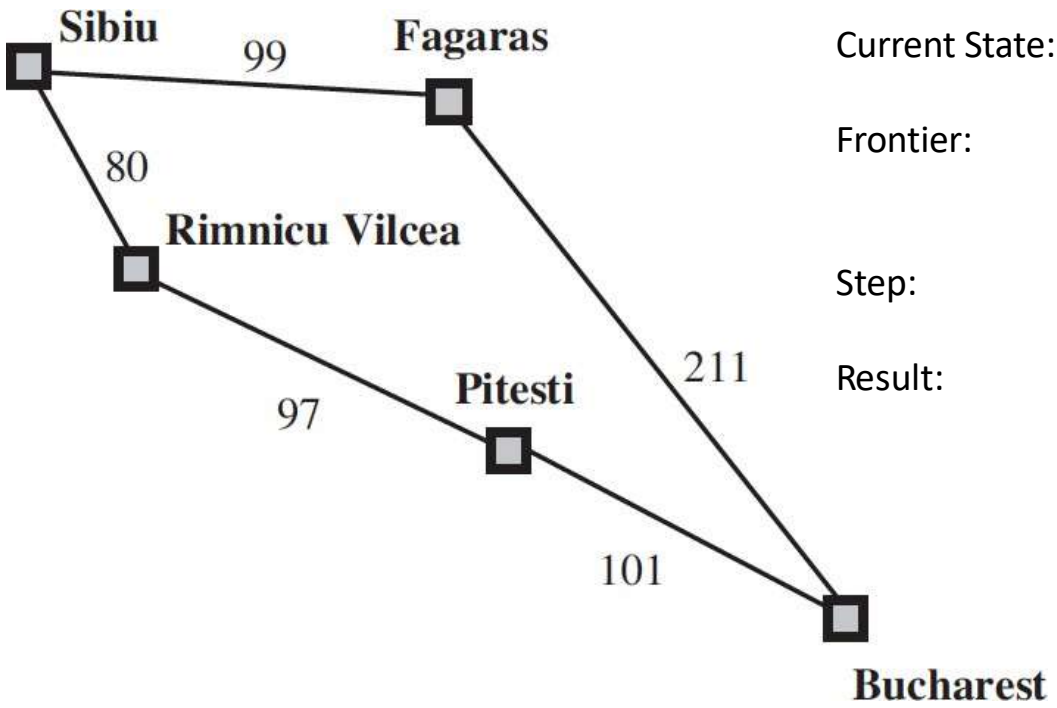
Uniform Cost Search



Current State:	Sibiu
Frontier:	[]
Step:	Expand Sibiu
Result:	Generates (“Rimnicu Vilcea” 80) (“Fagaras”, 99) Add to Frontier

Initial State:	Sibiu
Goal State:	Bucharest

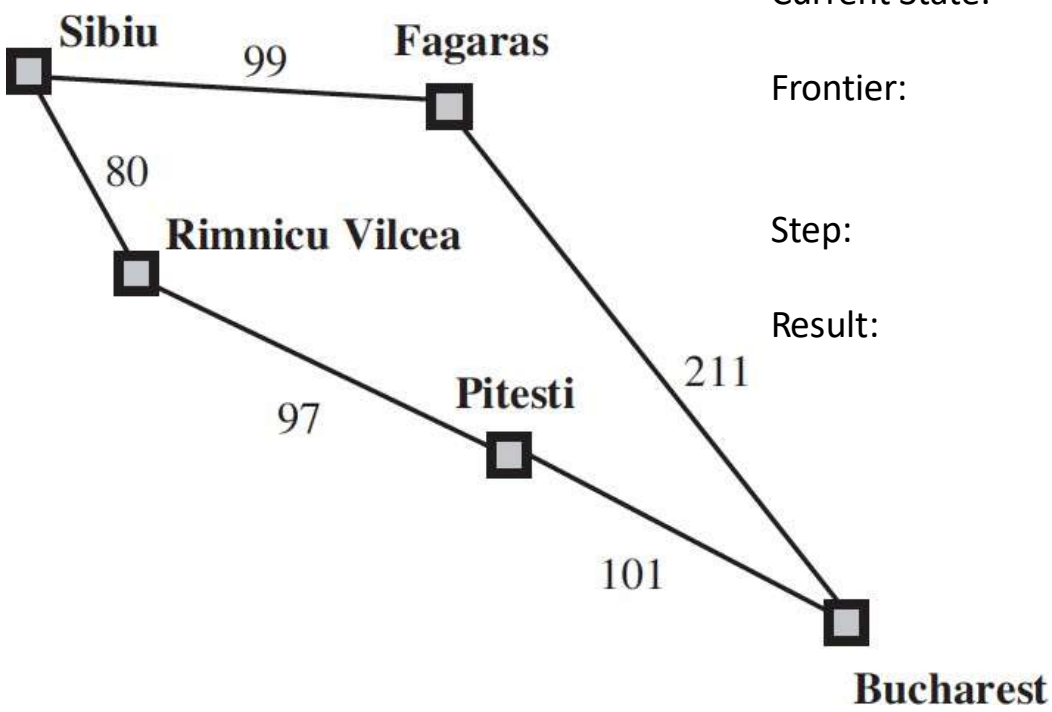
Uniform Cost Search



Current State:	Sibiu
Frontier:	[("Rimnicu Vilcea" 80) ("Fagaras", 99)]
Step:	Expand "Rimnicu Vilcea" (least cost)
Result:	Generates ("Pitesti", 177) Add to Frontier

Initial State:	Sibiu
Goal State:	Bucharest

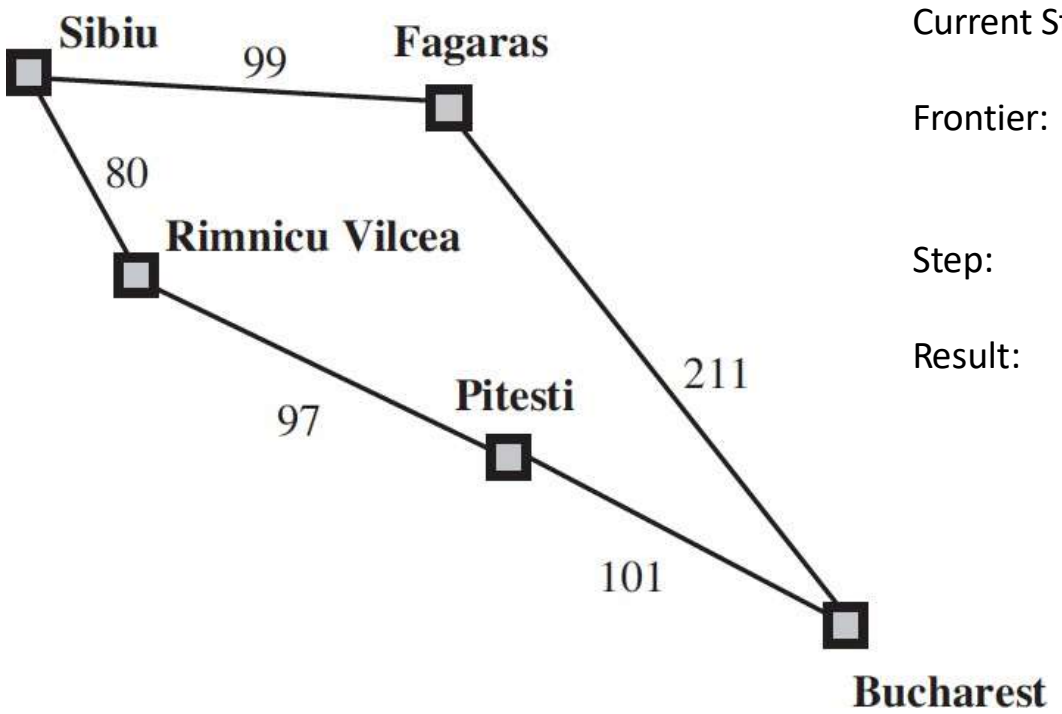
Uniform Cost Search



Current State:	Rimnicu Vilcea (not a Goal state)
Frontier:	[("Fagaras", 99) ("Pitesti", 177)]
Step:	Expand "Fagaras" (least cost)
Result:	Generates ("Bucharest", 310) Add to Frontier (It's a Goal State but we won't test during generation)

Initial State:	Sibiu
Goal State:	Bucharest

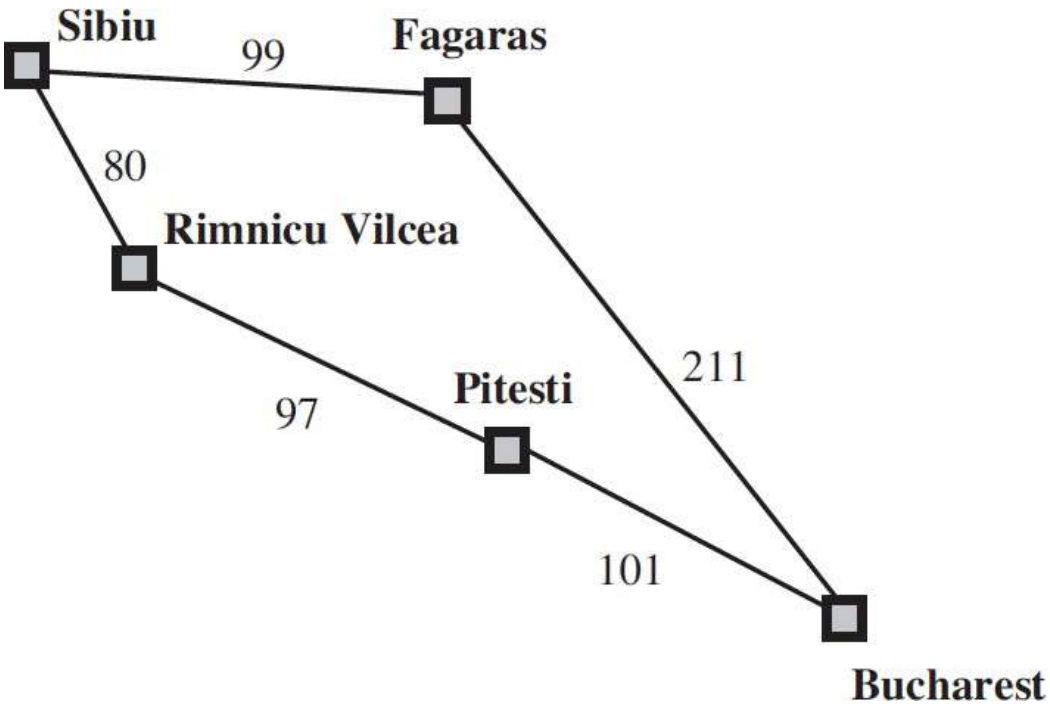
Uniform Cost Search



Current State:	Fagaras (not a goal state)
Frontier:	[("Pitesti", 177) ("Bucharest", 310)]
Step:	Expand "Pitesti" (least cost)
Result:	Generates ("Bucharest", 278) Replace in Frontier (It's a Goal State but we won't test during generation)

Initial State:	Sibiu
Goal State:	Bucharest

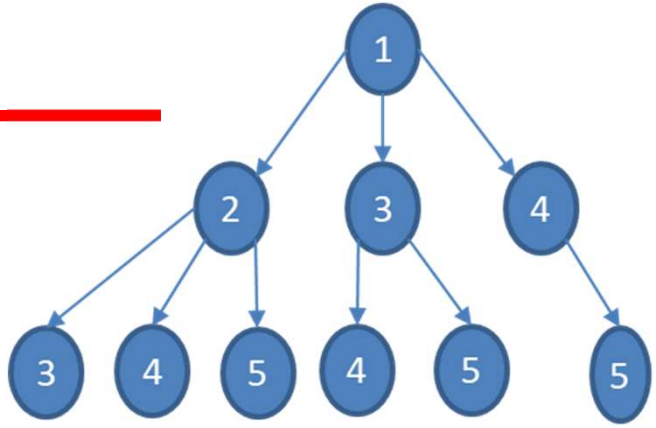
Uniform Cost Search



Current State:	Pitesti (not a goal state)
Frontier:	[("Bucharest", 278)]
Step:	Expand "Bucharest"
Result:	No further generation as Goal Test satisfied Return Solution

Initial State:	Sibiu
Goal State:	Bucharest

Sample Evaluation of the Algorithm



Complete – If the shallowest goal node is at a depth d, BFS will eventually find it by generating all shallower nodes

Optimal – Not necessarily. Optimal if path cost is non-decreasing function of depth of node. E.g., all actions have same cost

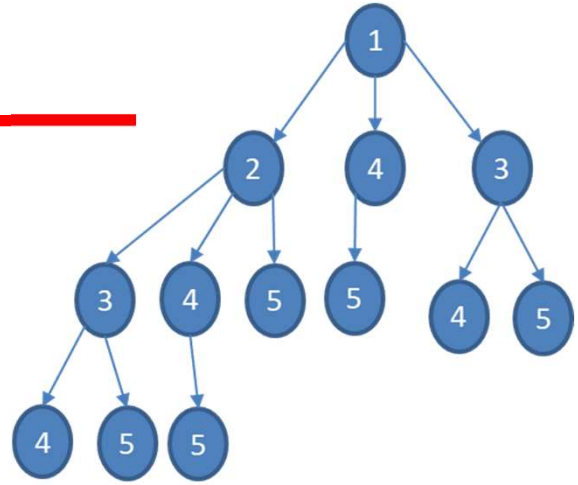
Time Complexity – $G(b^d)$ b - branching factor, d – depth

- Nodes expanded at depth 1 = b
- Nodes expanded at depth 2 = b^2
- Nodes expanded at depth d = b^d
- Goal test is applied during generation, time complexity would be $G(b^{d+1})$

Space Complexity – $G(b^d)$

- $G(b^{d-1})$ in explored set
- $G(b^d)$ in frontier set

Uniform Cost Search – Evaluation



Completeness – It is complete if the cost of every step > small +ve constant ∈

- It will stuck in infinite loop if there is a path with infinite sequence of zero cost actions

Optimal – It is Optimal. Whenever it selects a node, it is an optimal path to that node.

Time and Space complexity – Uniform cost search is guided by path costs not depth or branching factor.

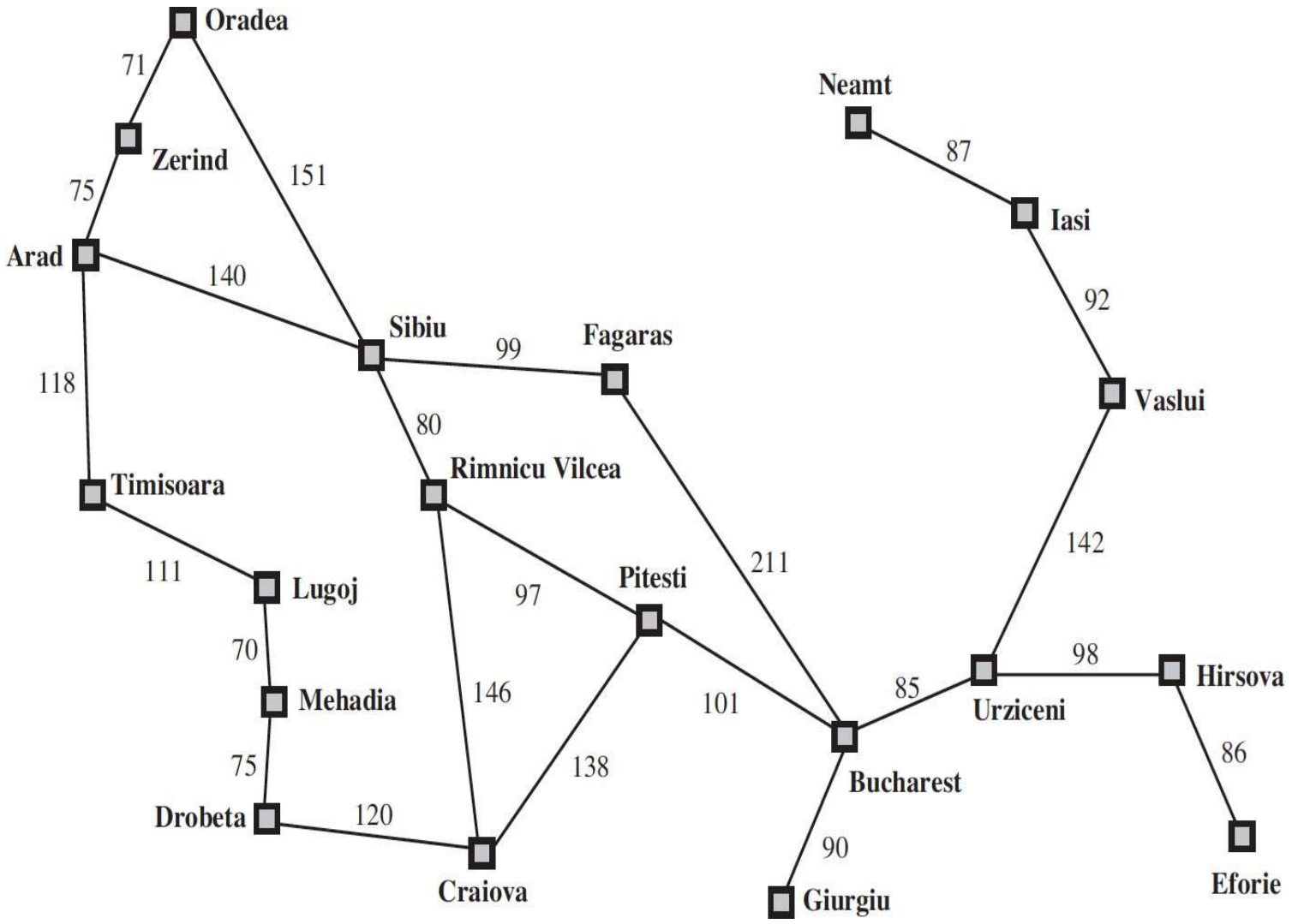
- If C* is the cost of optimal solution and ∈ is the min. action cost
- Worst case complexity = $G(b^{1+\frac{C^*}{\epsilon}})$,
- When all action costs are equal → $G(b^{d+1})$, the BFS would perform better
  - As Goal test is applied during expansion, Uniform Cost search would do extra work

Terminologies – Learnt Today

- Nodes
- States
- Frontier | Fringes
- Search Strategy : LIFO | FIFO | Priority Queue
- Performance Metrics
    - Completeness
    - Optimality
    - Time Complexity
    - Space Complexity
- Algorithm Terminology

- d Depth of a node	- m – maximum
- b Branching factor	- C* - Optimal Cost
- n – nodes	- E – least Cost
- l – level of a node	- N –total node generated

Tree Search Vs Graph Search


Oradea
71
Zerind
151
75
Arad
140
Sibiu
99
Fagaras
118
80
Rimnicu Vilcea
Timisoara
111
Lugoj
97
Pitesti
211
70
Mehadia
146
101
75
138
Drobeta
120
Craiova
Neamt
87
Iasi
92
Vaslui
142
98
Hirsova
85
Urziceni
86
Bucharest
90
Giurgiu
Eforie

Coding Aspects

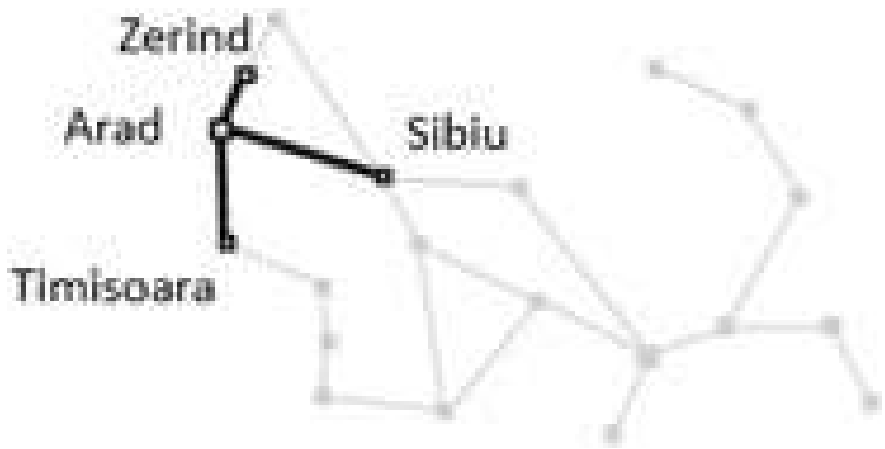
For each node n of the tree,

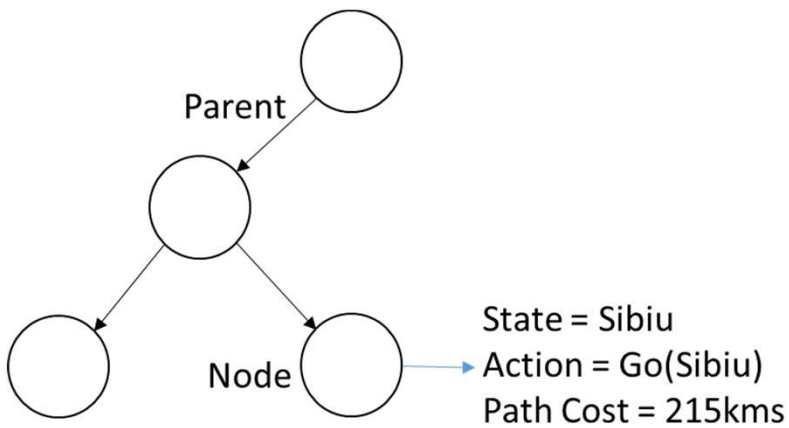
n.STATE : the state in the state space to which node corresponds

n.PARENT : the node in the search tree that generated this node

n.ACTION : the action that was applied to parent to generate the node

n.PATH-COST : the cost, denoted by g(n), of the path from initial state to node





Algorithm Tracing

Students must follow this in the exams for all the search algorithms in addition to the search tree constructions. The ordering of the Open Lists must be in consistent with the algorithm with a note on the justification of the order expected!

Iter	Open List / Frontiers / Fringes	Goal Test
1.	(1)	Fail on (1)
2.	(1 3), (1 4), (1 2)	Fail on (1 3)

Tree Search Algorithms

```
function Tree-Search (problem, strategy) returns a solution, or failure
        initialize the search tree using the initial state of problems
        loop do
                if there are no candidate for expansion
                        then return failure
                choose: leaf node for expansion according to strategy
                if the node contains a goal state
                        then return the corresponding solution
                else
                        Expand the node
                        Add the resulting nodes to the search tree
        end
```

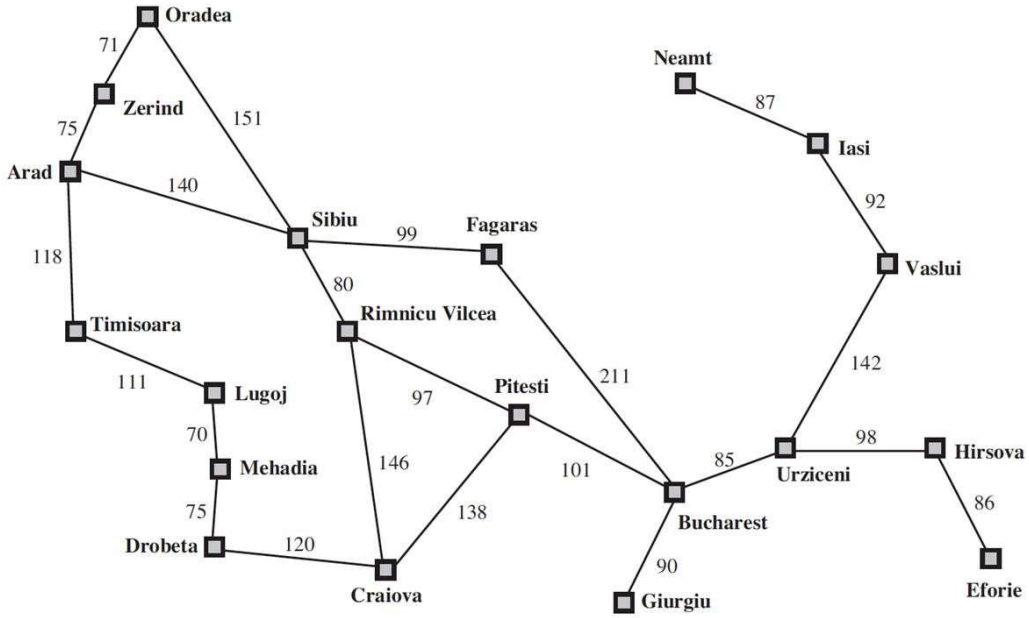
Tree Search Vs Graph Search Algorithms

Coding Aspects

Need:

Redundant Path Problem : More than one way to reach a state from another.

Infinite Loop Path Problem



Start : Arad

Goal : Craiova

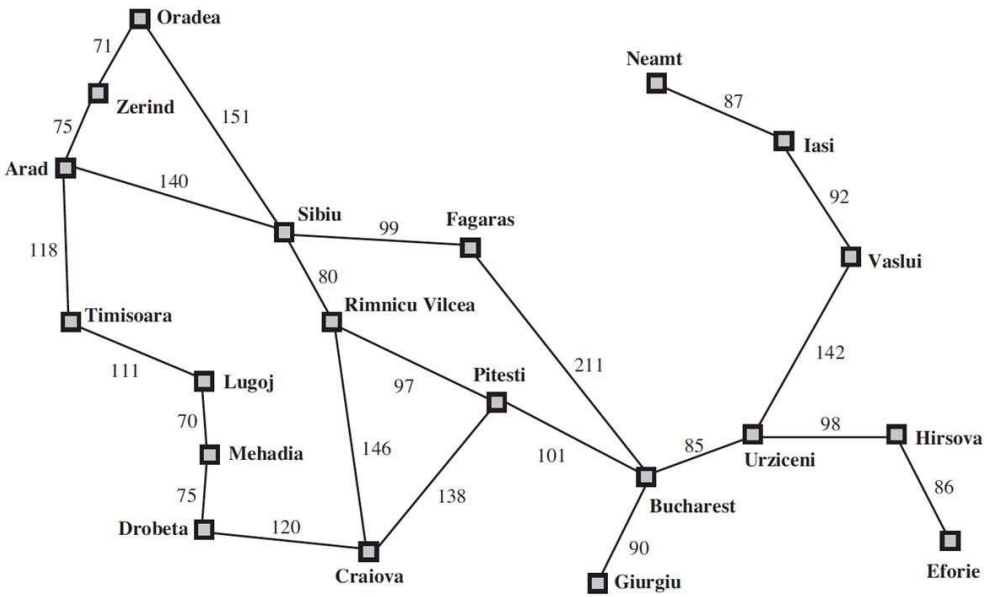
Tree Search Vs Graph Search Algorithms

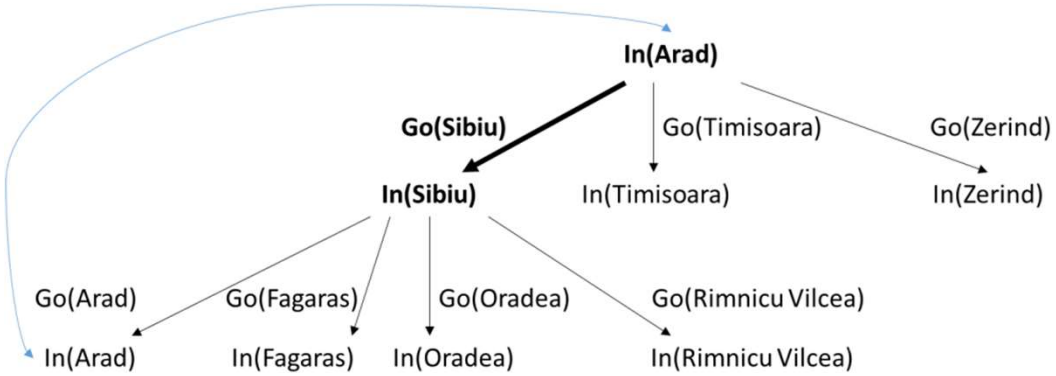
Coding Aspects

Need:

Redundant Path Problem

Infinite Loop Path Problem: Repeated State generated by looped path existence.





Start : Arad

Goal : Craiova

Algorithm Tracing

Students must follow this in the exams for all the search algorithms in addition to the search tree constructions. The ordering of the Open Lists must be in consistent with the algorithm with a note on the justification of the order expected!

Iter	Open List / Frontiers / Fringes	**Closed List**	Goal Test
1.	(1)		Fail on (1)
2.	(1 3), (1 4), (1 2)	(1)	Fail on (1 3)

Search

Coding Aspects

For each node n of the tree,

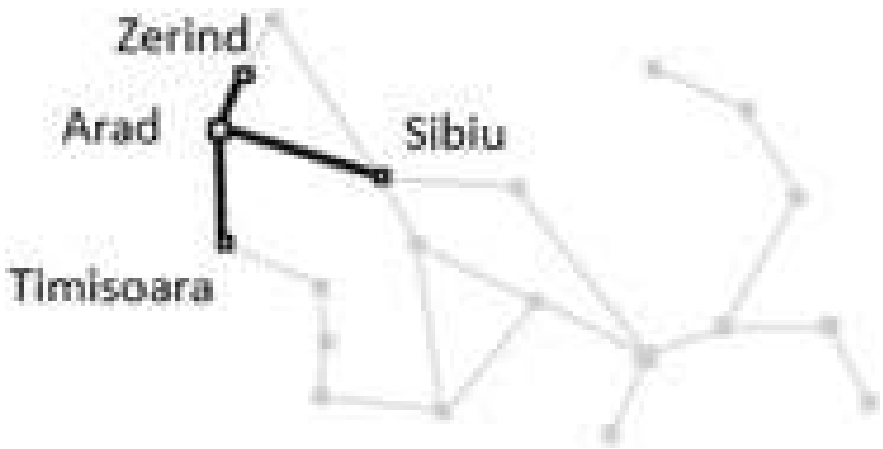
n.STATE : the state in the state space to which node corresponds

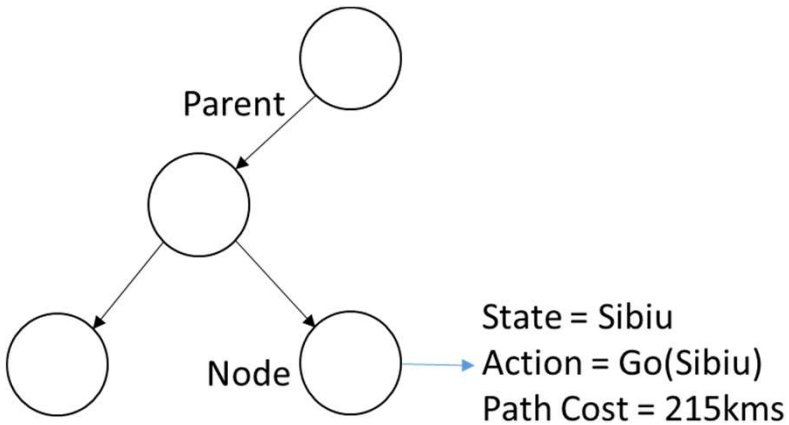
n.PARENT : the node in the search tree that generated this node

n.ACTION : the action that was applied to parent to generate the node

n.PATH-COST : the cost, denoted by g(n), of the path from initial state to node

n.VISITED : the boolean indicating if the node is already visited and tested (**or)** a global SET of visited nodes



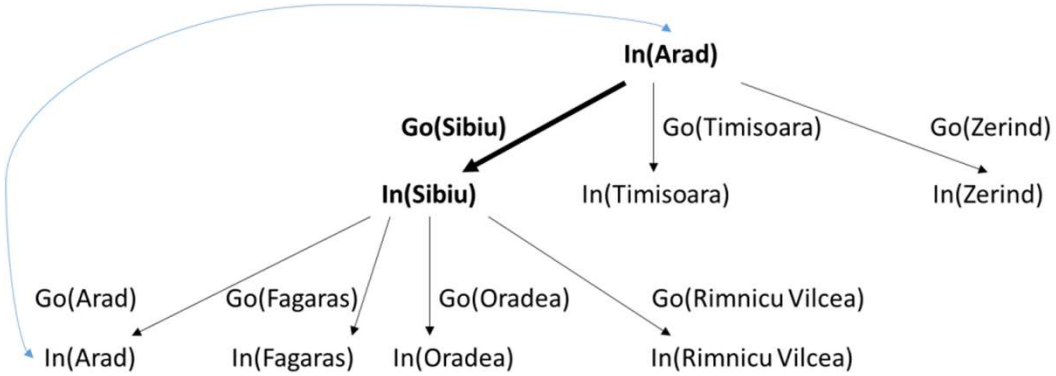


Tree Search Vs Graph Search Algorithms

Coding Aspects

Graph-Search Algorithm

Augments the Tree-Search algorithm to solve redundancy by keeping track of states that are already visited called as **Explored Set**. **Only one copy of each state is maintained/stored.**



Graph Search Algorithms

```
function Graph-Search (problem, fringe) returns a solution, or failure
        initialize the search space using the initial state of problems memory to store
the visited fringe
        closed ← an empty set
        fringe ← Insert(Make-Node(Initial-State[problem]), fringe)
        loop do
                if fringe is empty
                        then return failure
                node ← Remove-Front(fringe)
                if the node contains a goal state
                        then return the corresponding solution
                else
                        if the node is not in closed ie., not visited yet
                            Add the node to the closed set
                            Expand all the fringe of the node
                            Add all expanded sorted successors into the fringe
        end
```

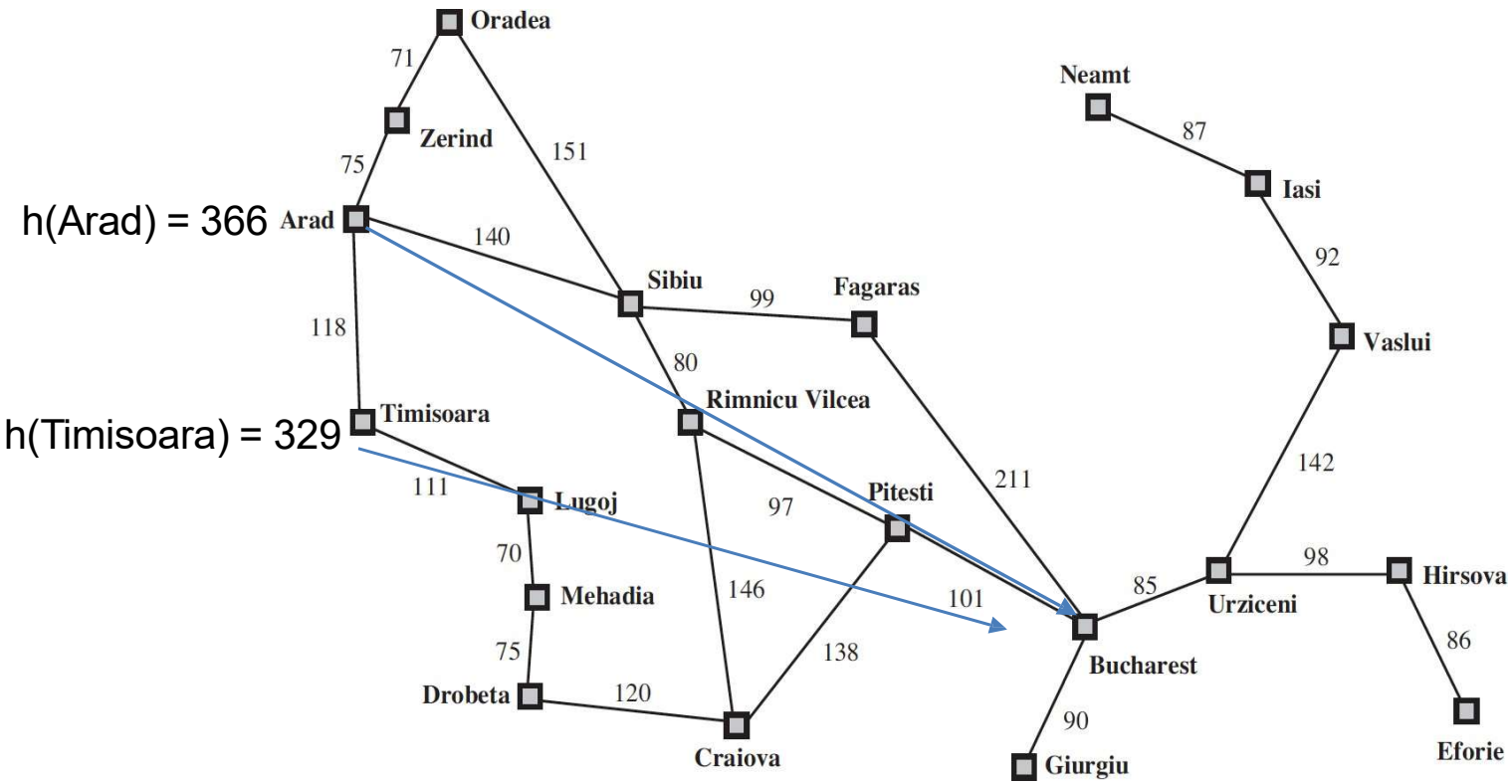
Informed Search

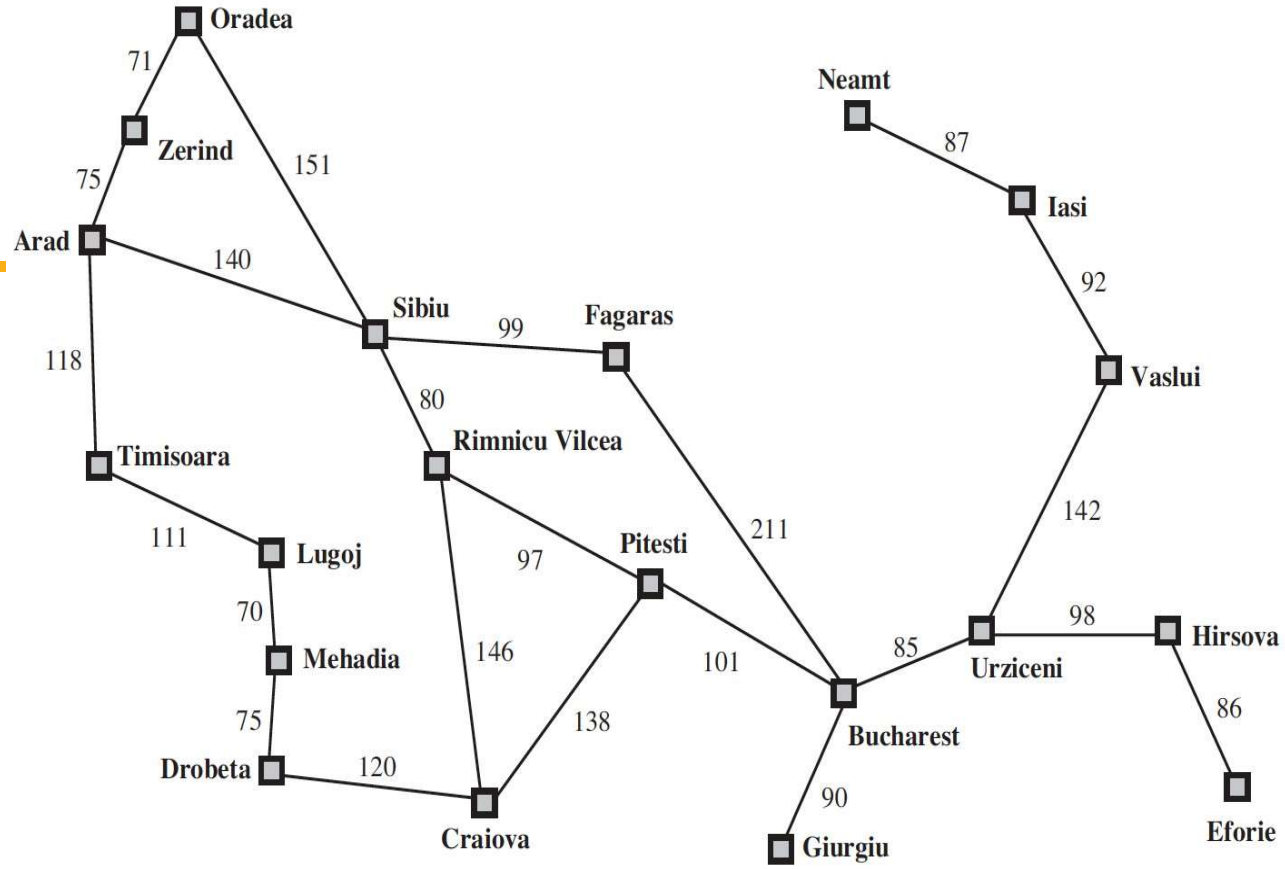
Greedy Best First

A*

Informed /Heuristic Search

Strategies that know if one non-goal state is more promising than another non-goal state





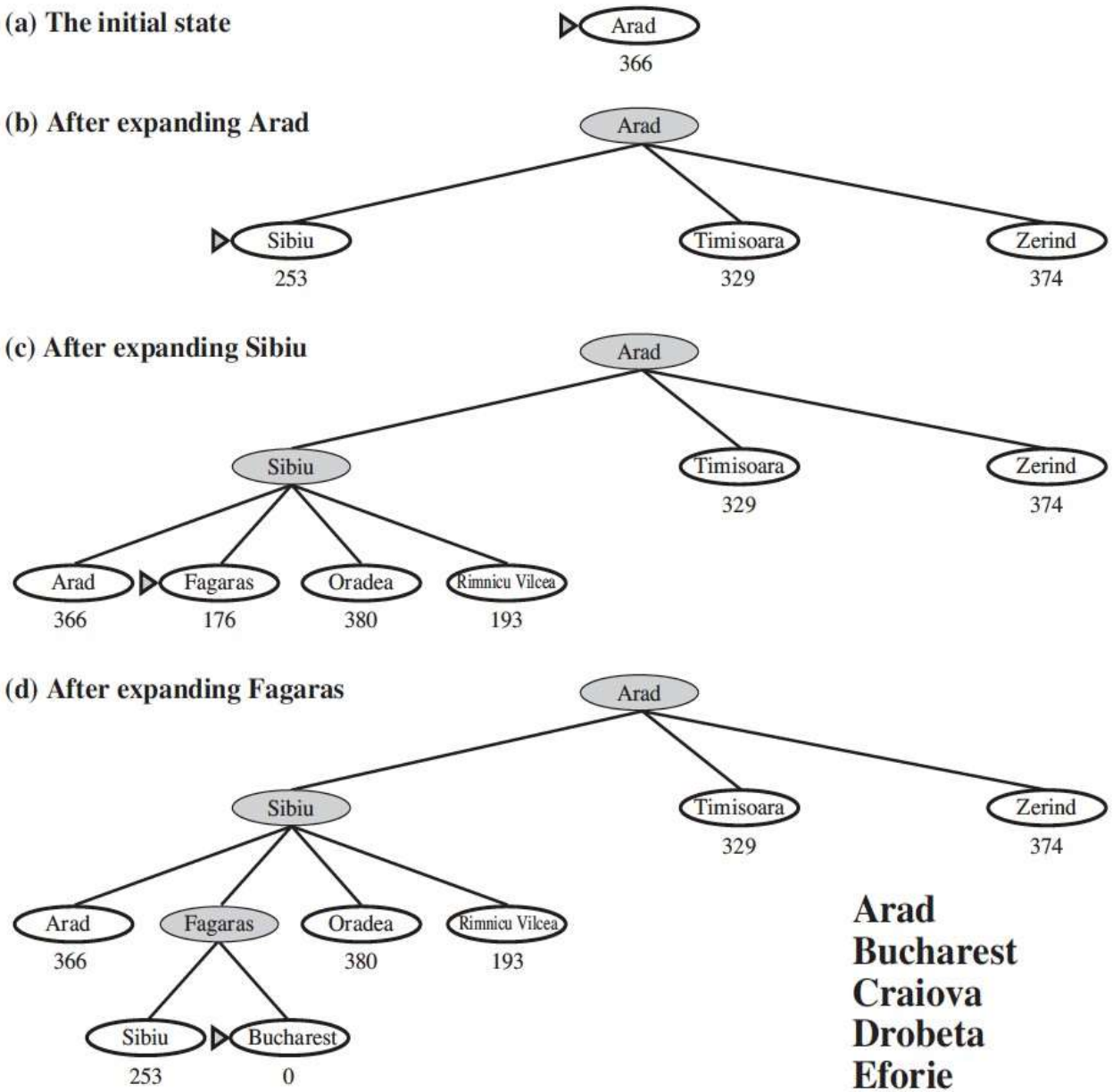
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy Best First Search

Expands the node that is closest to the goal

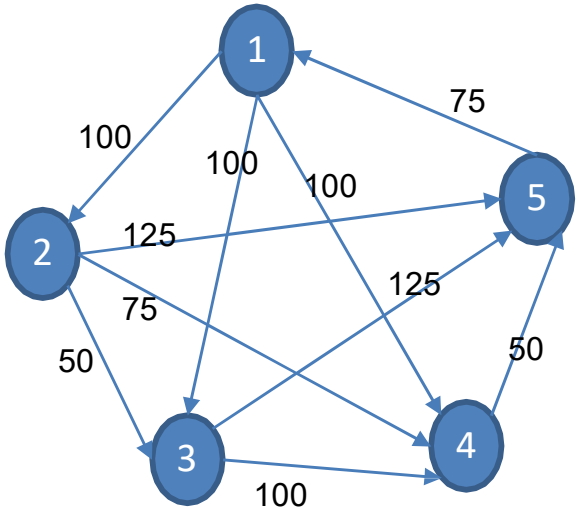
Thus, **f(n) = h(n)**

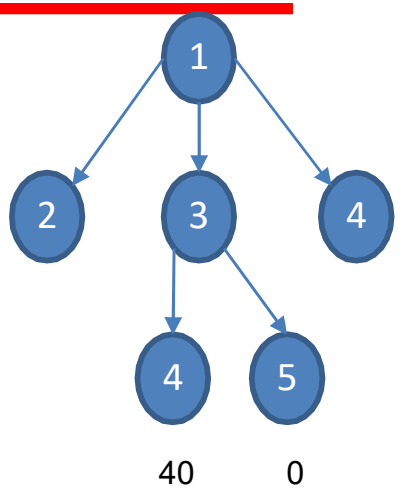
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374



Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Greedy Best First Search



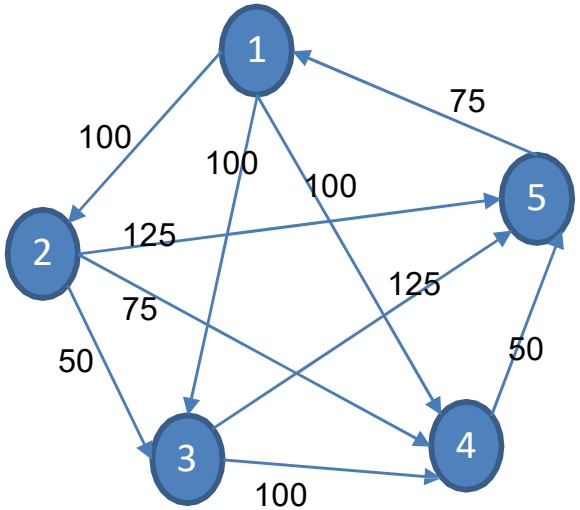


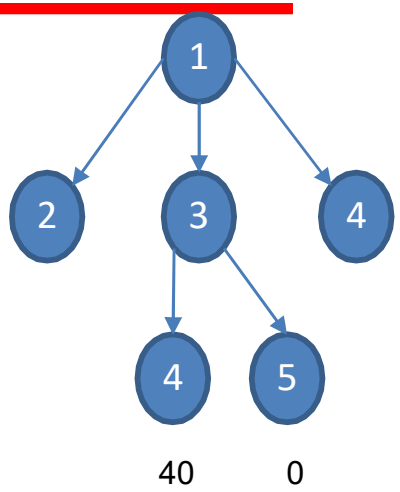
n	h(n)
1	60
2	120
3	30
4	40
5	0

(1)
(1 3) (1 4) (1 2)
(1 3 5) (1 3 4)

C(1-3-5) = 100 + 125 =225
Expanded : 2
Generated : 6
Max Queue Length : 3

Greedy Best First Search





n	h(n)
1	60
2	120
3	30
4	40
5	0

(1)
(1 3) (1 4) (1 2)
(1 3 5) (1 3 4)

C(1-3-5) = 100 + 125 =225
Expanded : 2
Generated : 6
Max Queue Length : 3

A* Search

Expands the node which lies in the closest path (estimated cheapest path) to the goal

Evaluation function **f(n) = g(n) + h(n)**

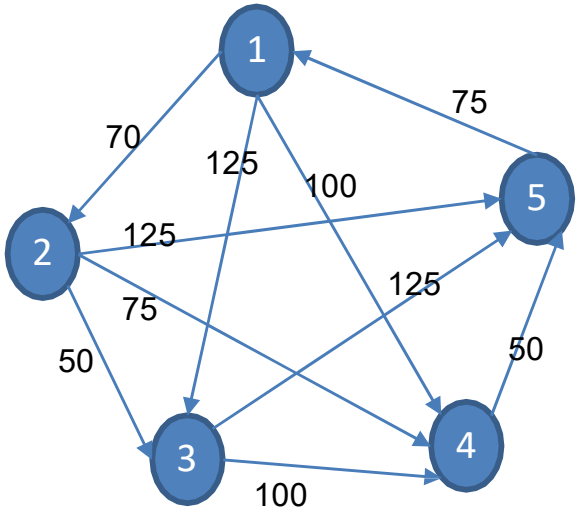
g(n) – the cost to reach the node

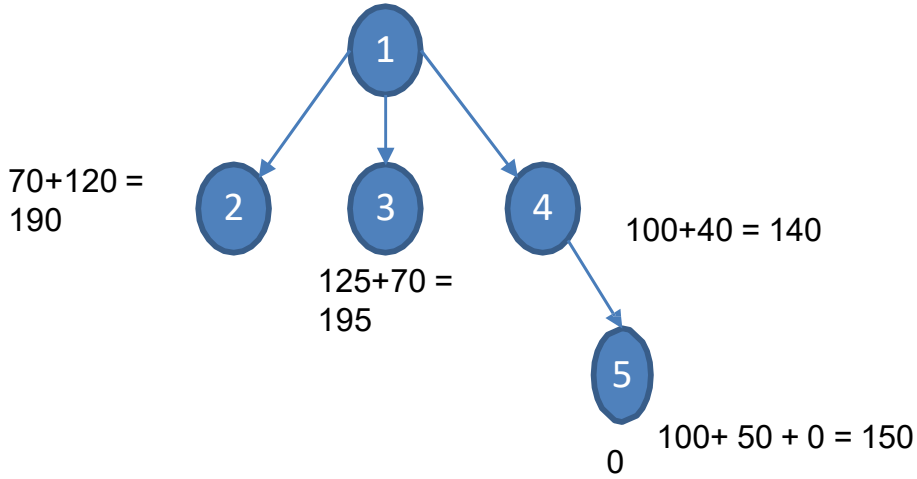
h(n) – the expected cost to go from node to goal

f(n) – estimated cost of cheapest path through node n

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

A* Search





n	h(n)
1	60
2	120
3	70
4	40
5	0

(1)
(1 4) (1 2) (1 3)
(1 4 5) (1 2) (1 3)

C(1-4-5) = 100 + 150 =150
Expanded : 2
Generated : 5
Max Queue Length : 3

Required Reading: AIMA - Chapter #3: 3.1, 3.2, 3.3, 3.4, 3.5

Next Class Plan :
Informed Search : GFBS & A*
Heuristic Design

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CLZG557

M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

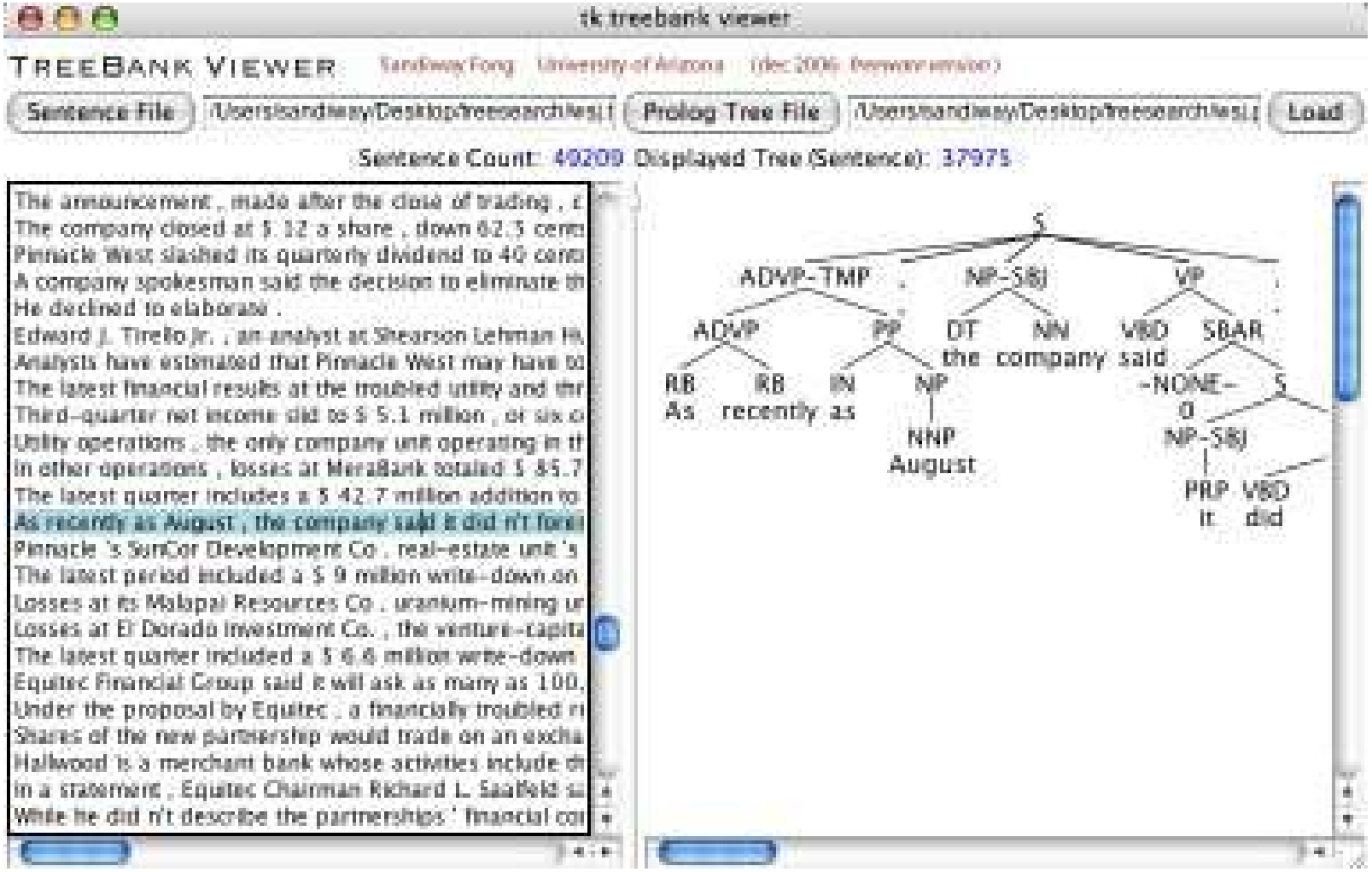
D. Local Search Algorithms & Optimization Problems

Learning Objective

At the end of this class , students Should be able to:

1. Apply A* variations algorithms to the given problem
2. Compare given heuristics for a problem and analyze which is the best fit
3. Differentiate between informed and local search requirements
4. Design relaxed problem with appropriate heuristic design

Case Study – 1 Search in Treebanks

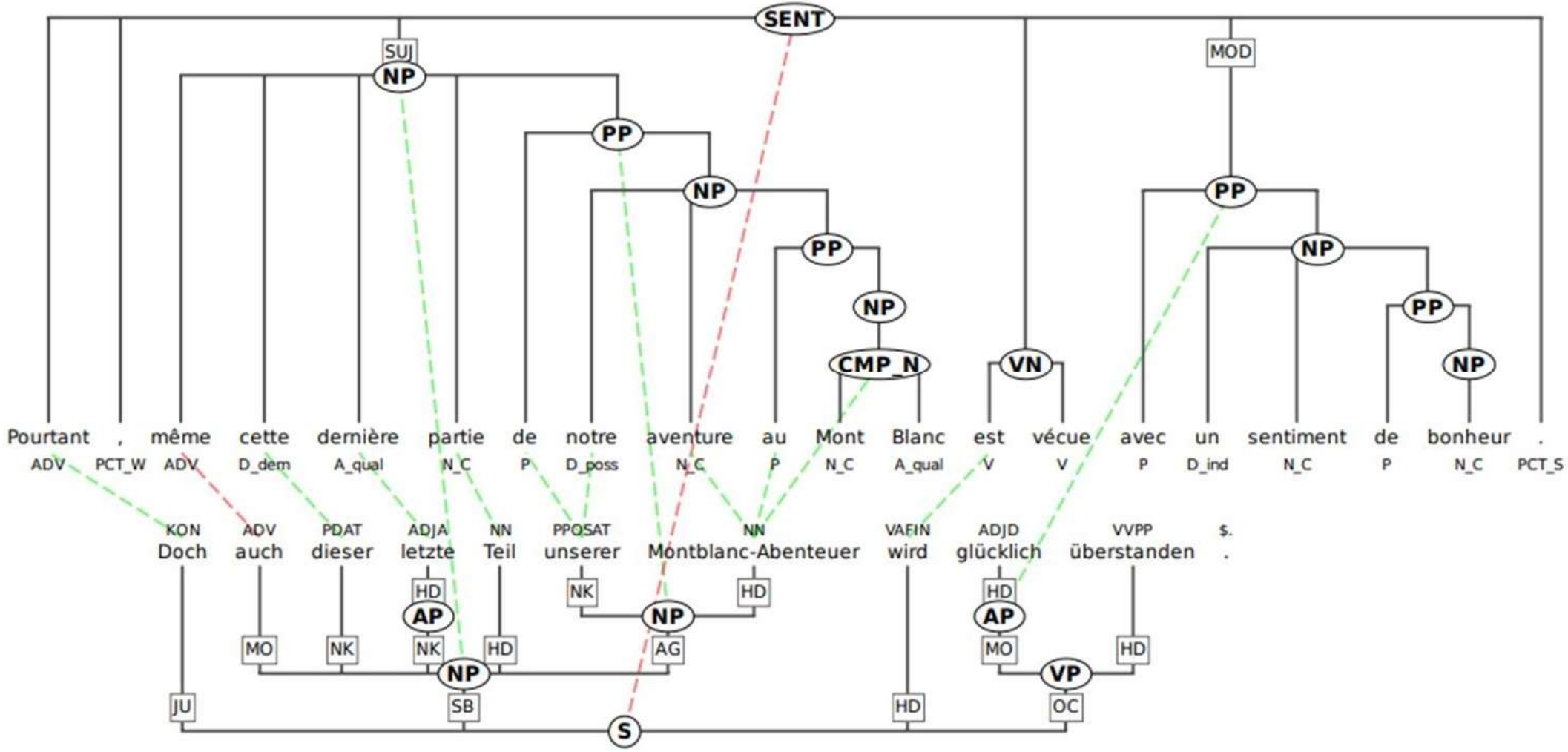


Source Credit :

https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

https://ufal.mff.cuni.cz/pdt3.5

Case Study – 1 Search in Treebanks

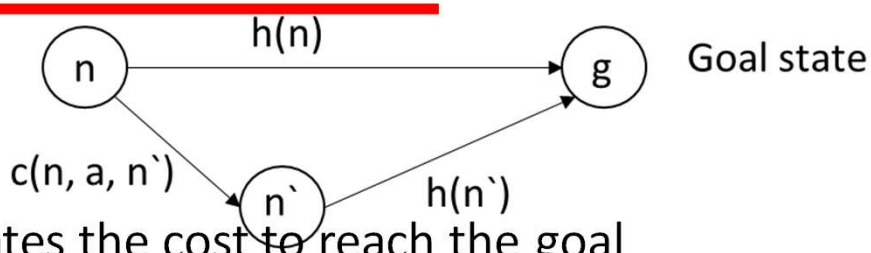


Source Credit :

https://catalog.ldc.upenn.edu/docs/LDC95T7/cl93.html

https://ufal.mff.cuni.cz/pdt3.5

A* Search



<u>Optimal on condition</u>

h(n) must satisfies two conditions:

- Admissible Heuristic – one that never overestimates the cost to reach the goal
- Consistency – A heuristic is consistent if for every node n and every successor node n` of n generated by action a, h(n) <= c(n, a, n`) + h(n`)

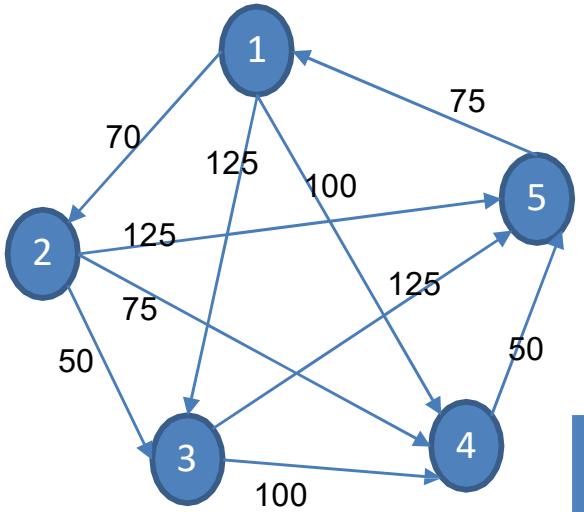
<u>Complete</u>

- If the number of nodes with cost <=C* is finite
- If the branching factor is finite
- A* expands no nodes with f(n) > C*, known as pruning

<u>Time Complexity</u> - $G\left(b^{\Delta}\right)$ where the absolute error $\Delta = h^{*} - h$

A* Search

Is the heuristic designed leads to optimal solution?

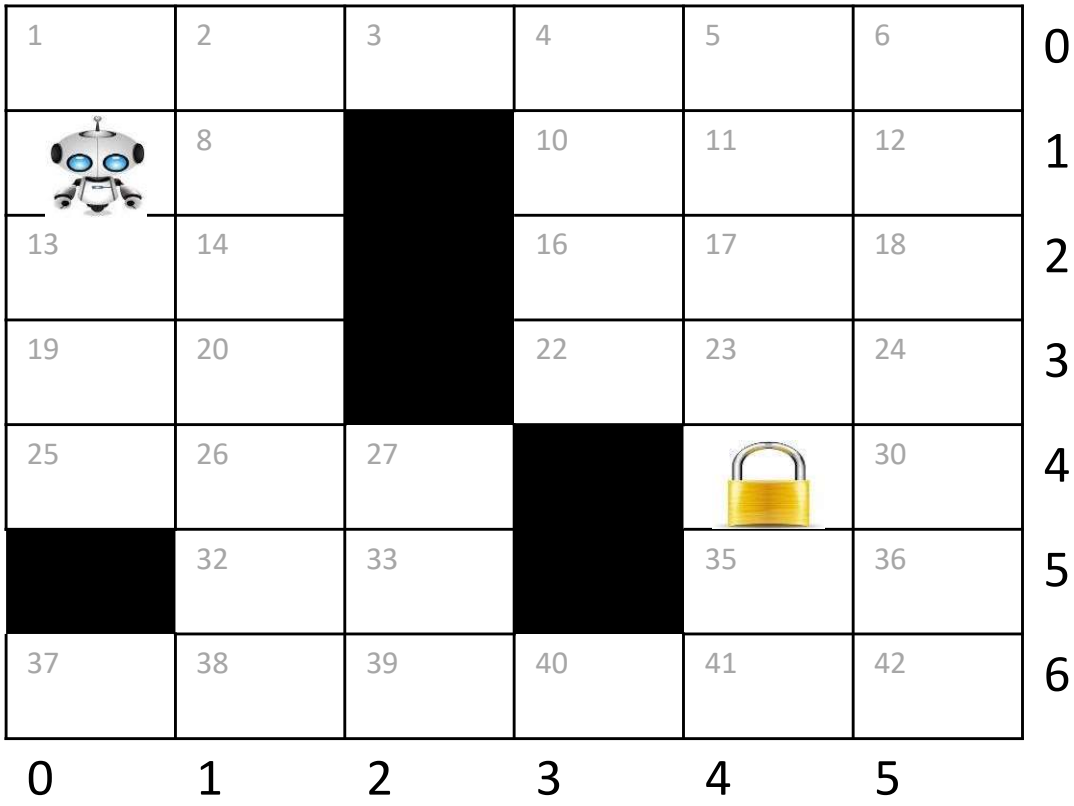


Assuming node 3 as goal, taking only sample edges per node below is checked for consistency

n	h(n)	Is Admissible? h(n) <= h*(n)	Is Consistent? For every arc (i,j): h(i) <= g(i,j) + h(j)
1	80	Y	N (5→1) : 190<=155
2	60	N	Y (1→2) : 80<=130
3	0	Y	
4	200	Y	Y (1→4) : 80<=300 Y (2→4) : 60<=275
5	190	Y	Y (2→5) : 60<=315 Y (4→5) : 200<=240

Path finding Robot – Sample Planning Agent Design

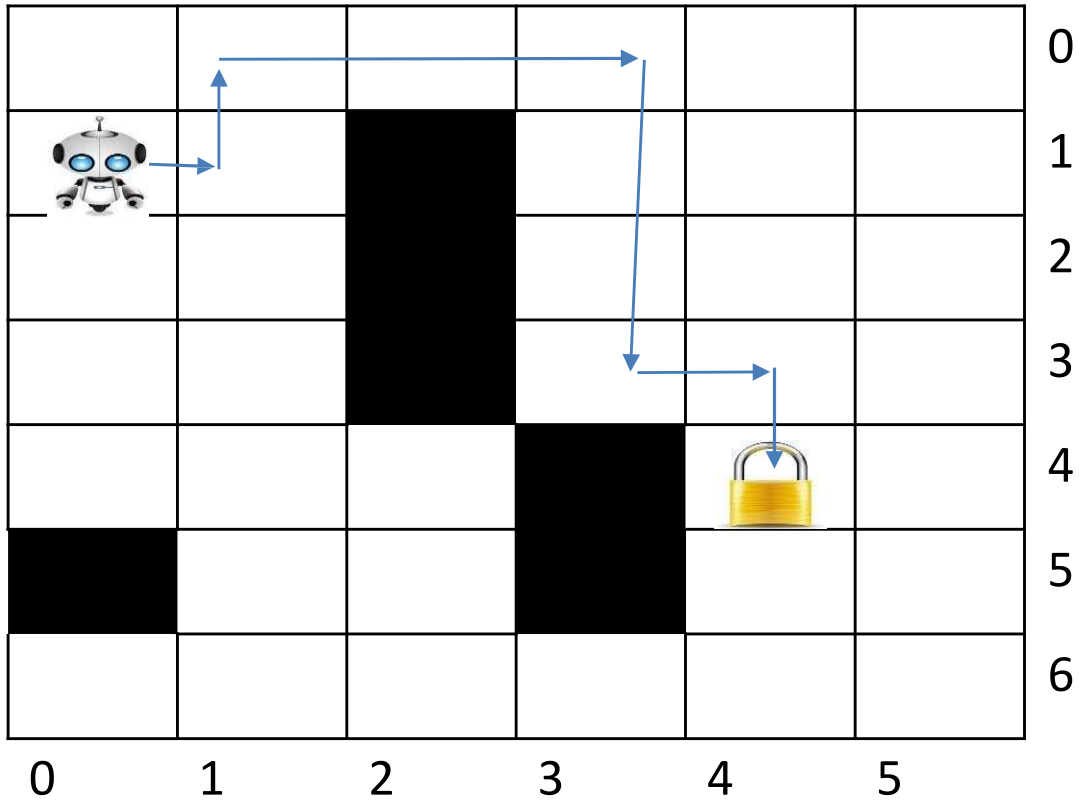
Successor Function Design



N-W-E-S

A*

Demo

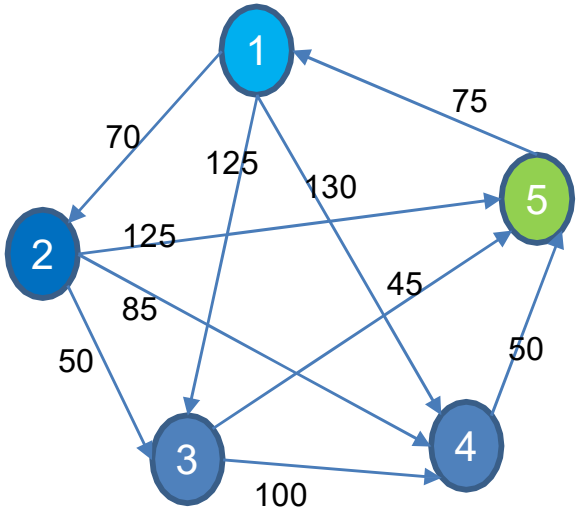


Variations of A*

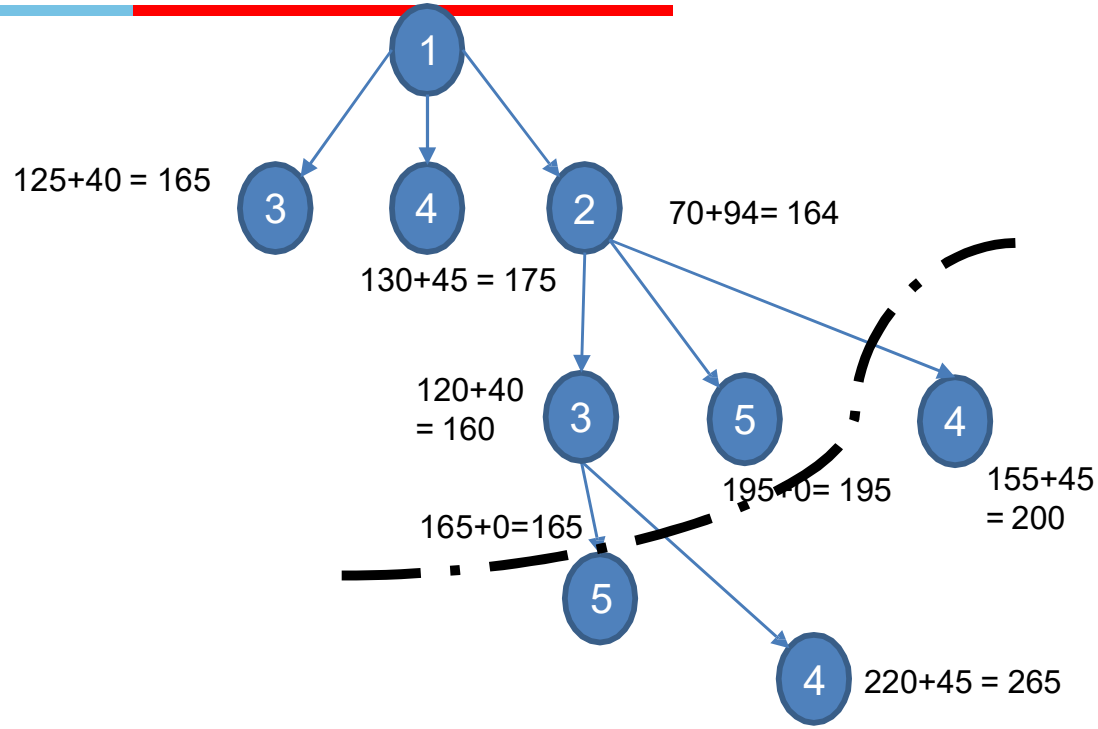
Memory Bounded Heuristics

Iterative Deepening A*

Set limit for f(n)



n	h(n)
1	60
2	94
3	40
4	45
5	0



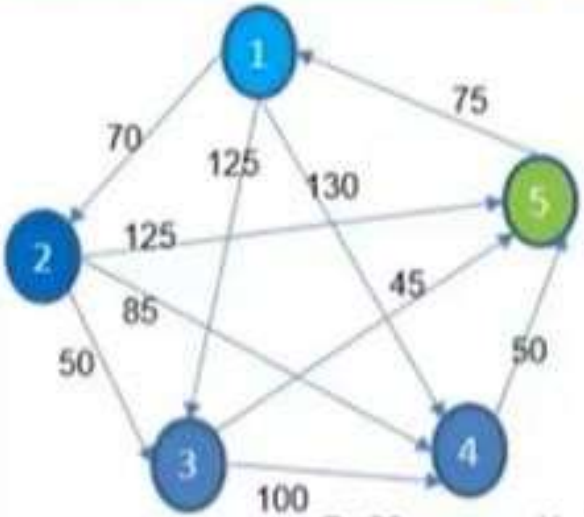
Cut off value is the smallest of f-cost of any node that exceeds the cutoff on previous iterations

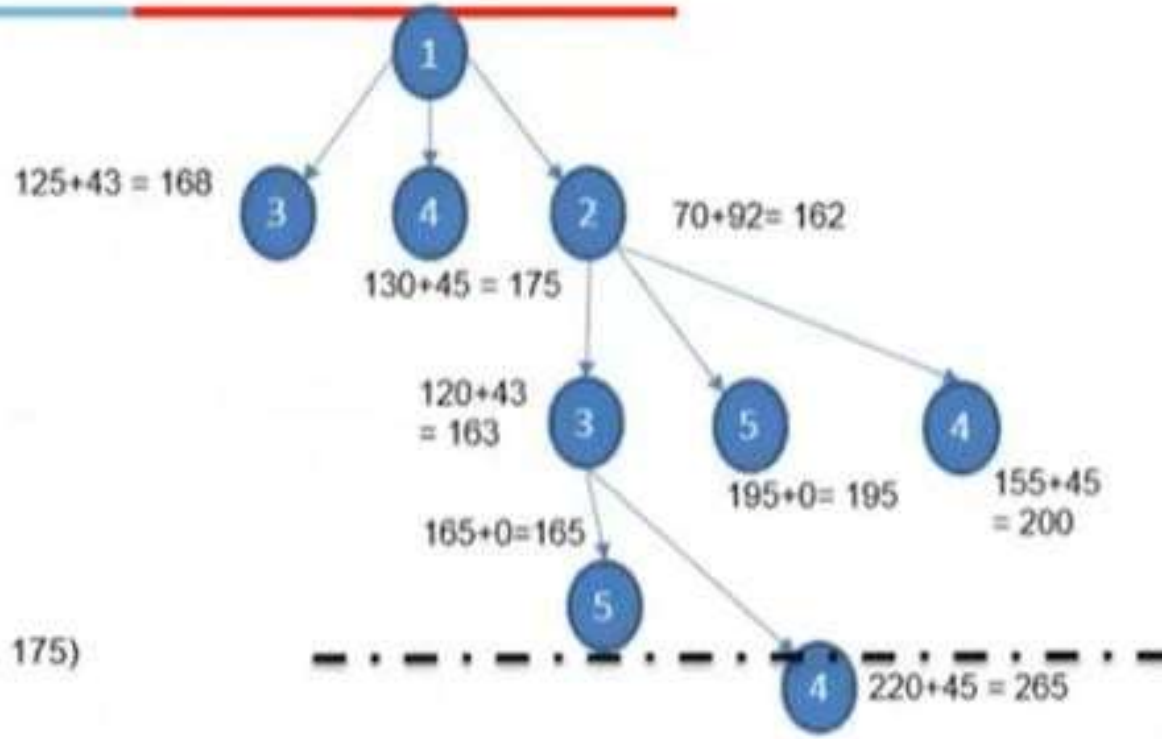
Iterative Limit : Eg

f(n) = 180
f(n) = 195
f(n) = 200
.
.
.
.

Iterative Deepening A*

Set limit for f(n)





n	h(n)
1	60
2	92
3	43
4	45
5	0

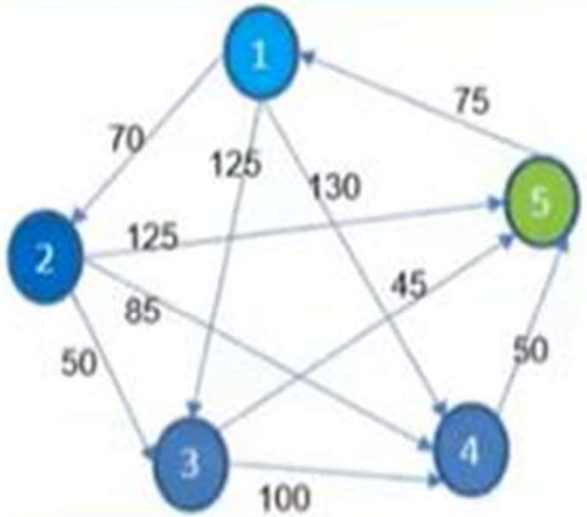
B=60
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)

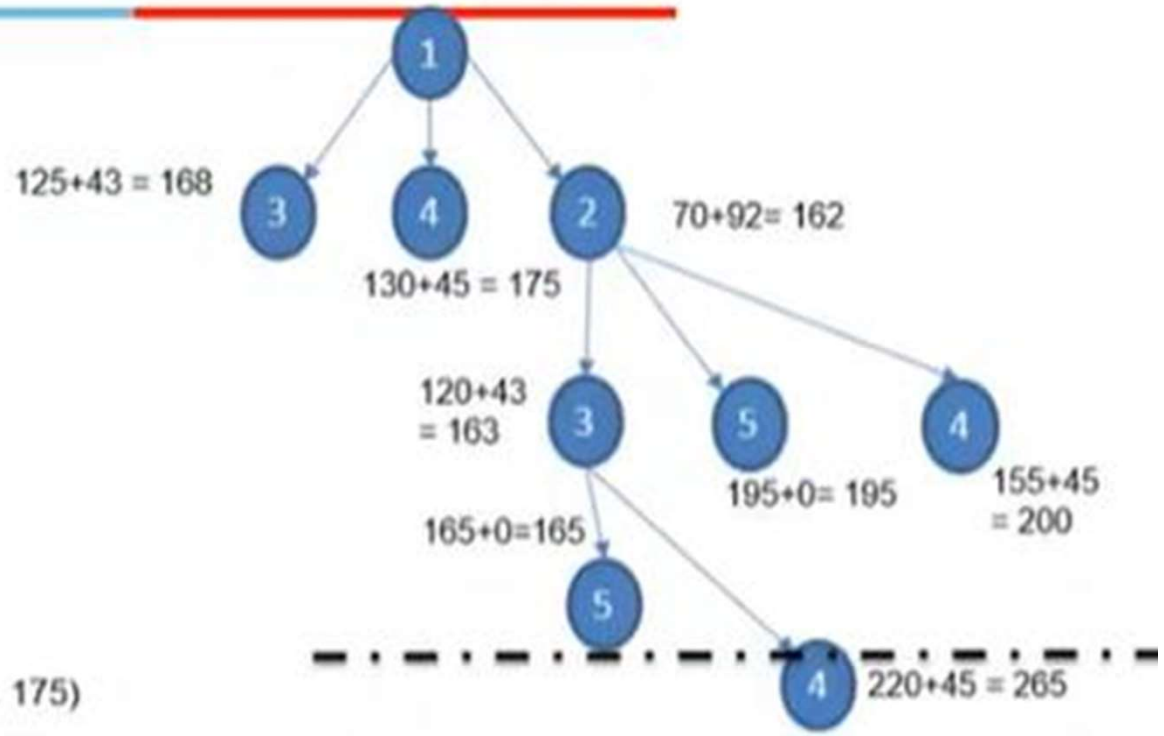
B=162
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

B=163
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F

Iterative Deepening A*

Set limit for f(n)





n	h(n)
1	60
2	92
3	43
4	45
5	0

B=163

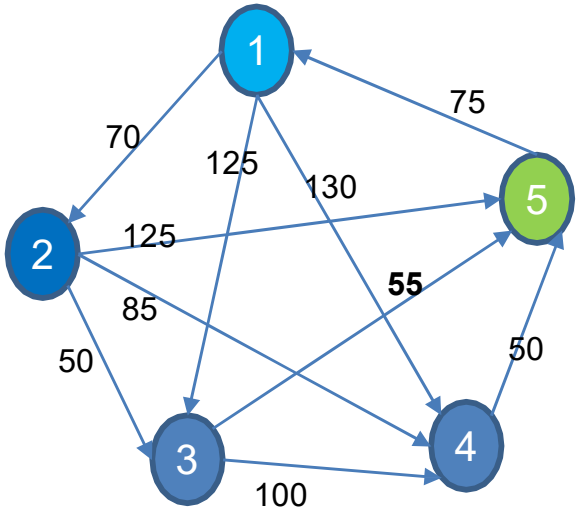
(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3 5: 165) (1 2 3 4: 265) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

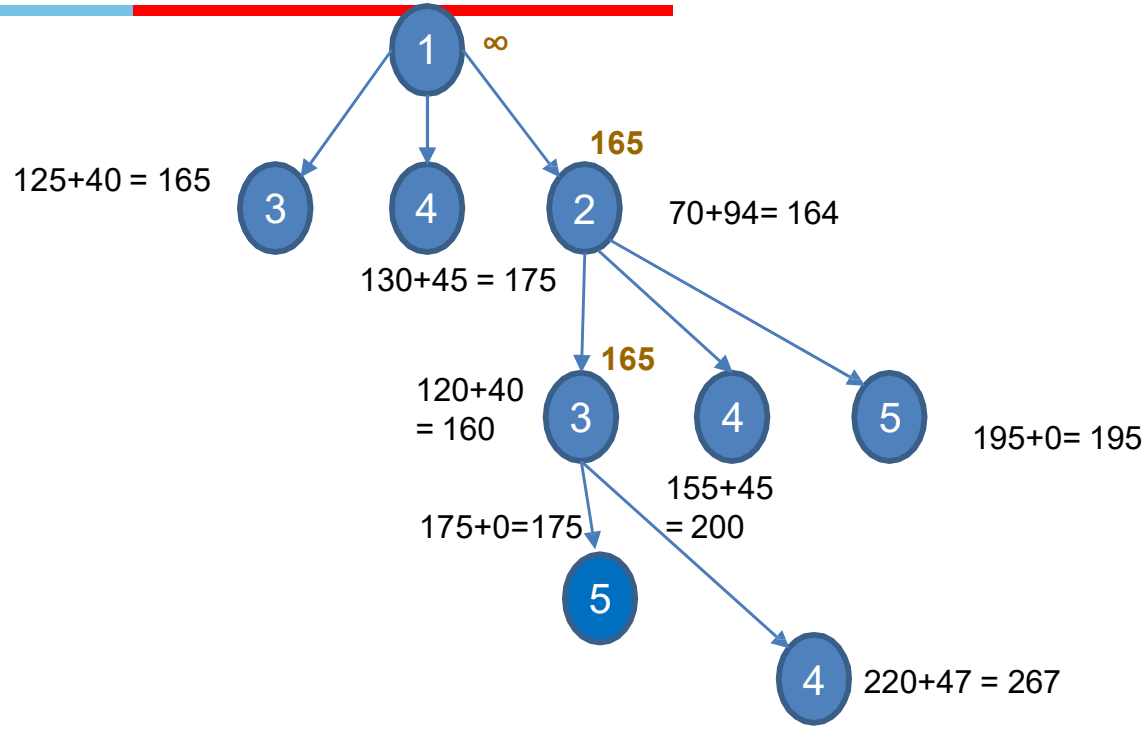
B=165

(1: 60)
TEST-F
(1 2: 162) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3: 163) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)
TEST-F
(1 2 3 5: 165) (1 2 3 4: 265) (1 2 4: 200) (1 2 5: 195) (1 3: 168) (1 4: 175)

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate





n	h(n)
1	60
2	94
3	40
4	45
5	0

(1, 60)
(1 2 | 164) (1 3 | 165) (1 4 | 175)

(1 2 | 175) (1 4 | 175) (1 3 | 180)

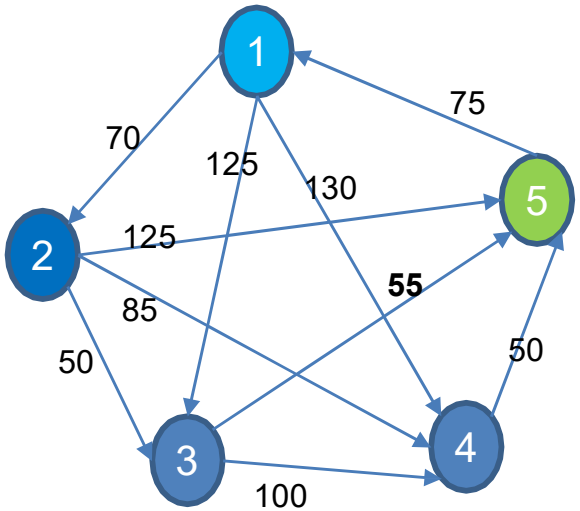
(1 2 3 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200)

(1 2 3 5 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200) (1 2 3 4 | 267)

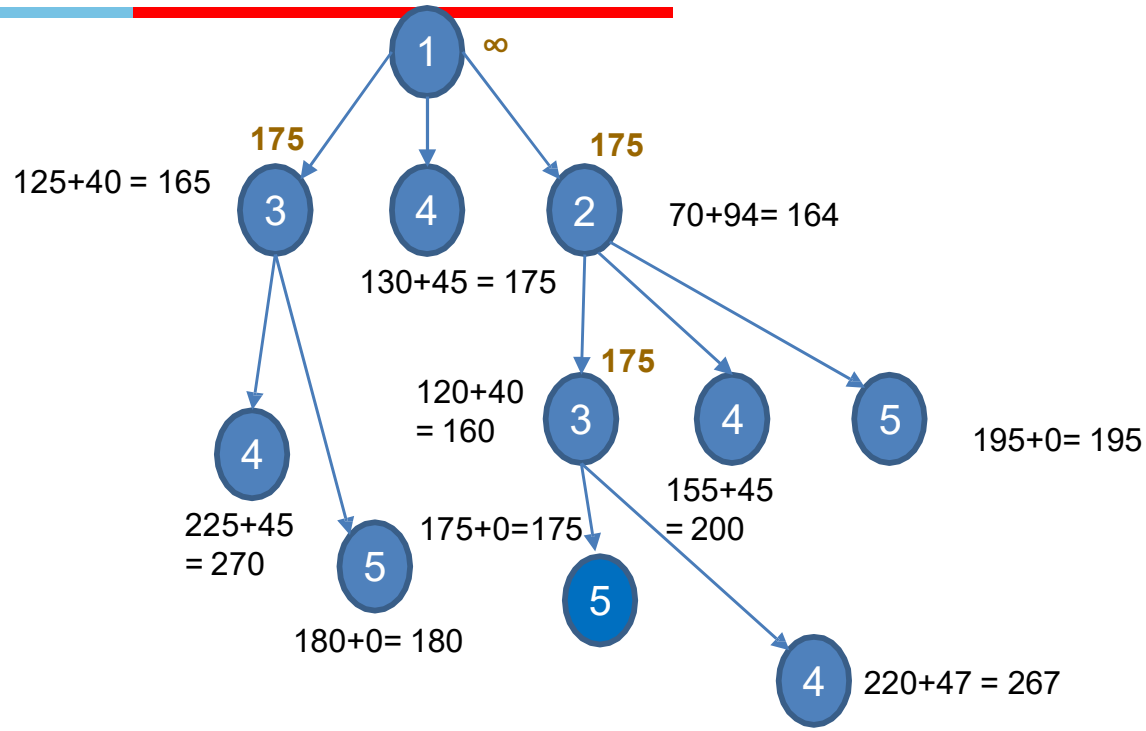
PASS

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate



n	h(n)
1	60
2	94
3	40
4	45
5	0



(1, 60)
(1 2 | 164) (1 3 | 165) (1 4 | 175)

(1 2 | 175) (1 4 | 175) (1 3 | 180)

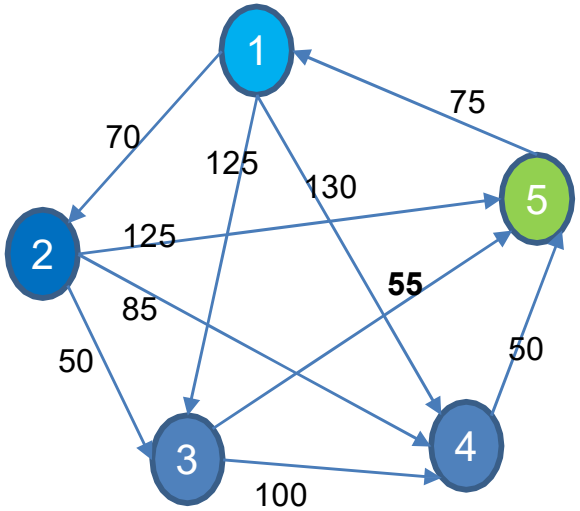
(1 2 3 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200)

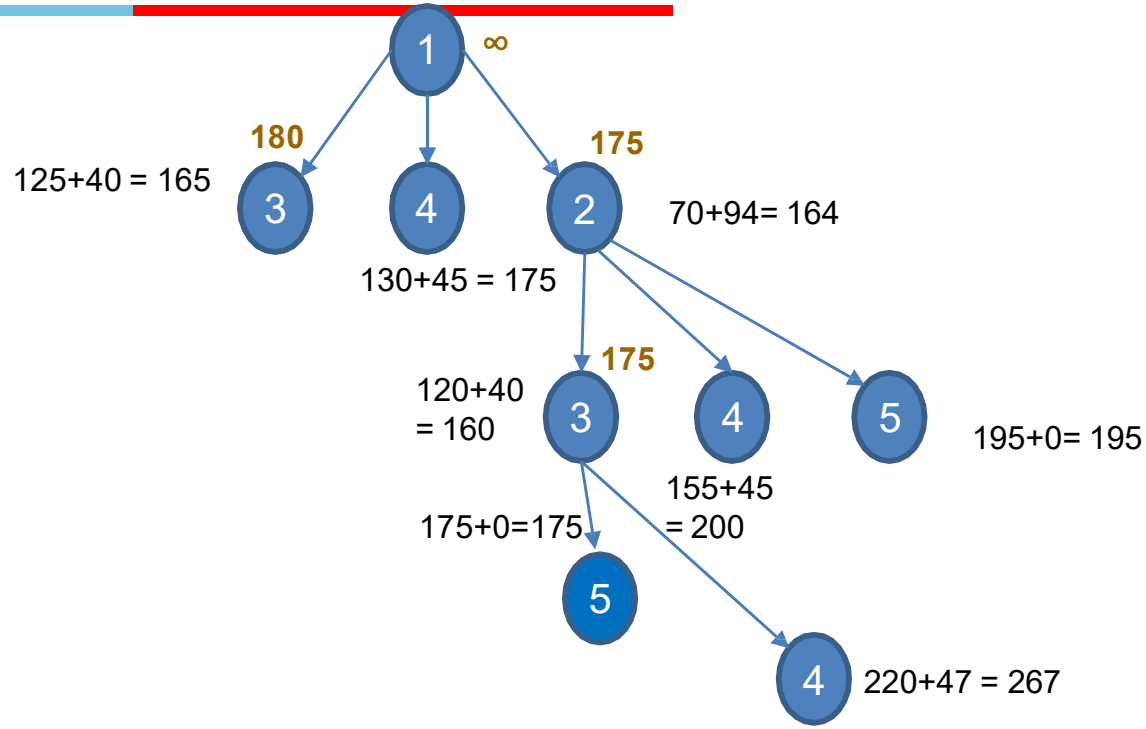
(1 2 3 5 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200) (1 2 3 4 | 267)

PASS

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate





n	h(n)
1	60
2	94
3	40
4	45
5	0

(1, 60)
(1 2 | 164) (1 3 | 165) (1 4 | 175)

(1 2 | 175) (1 4 | 175) (1 3 | 180)

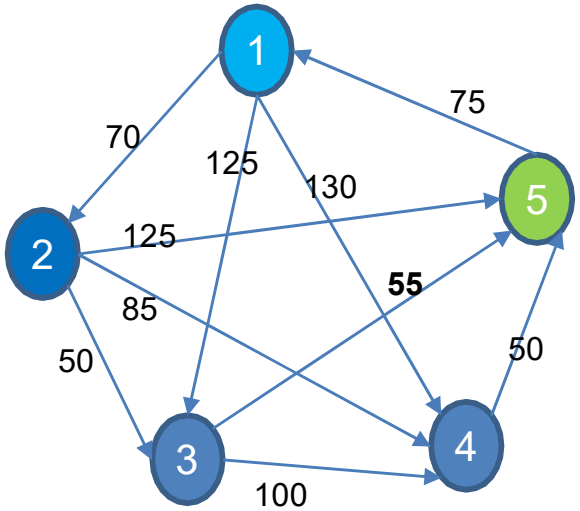
(1 2 3 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200)

(1 2 3 5 | 175) (1 4 | 175) (1 3 | 180) (1 2 5 | 195) (1 2 4 | 200) (1 2 3 4 | 267)

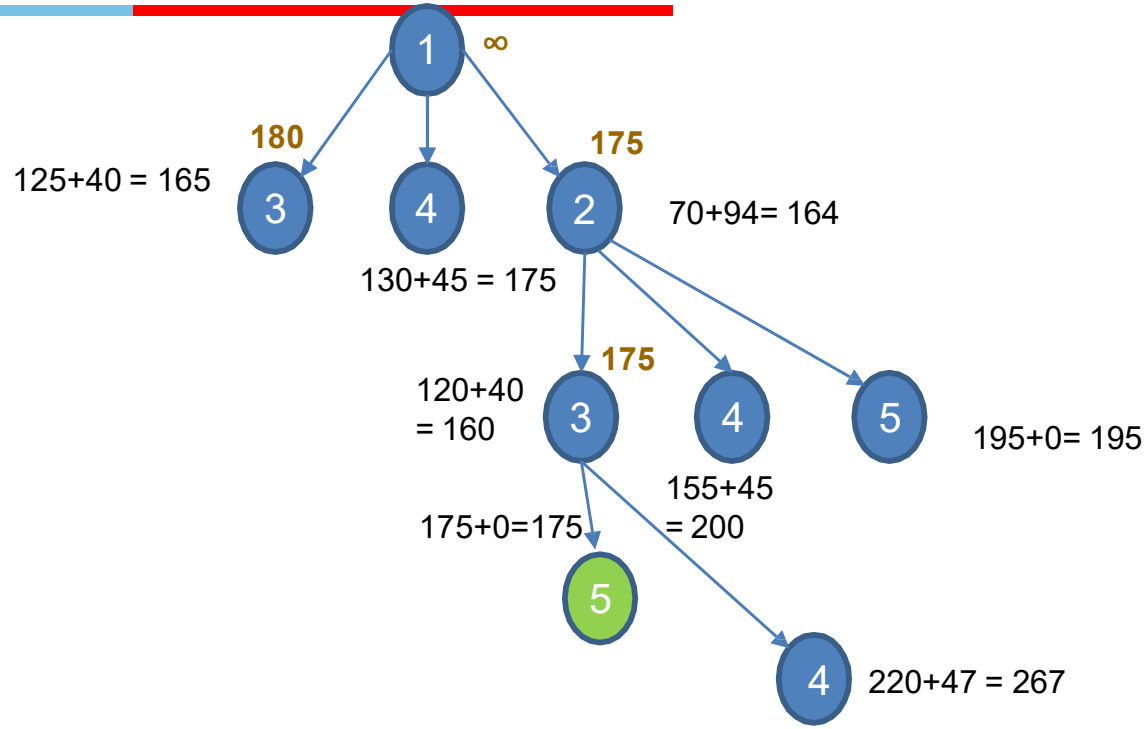
PASS

Recursive Best First Search A*

Remember the next best alternative f-Cost to regenerate



n	h(n)
1	60
2	94
3	40
4	45
5	0



If the current best leaf value > best alternative path

Best leaf value of the forgotten subtree is backed up to the ancestors

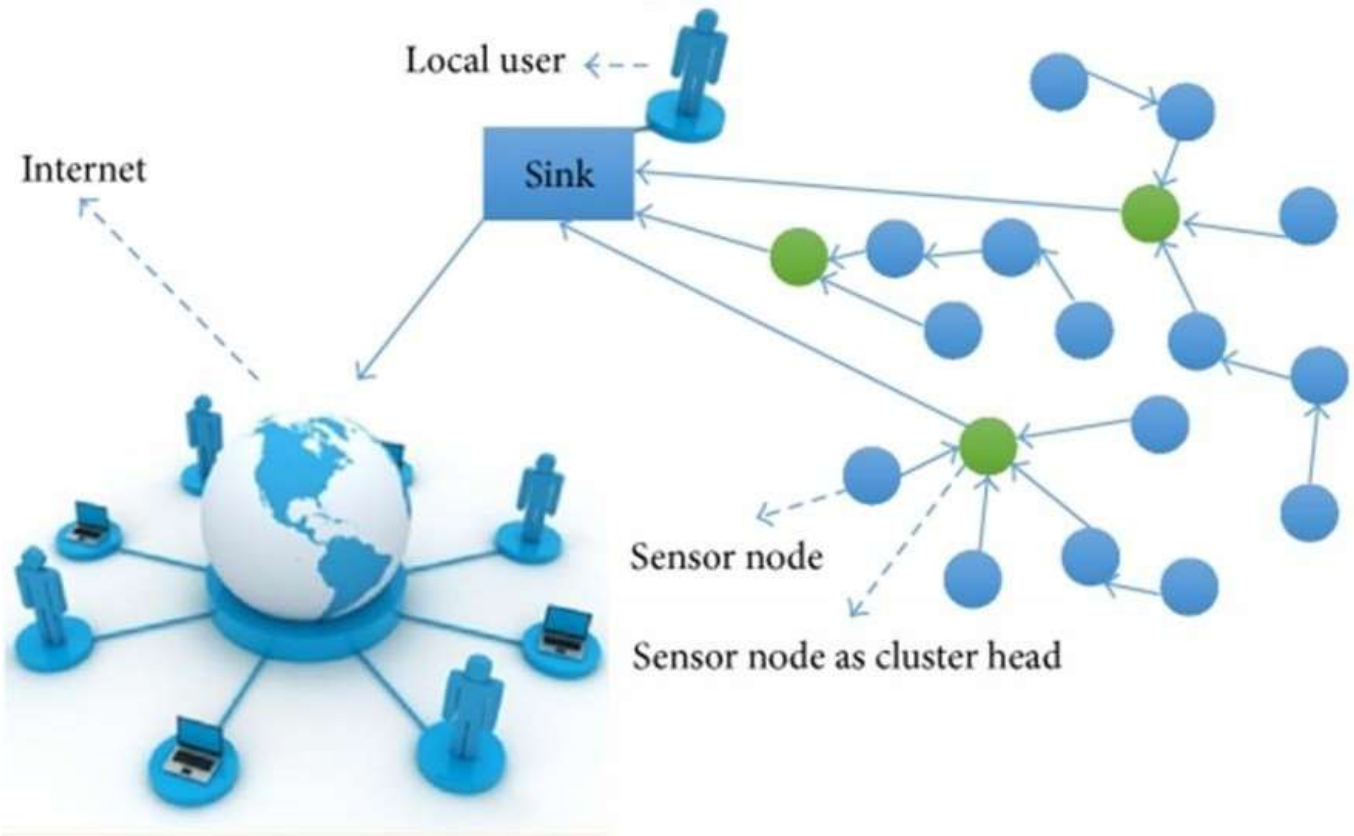
Recursion unwinds

Else

Continue expansion

Space Usage = O(bd) very less

Case Study – Search in Network Routing



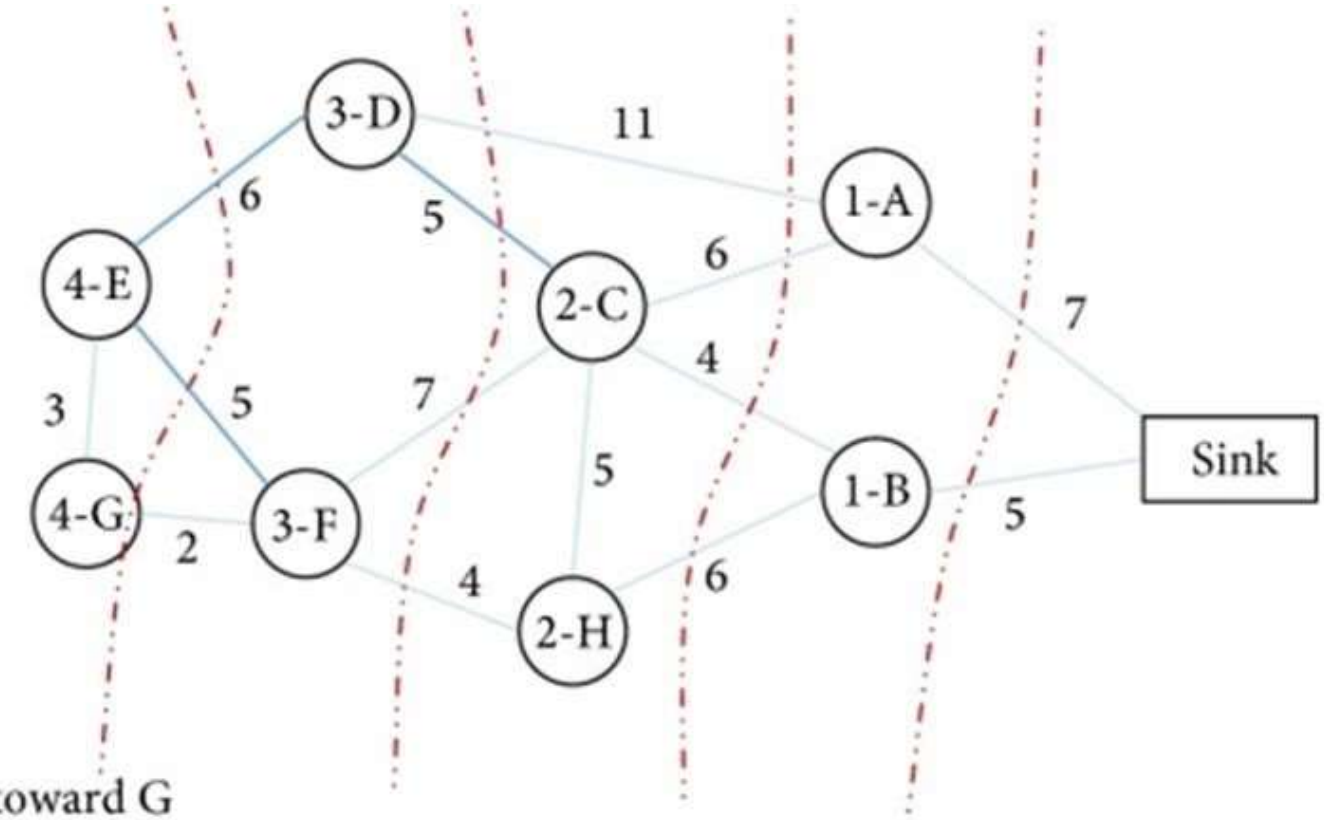
Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks
https://doi.org/10.1155/2016/8743927

Case Study – Search in Network Routing

A	14.1
B	11.3
C	8.2
H	6.6
F	2
E	3
D	4.8

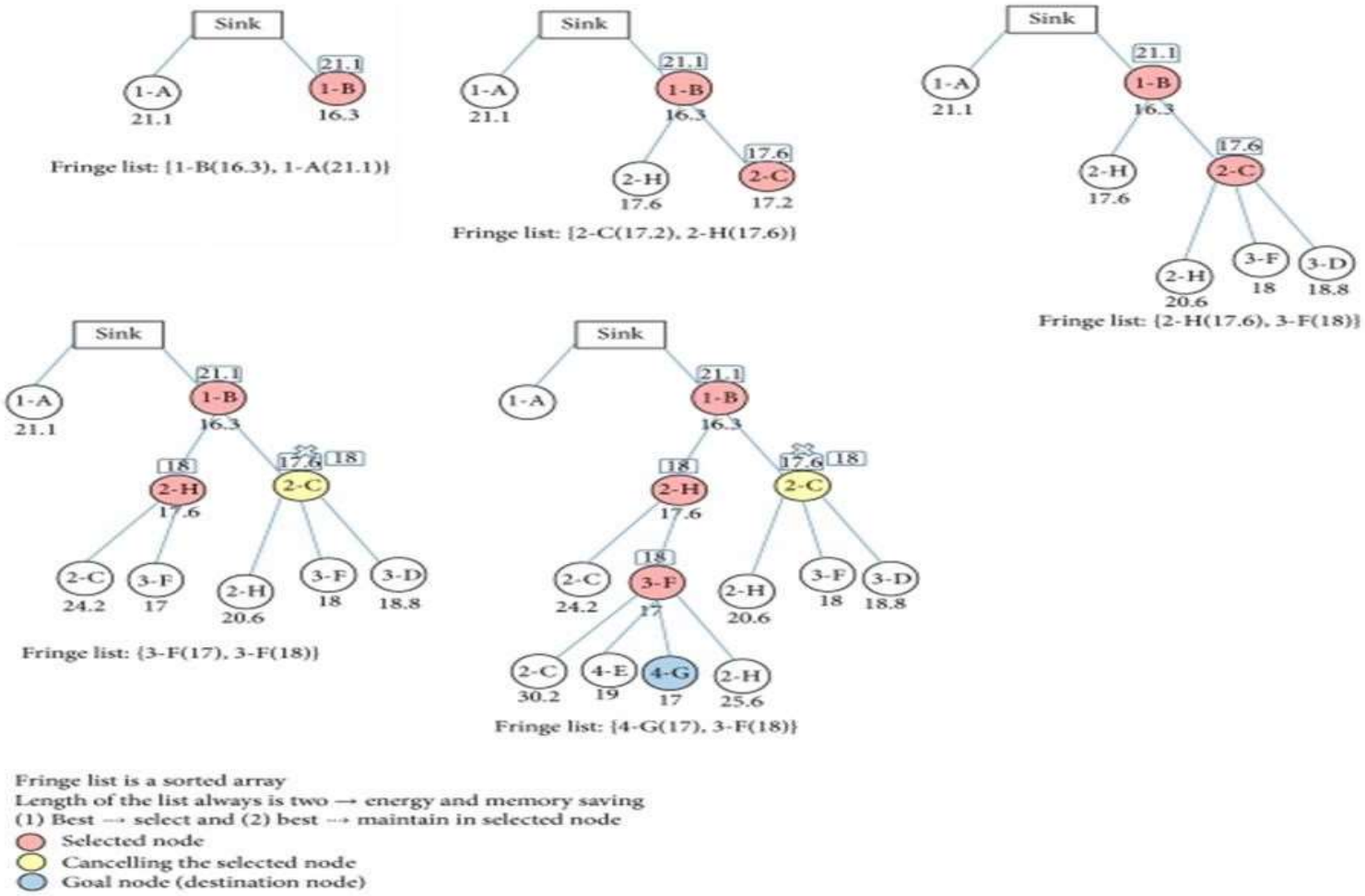
Heuristic values toward G



Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks
https://doi.org/10.1155/2016/8743927

Case Study – Search in Network Routing



Source Credit :

AR-RBFS: Aware-Routing Protocol Based on Recursive Best-First Search Algorithm for Wireless Sensor Networks

https://doi.org/10.1155/2016/8743927

Design of Heuristics

Heuristic Design

- **Effective Branching Factor**
- Good Heuristics
- Notion of Relaxed Problems
- Generating Admissible Heuristics

Effective branching factor (b^*):

If the algorithm generates N number of nodes and the solution is found at depth d, then

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

Heuristic Design

- Effective Branching Factor
- Good Heuristics
- **Notion of Relaxed Problems**
- Generating Admissible Heuristics

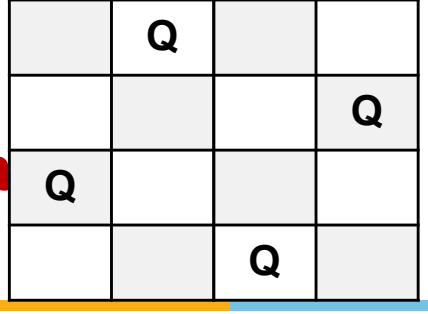
Simplify the problem

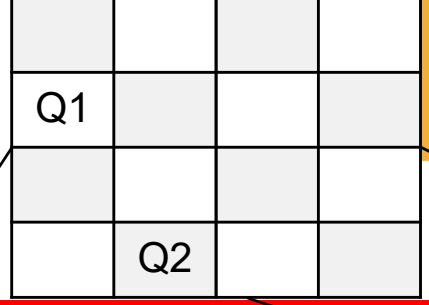
Assume no constraints

Cost of optimal solution to relaxed problem ≤ Cost of optimal solution for real problem

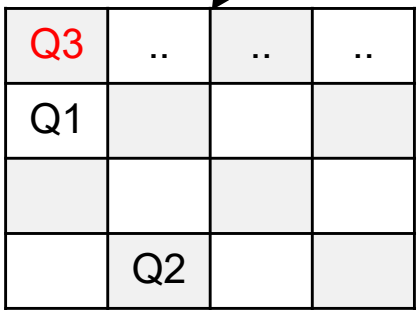
Design of Heuristics

N-Queen

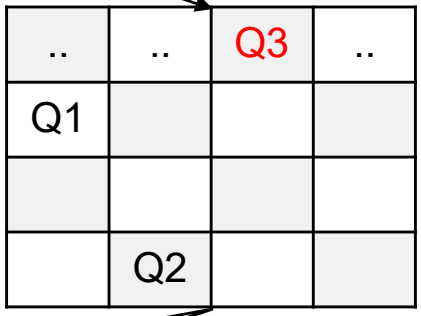




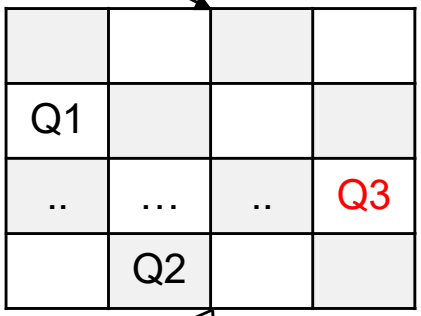
- Construct the search tree by considering one row of the board at a time
- State space graph of relaxed problem is a super graph of original state space because of removal of restrictions



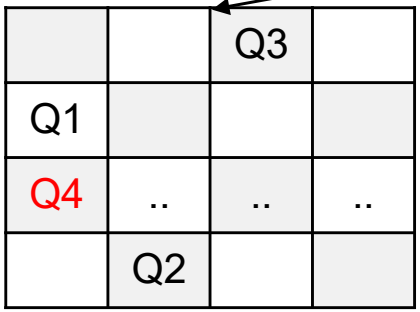
1+0+_ = 1



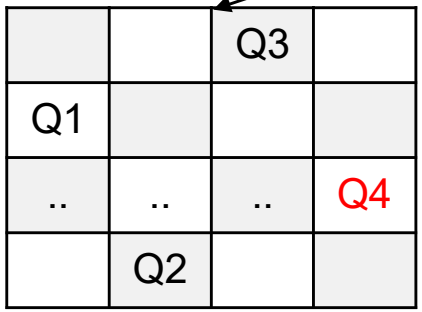
_+_+_ =



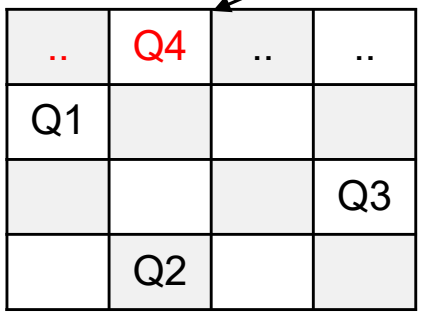
_+_+__ =



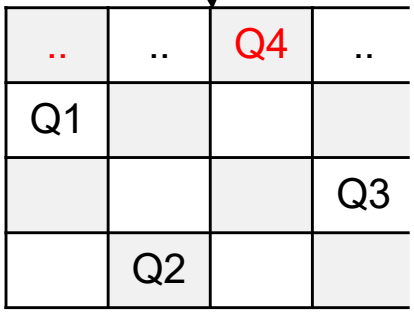
_+_+_+_=_



_+_+_+_=_



_+_+_+_=_

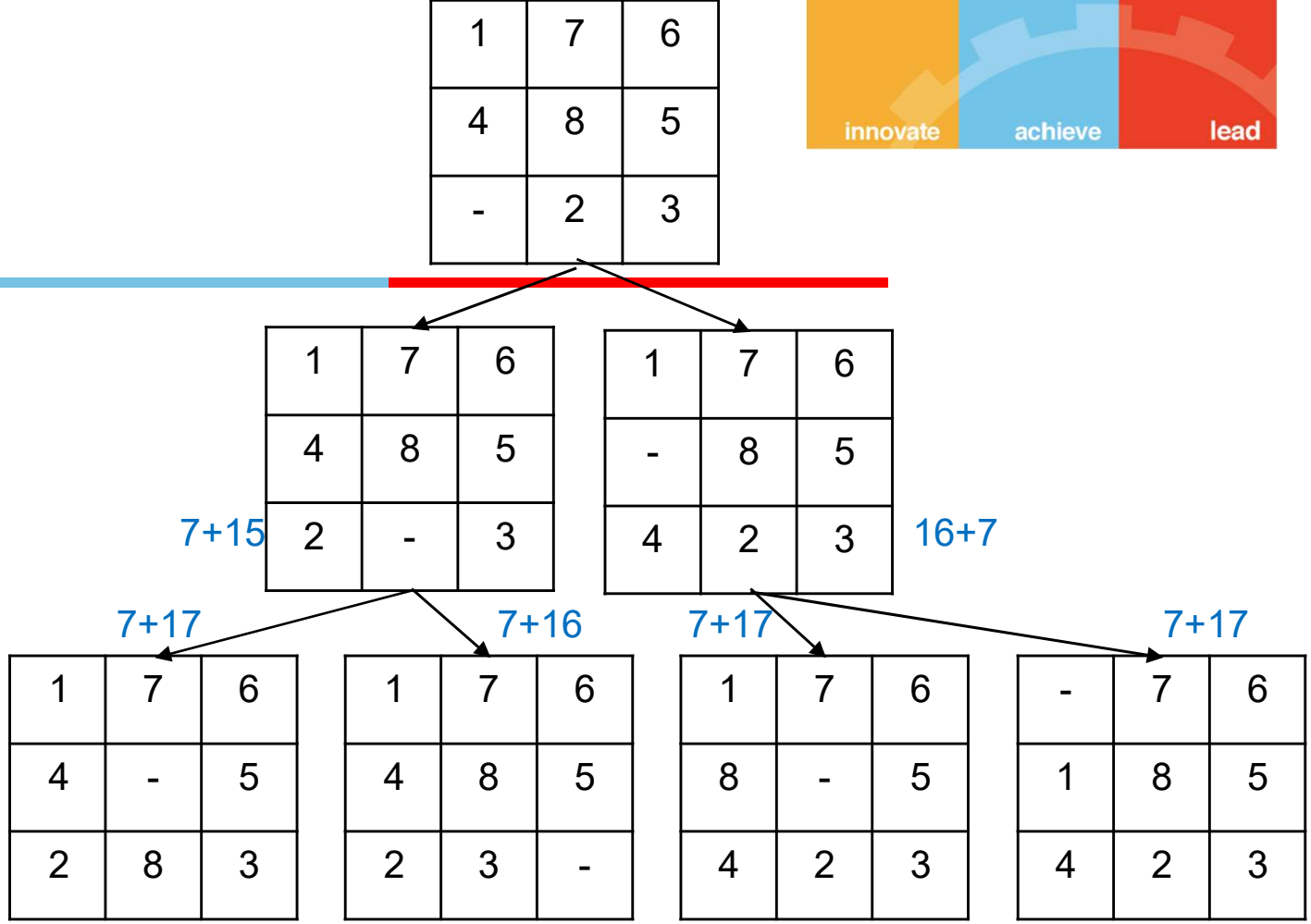


_+_+_+_=_

Initial State	Possible Actions	Transition Model	Goal Test	Path Cost	No.Of.States
< Xi , Yi >	Place in any non-occupied row in board		isValid Non-Attacking	Transition + Valid Queens	n!

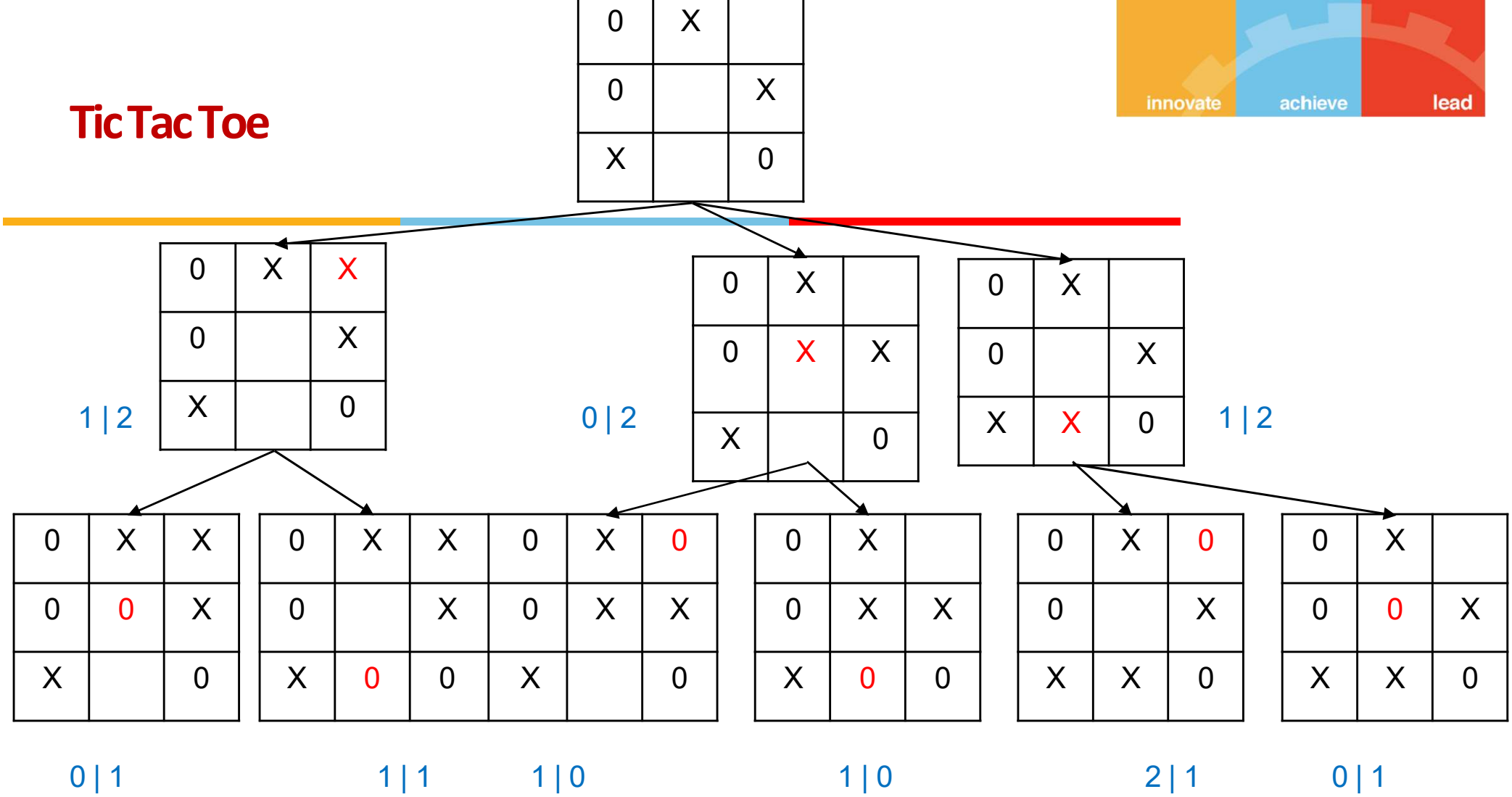
N-Tile

-	1	2
3	4	5
6	7	8



Initial State	Possible Actions	Transition Model	Goal Test	Path Cost	No.Of.States
<LOC, ID>	Move Empty to near by Tile		ID=LOC+1	Transition + Positional + Distance+ Other approaches	9!

Tic Tac Toe



Opposite Win | Player Win

Initial State	Possible Actions	Transition Model	Goal Test	Path Cost	No.Of.States
([Xij], [Yij])	Place a coin in unoccupied (i,j)		N : i's N : j's N : i=j	No.of.Steps + Opp.Win + (N-1-Curr.Win)	$19{,}683=3^9$

Learn from experience

Trail / Puzzle	X1(n) : No.of.Misplaced Tiles	X2(n): Pair of adjacent tiles that are not in goal	X3(n): Position of the empty tile	……h`(n)
Example 1	7	10	7	……
Example 2	5	6	6	……
……	..	..	..	……

-	1	2
3	4	5
6	7	8

1	7	6
4	8	5
-	2	3

Create a suitable model:
h(n) = c1*X1(n) + c2*X2(n) + ……..

Required Reading: AIMA - Chapter #3: 3.5, 3.6

Next Class Plan :
Local Search Algorithms &
Optimization Problems

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CLZG557

M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

Learning Objective

At the end of this class , students should be able to:

1. Differentiate which local search is best suitable for given problem
2. Design fitness function for a problem
3. Construct a search tree
4. Apply appropriate local search and show the working of algorithm at least for first 2 iterations with atleast four next level successor generation(if search tree is large)
5. Design and show local search Algorithm steps for a given problem

Local Search & Optimization

Task Environment

Goal Formulation

Problem Formulation

Search Phase

Execution Phase

Phases of Solution Search by PSA

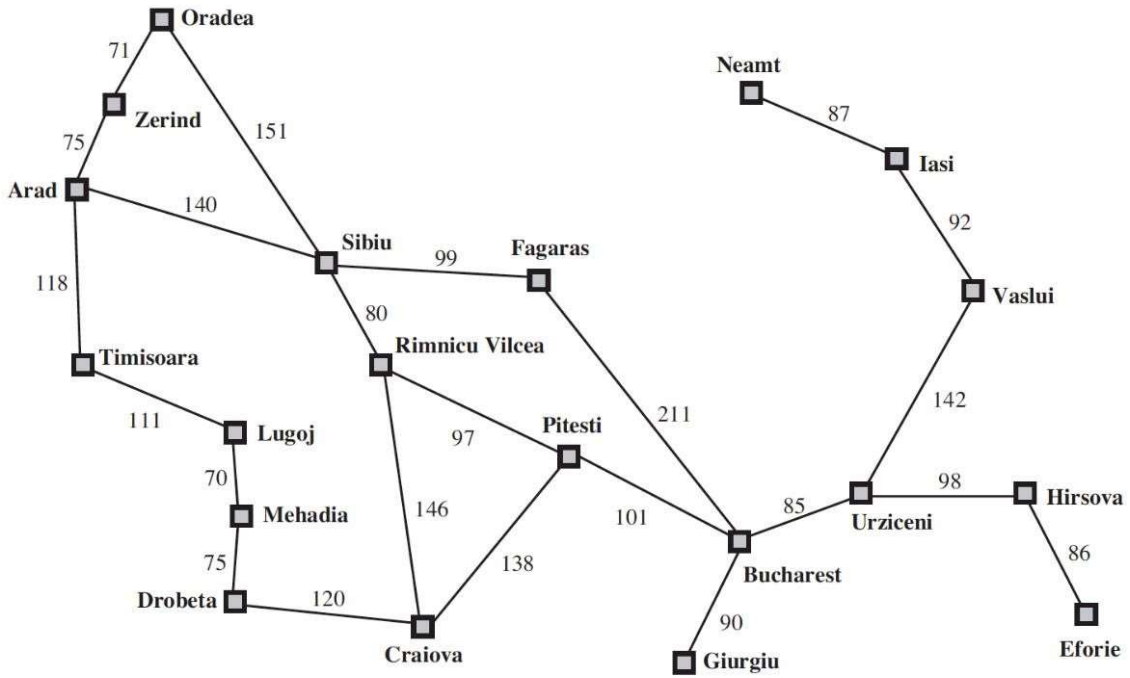
Assumptions – Environment :

Static

Observable

Discrete

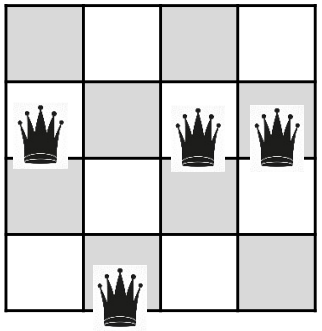
Deterministic

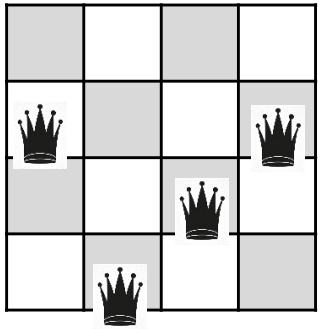


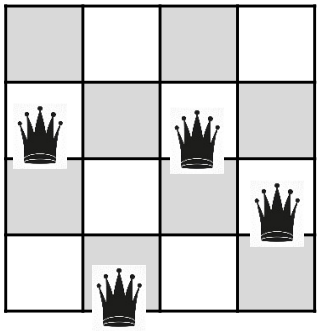
Local Search

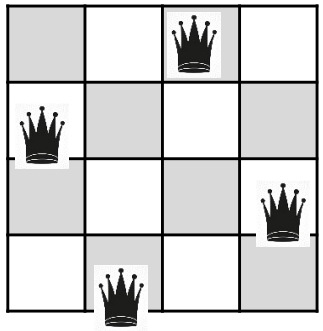
Terminology

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found









Feasible State/Solution **Neighboring States** **Optimal Solution**

Fitness Value:

h(n) = 4 h(n) = 4 h(n) = 2 h(n) = 0

h(n) = No.of.Conflicting **pairs** of queens

h(n) = 2 h(n) = 2 h(n) = 4 h(n) = 6

h(n) = No.of.**Non**-Conflicting **pairs** of queens.

Local Search

Terminology

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

Algorithms:

- Choice of Neighbour
- Looping Condition
- Termination Condition

2	5	3	2
	6		
3	5	4	2
4		4	2

Local Search

Optimization Problem

Goal : Navigate through a state space for a given problem such that an optimal solution can be found

Objective : Minimize or Maximize the objective evaluation function value

Scope : Local

Objective Function : Fitness Value evaluates the goodness of current solution

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

Single Instance Based	Multiple Instance Based
Hill Climbing	Genetic Algorithm
Simulated Annealing	Particle Swarm Optimization
Local Beam Search	Ant Colony Optimization
Tabu Search	

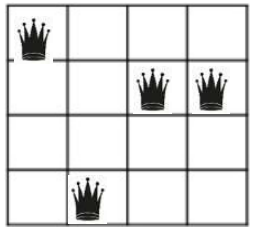
Hill Climbing

Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

h(n) = No.of non-conflicting pairs of queens in the board.

Q1-Q2
Q1-Q3
Q1-Q4
Q2-Q3
Q2-Q4
Q3-Q4



1	4	2	2	4

Note : Steps 3 & 4 in the above algorithm will be a part of variation of Hill climbing

Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

h(n) = No.of non-conflicting pairs of queens in the board.

Q1-Q2

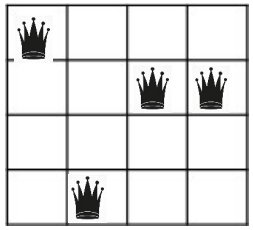
Q1-Q3

Q1-Q4

Q2-Q3

Q2-Q4

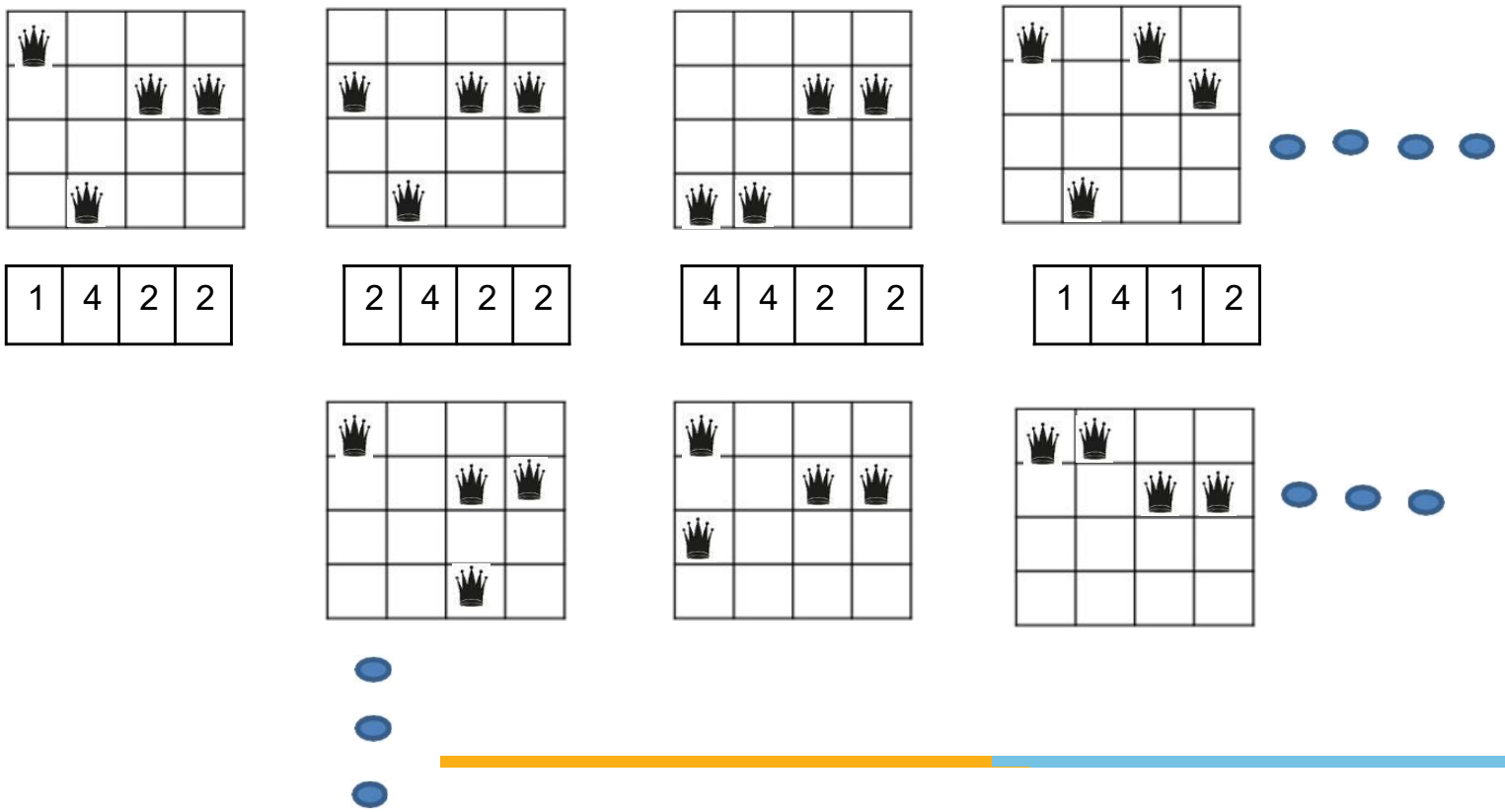
Q3-Q4



1	4	2	2	4

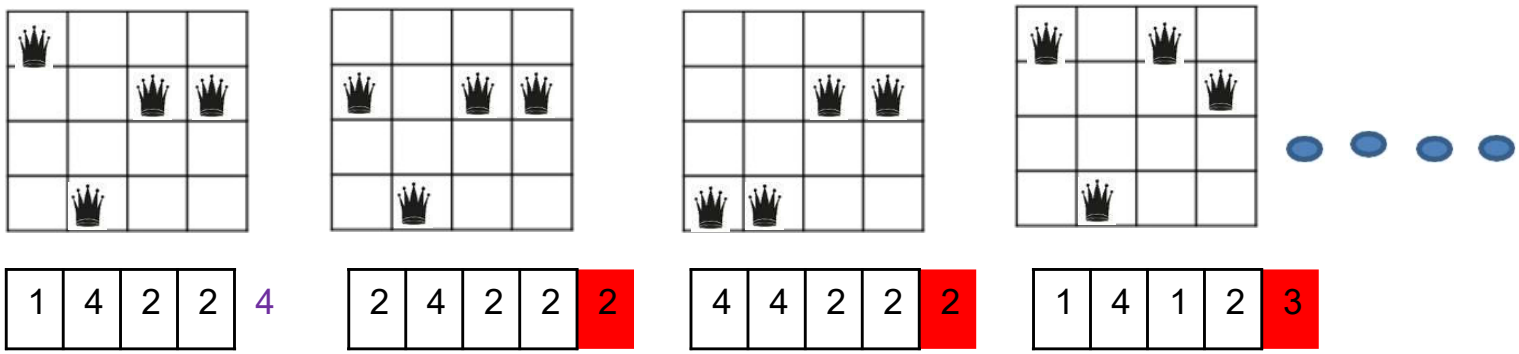
Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

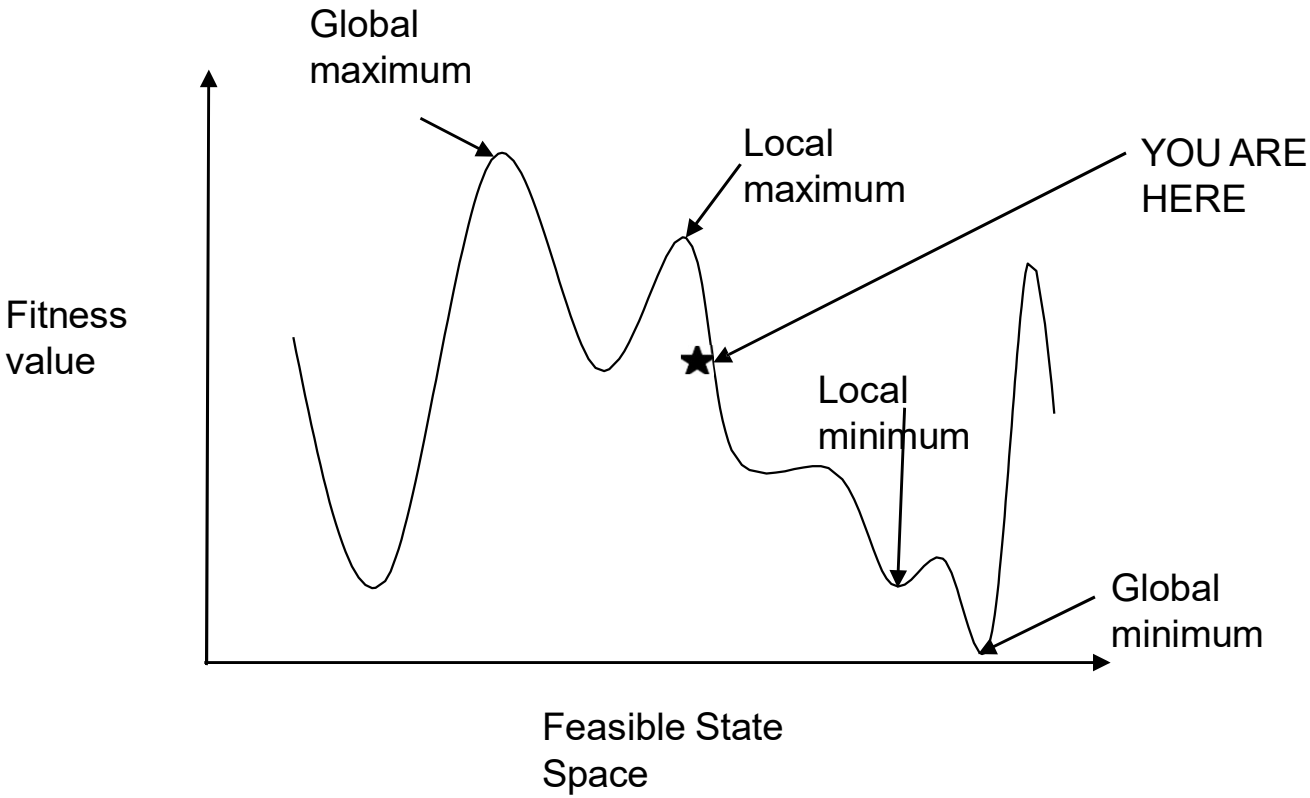


Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



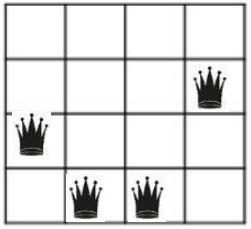
Hill Climbing



Hill Climbing

Random Restart

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

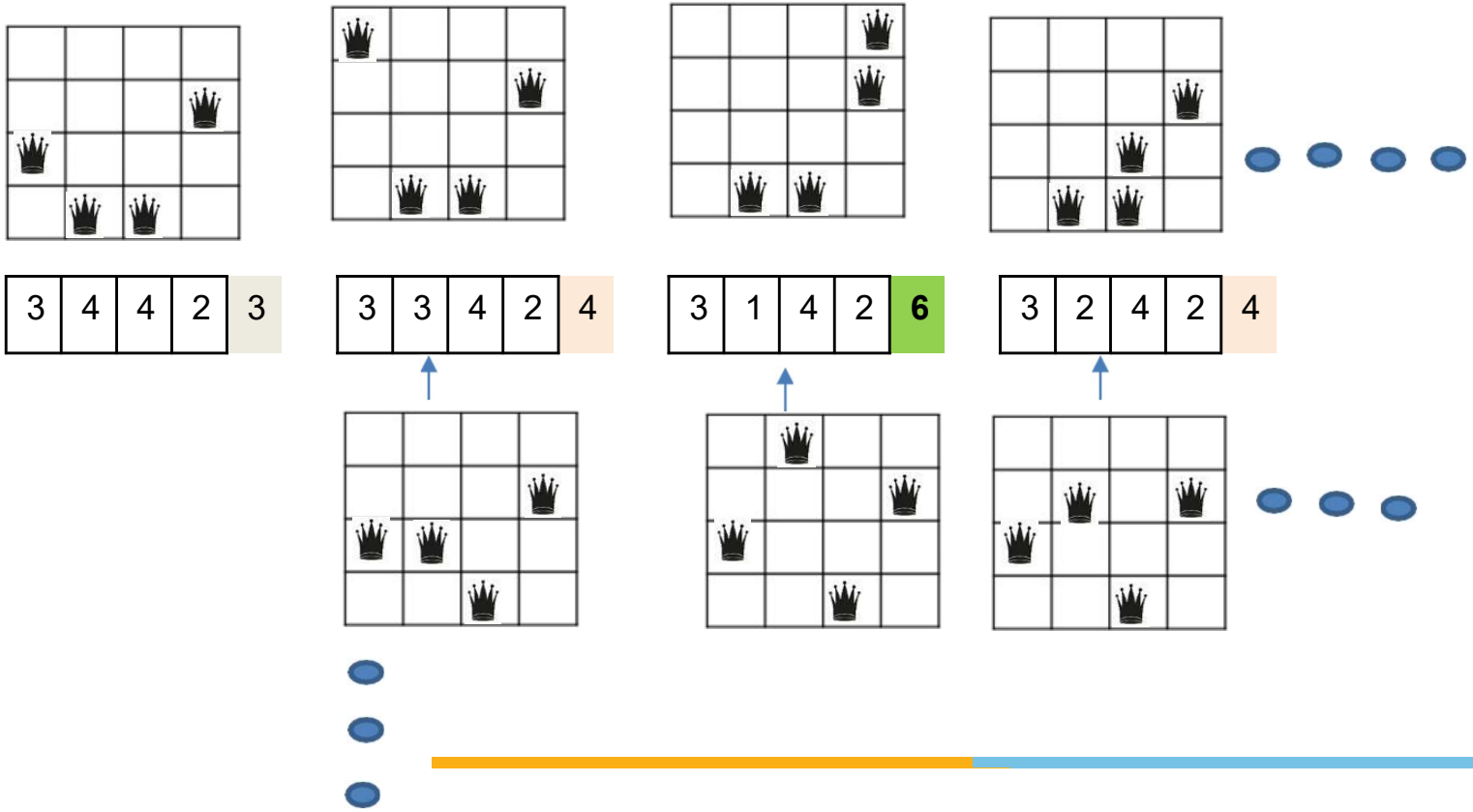


3	4	4	2	3

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
     neighbor ← a highest-valued successor of current
     if neighbor.VALUE ≤ current.VALUE then return current.STATE
     current ← neighbor
```

Hill Climbing

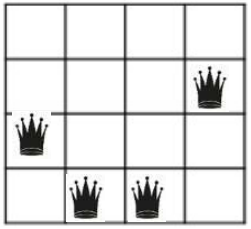
1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



Hill Climbing

Random Restart

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

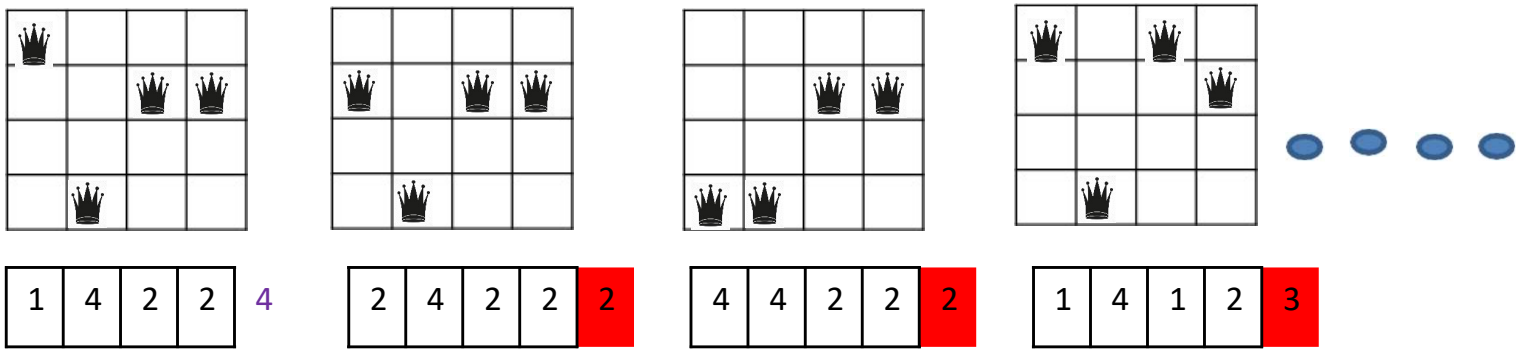


3	4	4	2	3

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
     neighbor ← a highest-valued successor of current
     if neighbor.VALUE ≤ current.VALUE then return current.STATE
     current ← neighbor
```

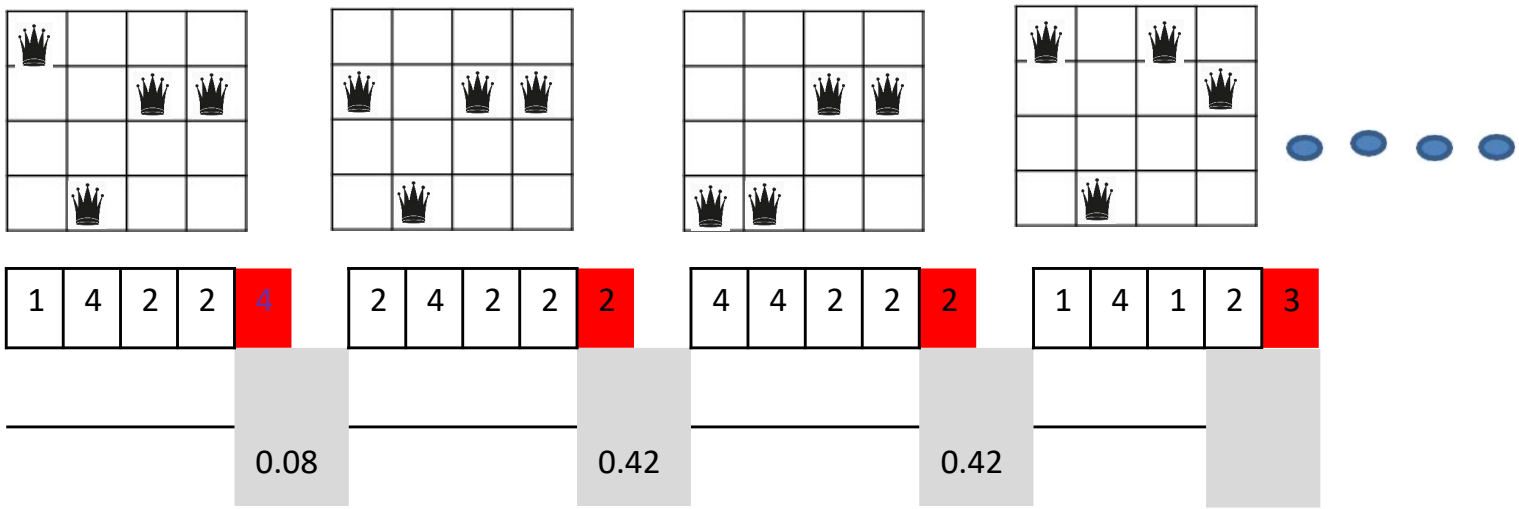
Stochastic Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



Stochastic Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



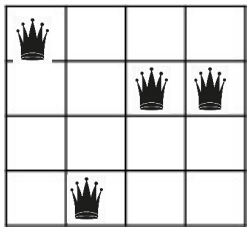
12 N = {4,2,2,3,3,2,2,0,2,1,3,0}

$next \leftarrow$ a randomly selected successor of $current$
$\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
if $\Delta E > 0$ **then** $current \leftarrow next$
else $current \leftarrow next$ only with probability $e^{\Delta E/T}$

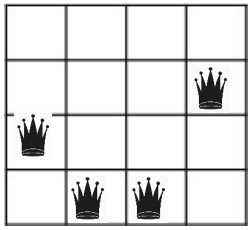
Local Beam Search

Beam Search

1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probabilit
6. Repeat from Step 2



1	4	2	2	4

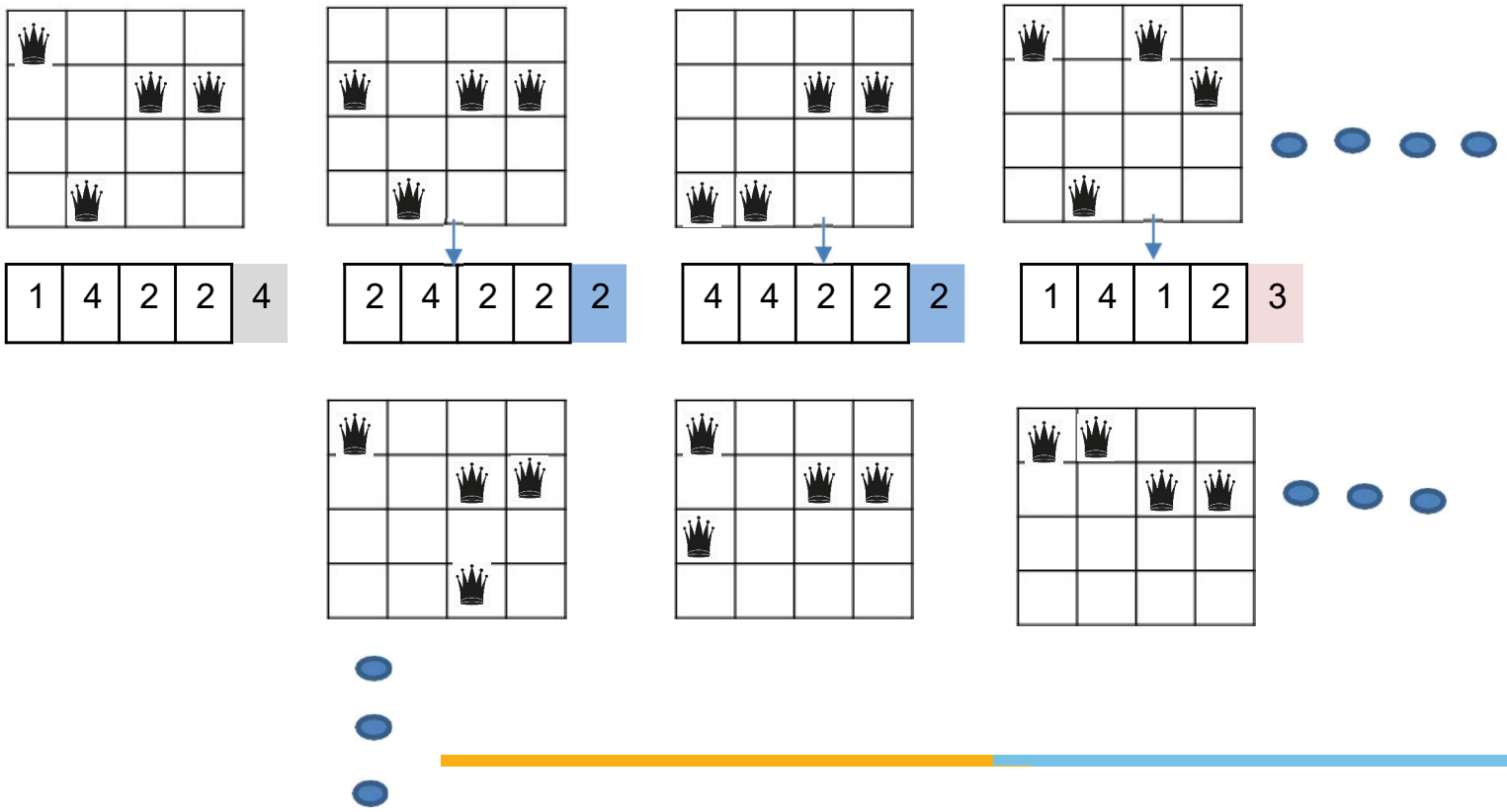


3	4	4	2	3

Beam Search

1st State

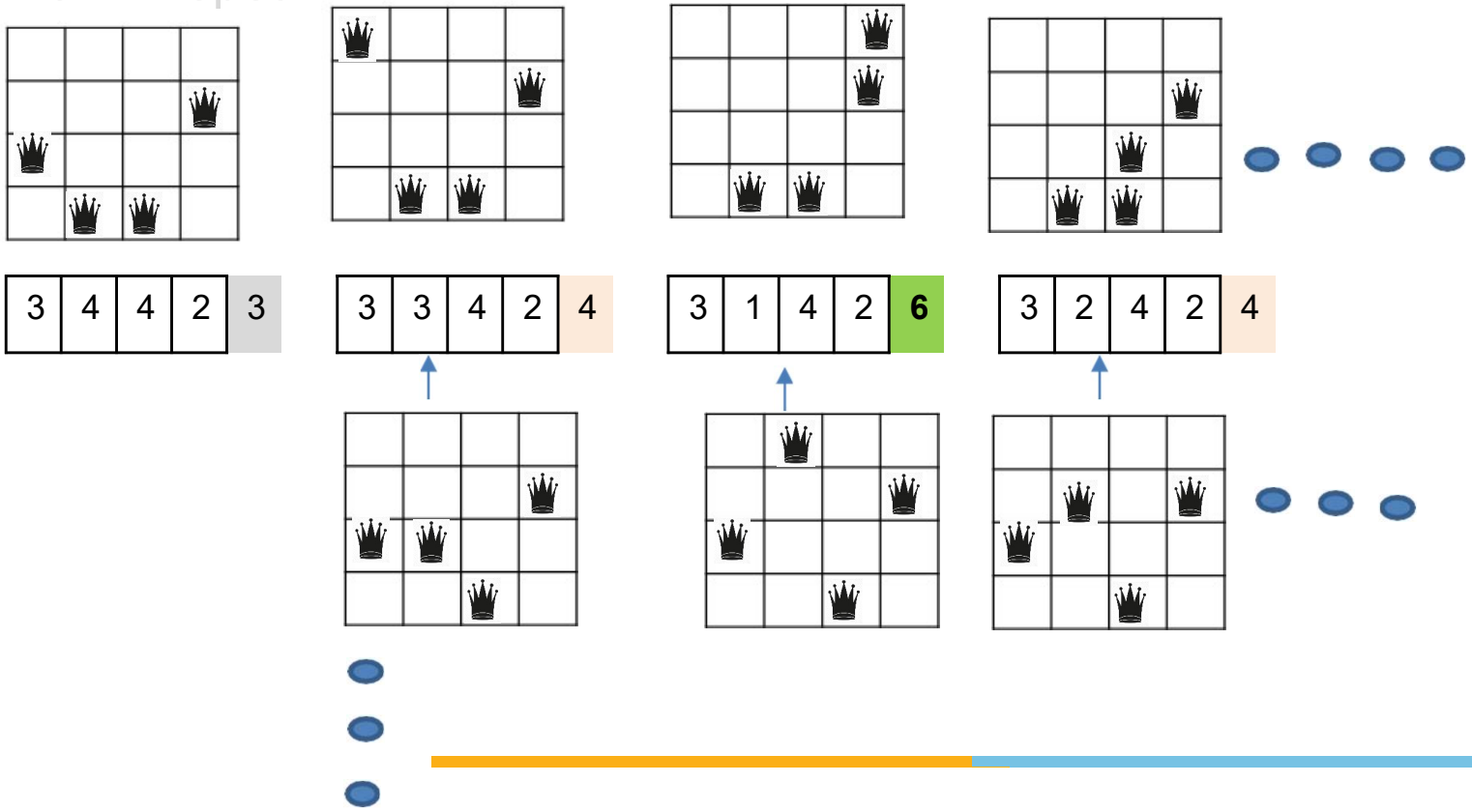
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2



Beam Search

2nd State

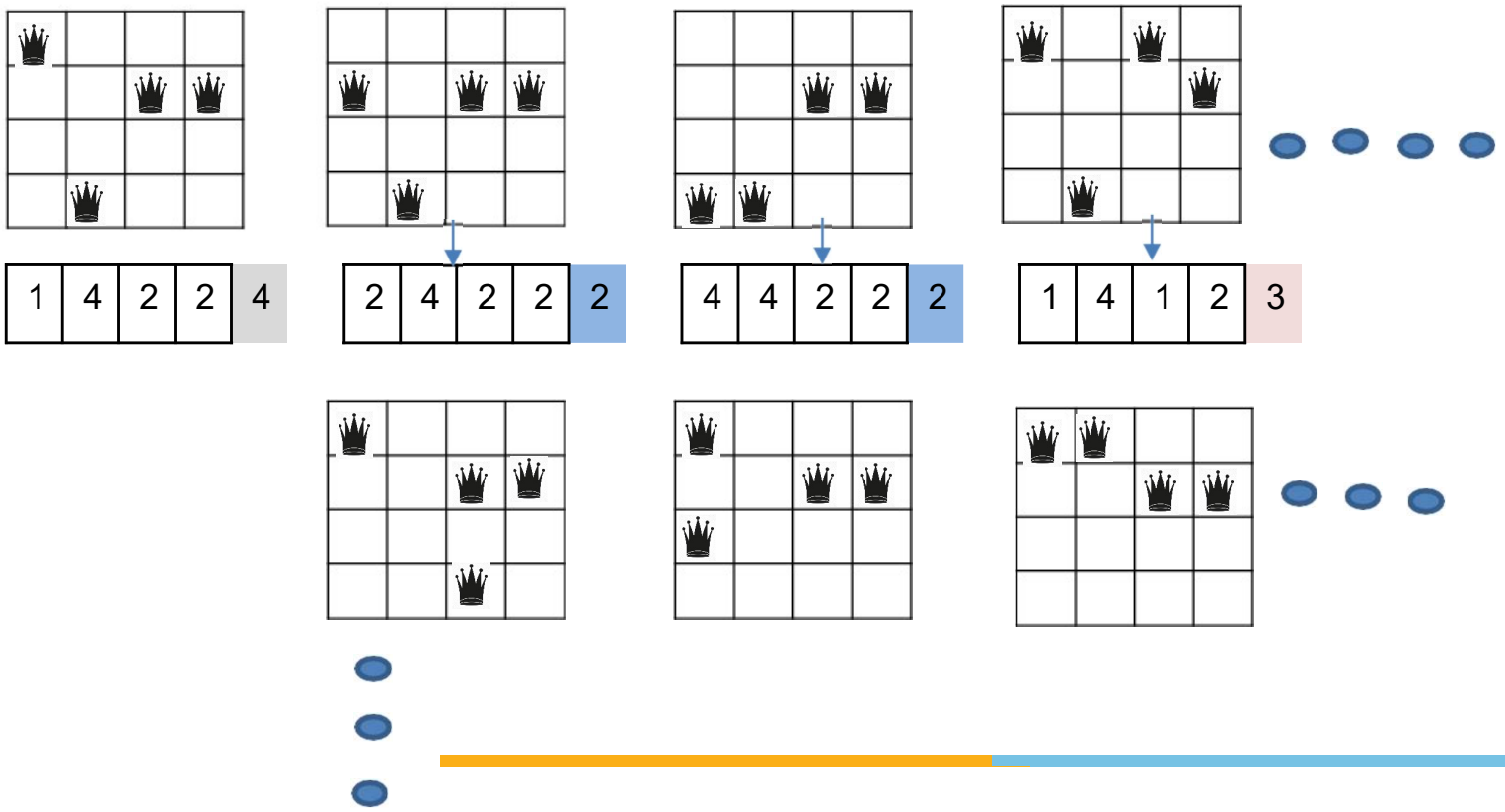
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2



Stochastic Beam Search

Sample from 1st State

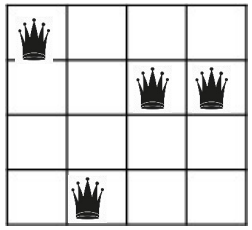
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next 'k' states randomly based on the probability
6. Repeat from Step 2



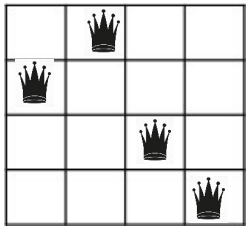
Genetic Algorithm

Genetic Algorithm

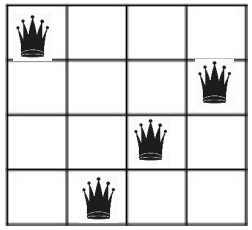
1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



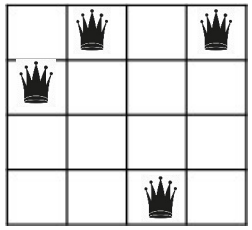
1	4	2	2



2	1	3	4



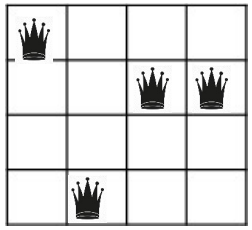
1	4	3	2



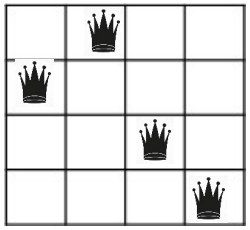
2	1	4	1

Genetic Algorithm

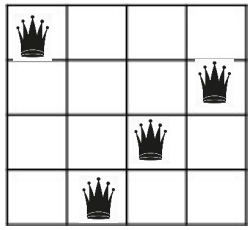
1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



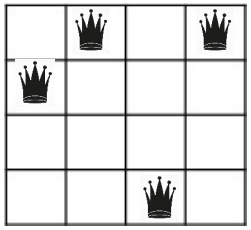
1	4	2	2	4



2	1	3	4	4



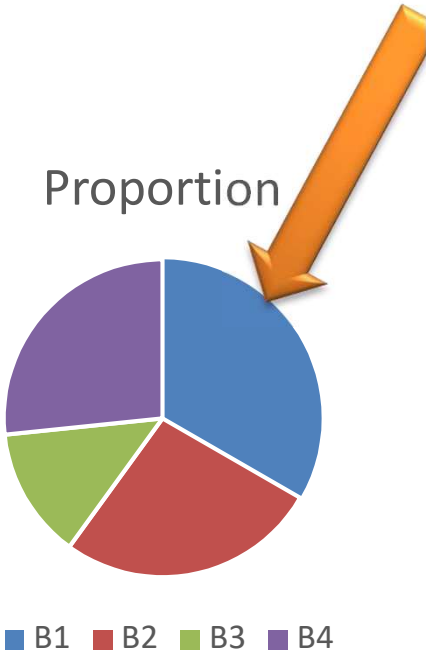
1	4	3	2	2

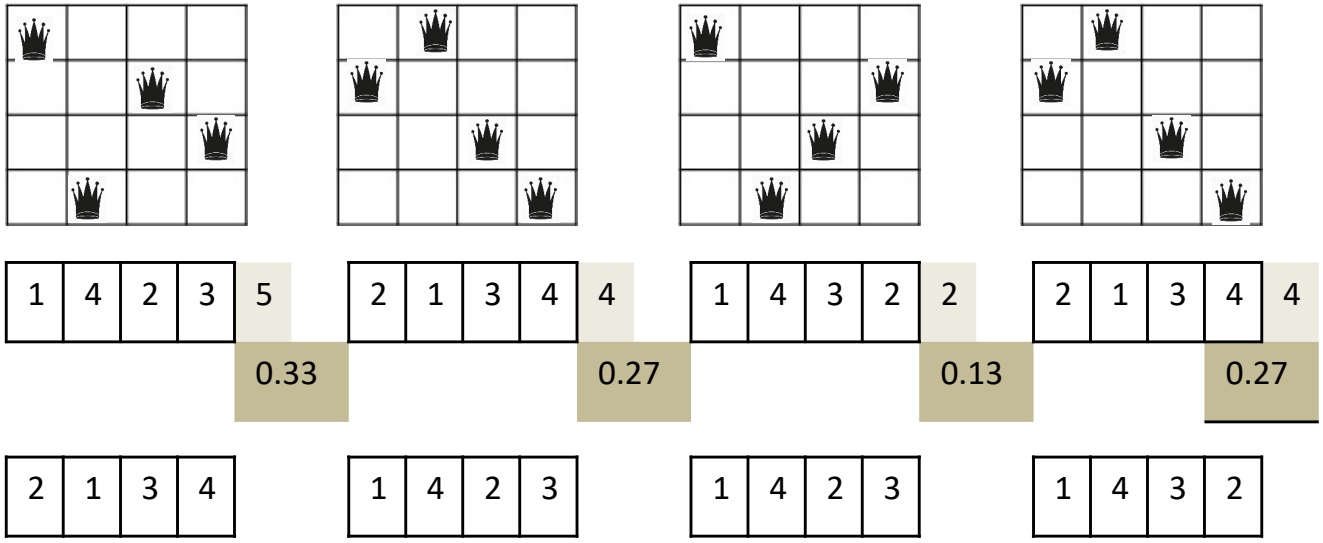


2	1	4	1	3

Genetic Algorithm

Eg., use roulette wheel mechanism to select pair/s

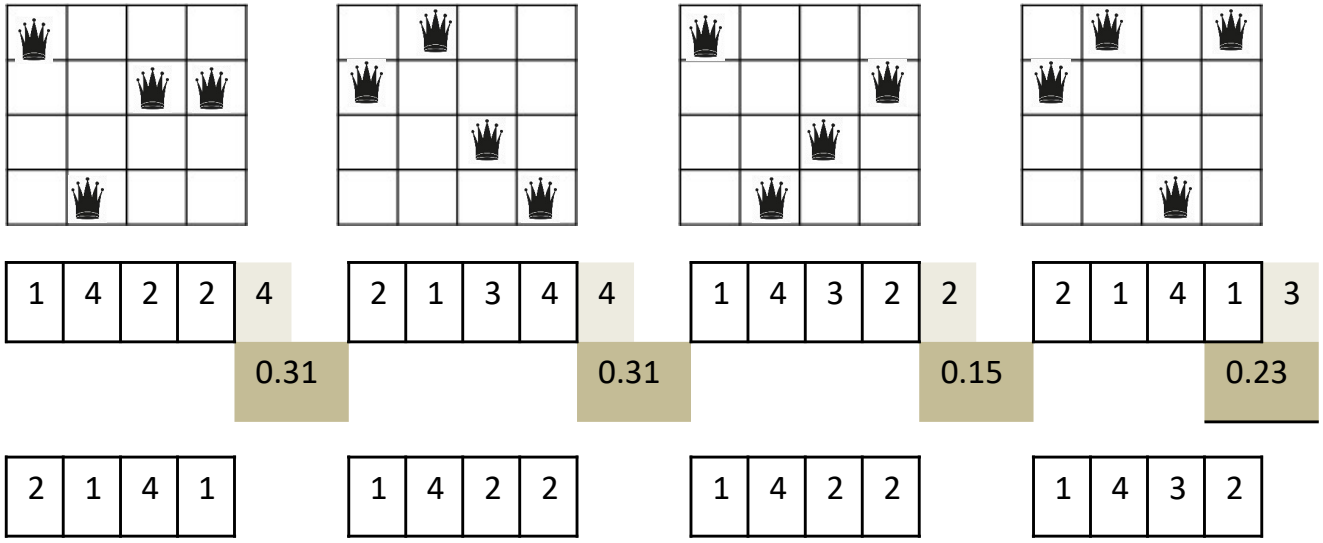




Sample winners of game -1 ,2,3,4 : B4, B1, B1, B3

Genetic Algorithm

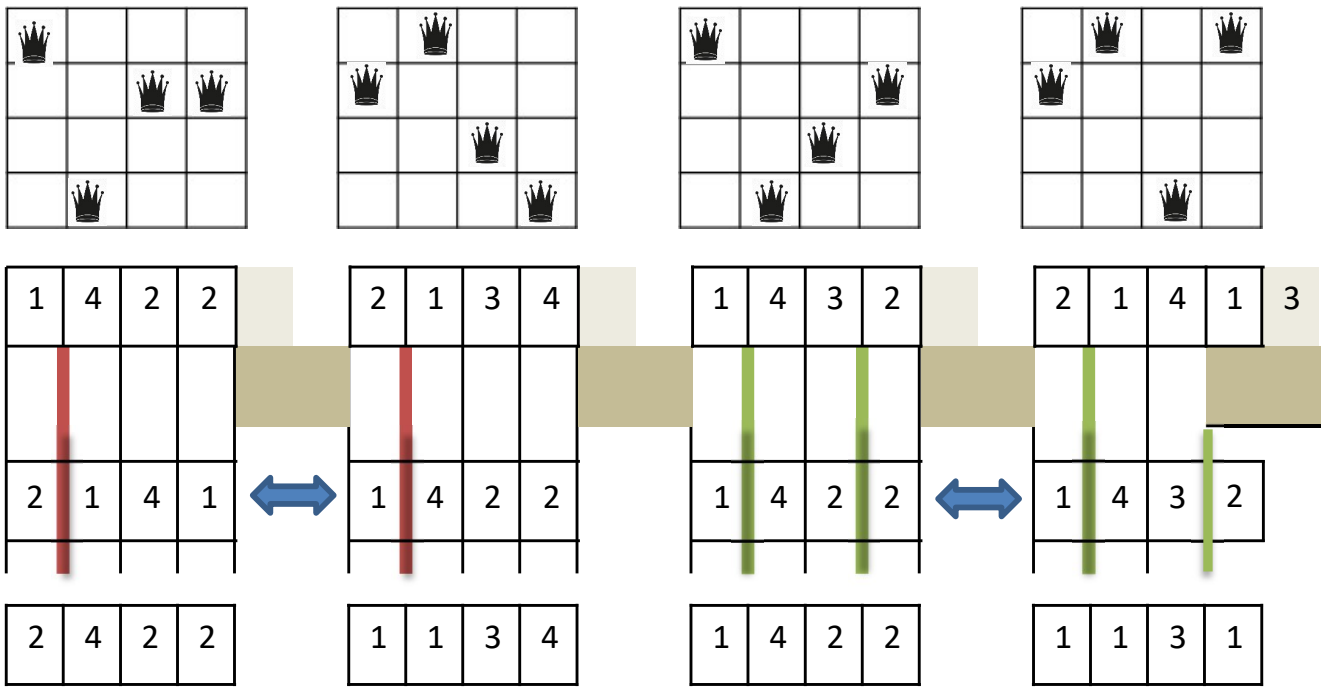
1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Sample winners of game -1 ,2,3,4 : B4, B1, B1, B3

Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Genetic Algorithm – Reference to alternative approaches of crossover

2	4	2	2

1	1	3	4

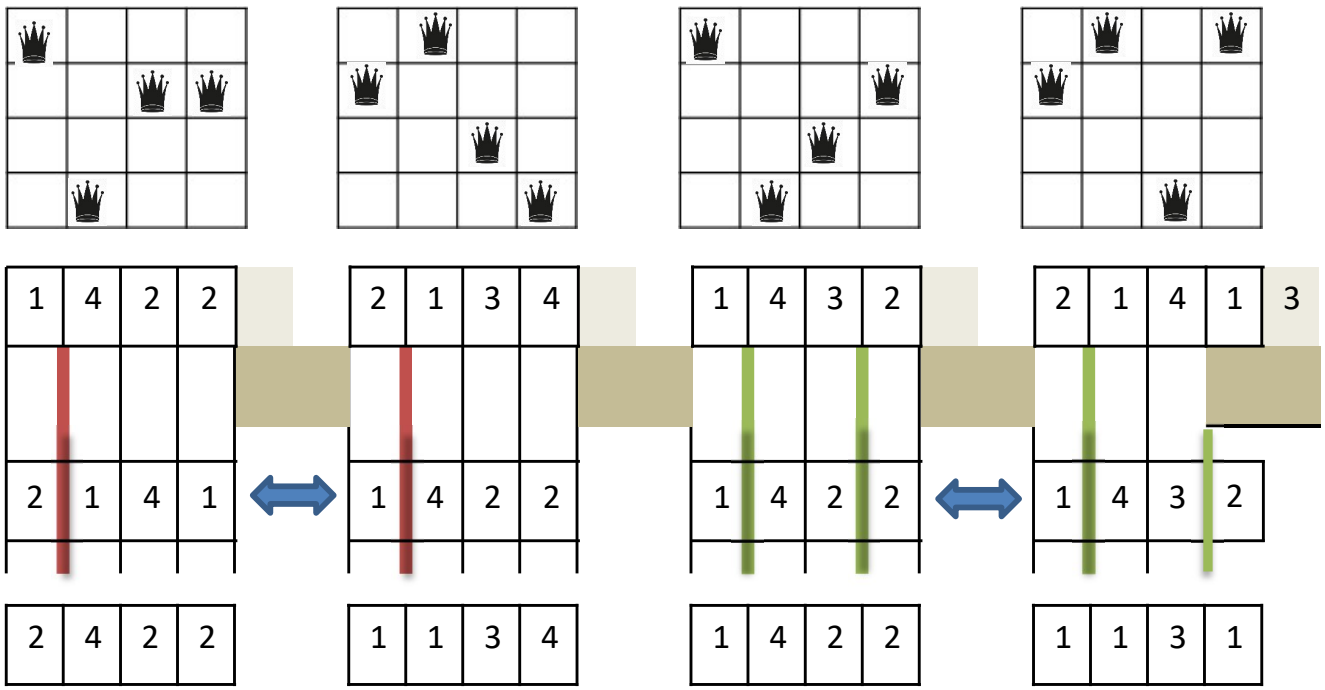
1	4	2	2

1	1	3	1

http://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf

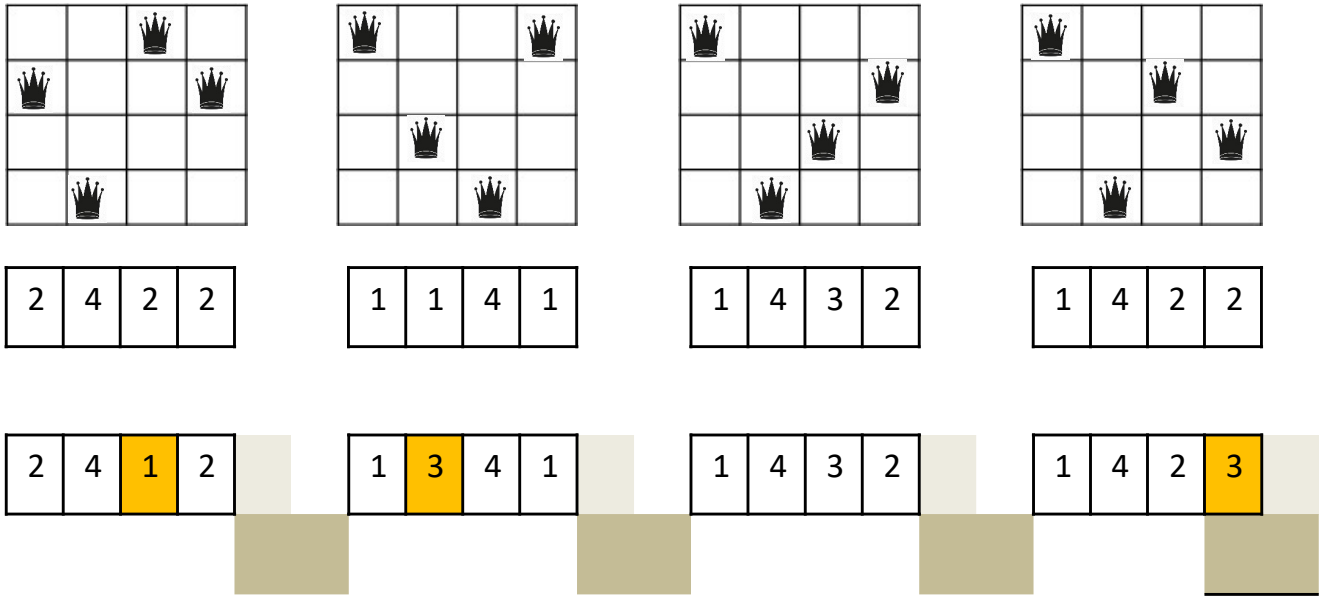
Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



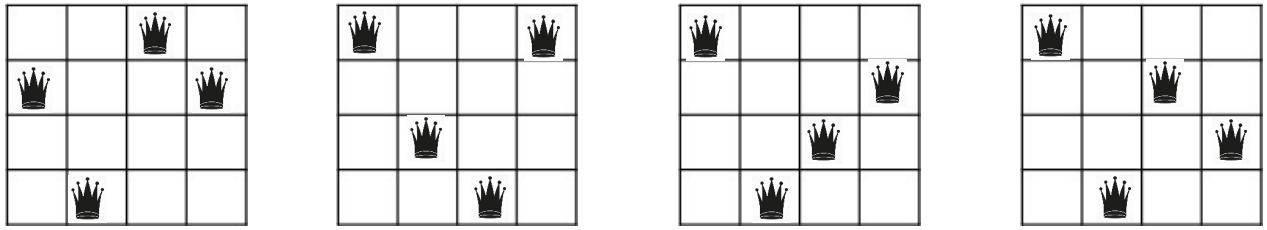
Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



2	4	2	2

1	1	4	1

1	4	3	2

1	4	2	2

2	4	1	2	3
				0.23

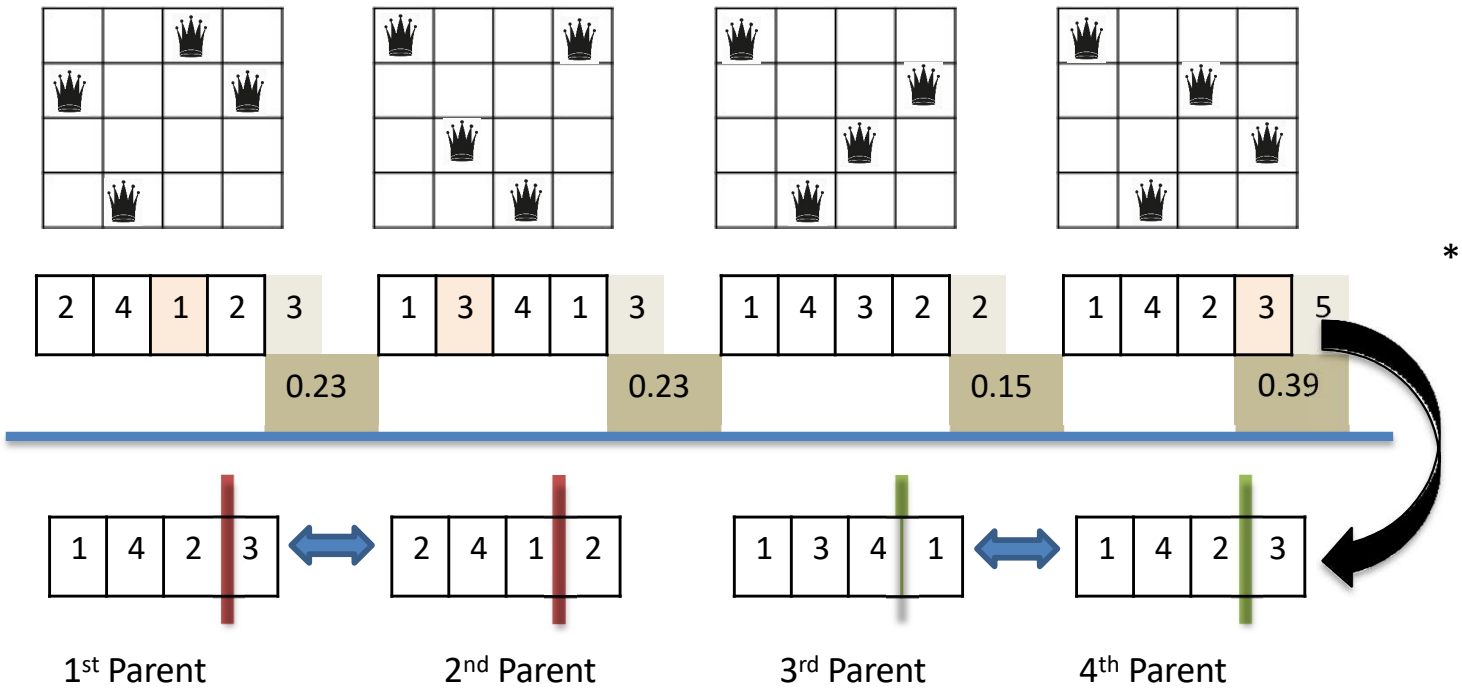
1	3	4	1	3
				0.23

1	4	3	2	2
				0.15

1	4	2	3	5
				0.39

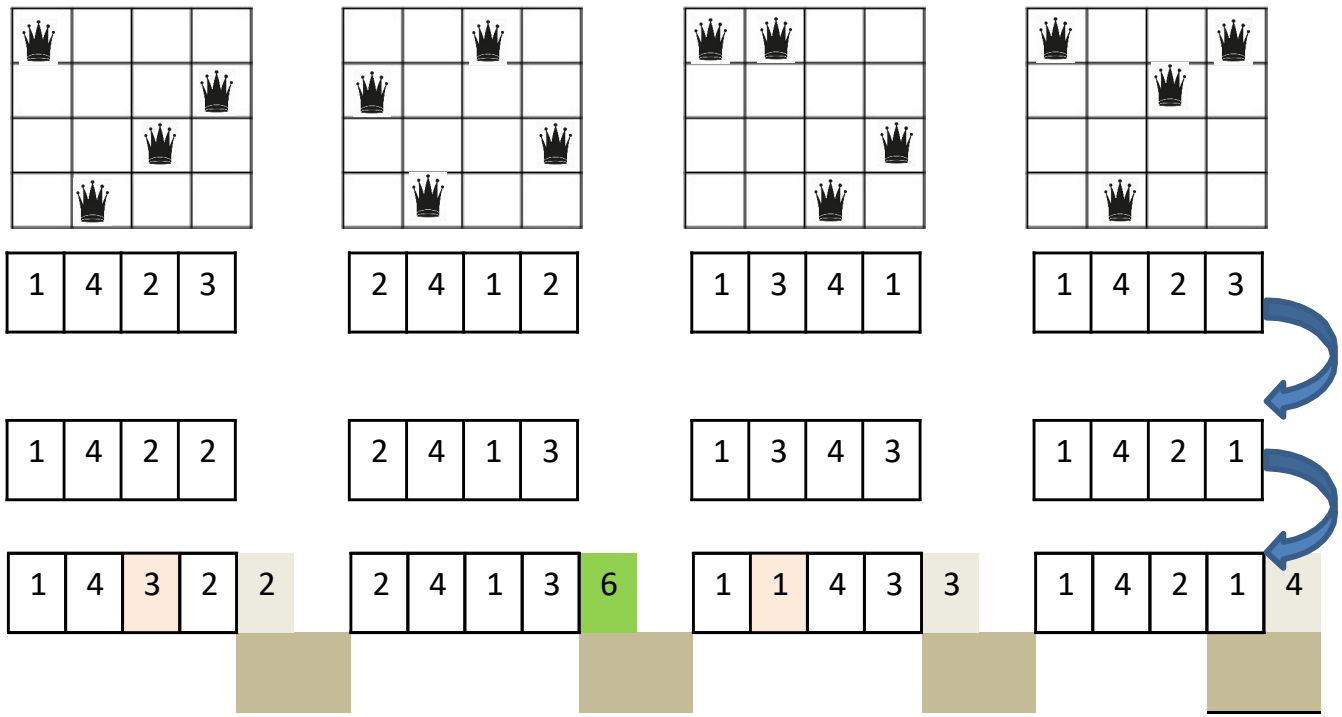
Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Genetic Algorithm

Techniques:

1. Design of the fitness function
2. Diversity in the population to be accounted
3. Randomization

Application:

- Creative tasks
- Exploratory in nature
- Planning problem
- Static Applications

Genetic Algorithm

```
function GENETIC-ALGORITHM( population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
     new_population ← empty set
     for i = 1 to SIZE( population) do
        x ← RANDOM-SELECTION( population, FITNESS-FN)
        y ← RANDOM-SELECTION( population, FITNESS-FN)
        child ← REPRODUCE(x, y)
        if (small random probability) then child ← MUTATE(child)
        add child to new_population
     population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Required Reading: AIMA – Chapter 4 - 4.1

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CLZG557

M2 : Problem Solving Agent using Search

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

Learning Objective

At the end of this class, students should be able to:

1. Convert a given problem into local search problem
2. Understand the significance of hyperparameters of the evolutionary algorithms
3. Apply appropriate local search and show the working of algorithm at least for first 2 iterations

Local Search & Optimization

Local Search

Optimization Problem

Goal : Navigate through a state space for a given problem such that an optimal solution can be found

Objective : Minimize or Maximize the objective evaluation function value

Scope : Local

Objective Function : Fitness Value evaluates the goodness of current solution

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

Single Instance Based	Multiple Instance Based
Hill Climbing	Genetic Algorithm
Simulated Annealing	Particle Swarm Optimization
Local Beam Search	Ant Colony Optimization
Tabu Search	

Particle Swarm Optimization

Swarm Intelligence

- Swarm Intelligence (SI) is artificial intelligence based on the collective behaviour of decentralized, self-organized systems.
- Characteristics of Swarms:
  - Composed of many individuals
  - Individuals are homogeneous
  - Local interaction based on simple rules (collision avoidance, velocity matching..)
  - Self-organization

Particle Swarm Optimization (PSO) (Contd..)

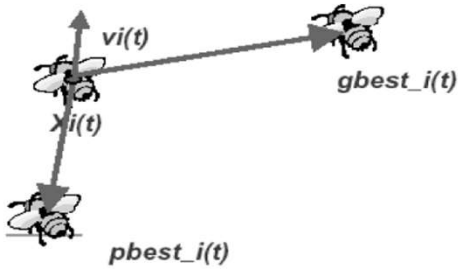
- Collection of flying particles (swarm) – Changing solutions
- Search area – Possible solutions
- Movement towards a promising area to get the global optimum
- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle keeps track:
  - Its best solution, personal best - pbest
  - The best value of any particle, global best – gbest.
- Each particle modifies its position according to:
  - Its current position
  - Its current velocity
  - The distance between its current position and pbest.
  - The distance between its current position and gbest

Particle Swarm Optimization (Contd..)

Each individual is called a particle

Particle = Position vector $x_i(t)$+ Velocity vector $v_i(t)$

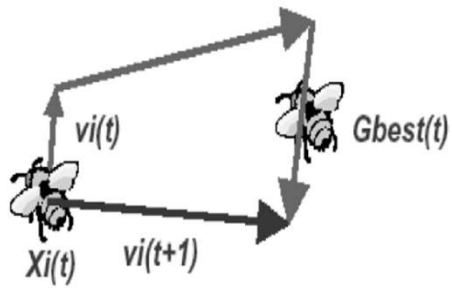
Heuristics Considered = function (Individual's Best So-far , Group's Best So-far)



Updating velocity vector:

$$X_i(t+1) = X_i(t) + V_i(t+1)$$

$$v_i(t+1) = \alpha\ v_i + c_1 \times rand \times (pbest(t)\text{-}x_i(t)) + c_2 \times rand \times (gbest(t)\text{-}x_i(t))$$

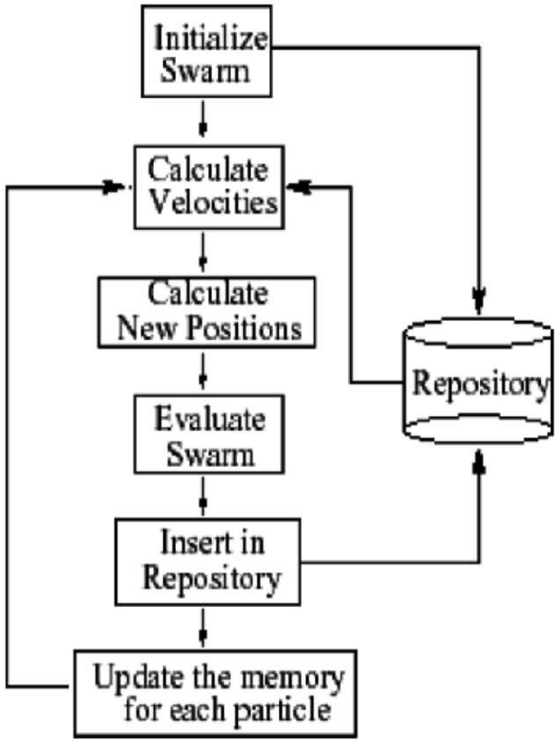


α is inertia weight and controls exploration and exploitation
c_1 and c_2 the cognition and social components respectively
rand is a random number generator

Particle Swarm Optimization (Contd..)

Basic Flow of PSO

1. Initialize the swarm with random initializations
2. Evaluate fitness value for each of these individuals
3. Modify **g_{best}, p_{best},** and velocity
4. Move each particle to new particle
5. Go to step 2, and repeat until convergence



Particles velocities on each dimension are clamped to a max velocity v_{max}

PSO vs GA

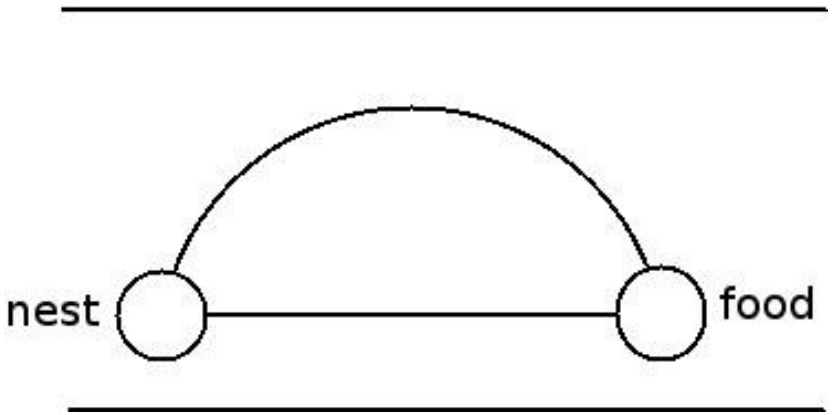
- Common points
  - Both algorithms start with a group of randomly generated population
  - Both have fitness value to evaluate the population
  - Both update the population and search for optimum with random sampling
  - Both do not guarantee success

- Difference
  - PSO does not have genetic operators like crossover and mutation
  - Particles update themselves
  - They also have memory which is critical to the algorithm
  - In GA, the information is shared between individuals, in PSO, the information is shared one-way.

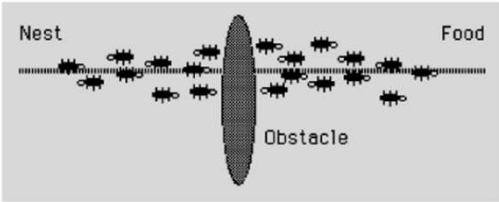
Ant Colony Optimization

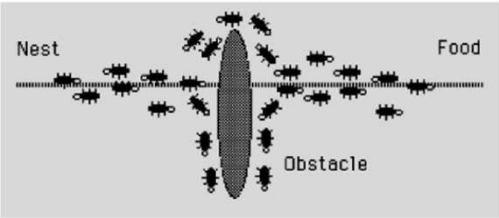
Ant Colony Optimization

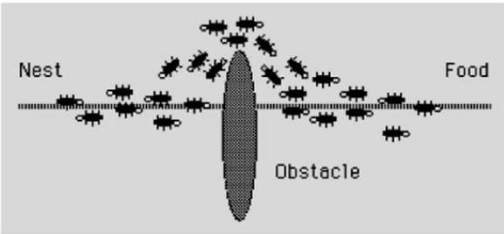
- The way ants find their food in shortest path is interesting
- Ants secret pheromones to remember their path
- These pheromones evaporate with time



Ant Colony Optimization







As ants have weak vision and weak memory, they communicate using the stimulus

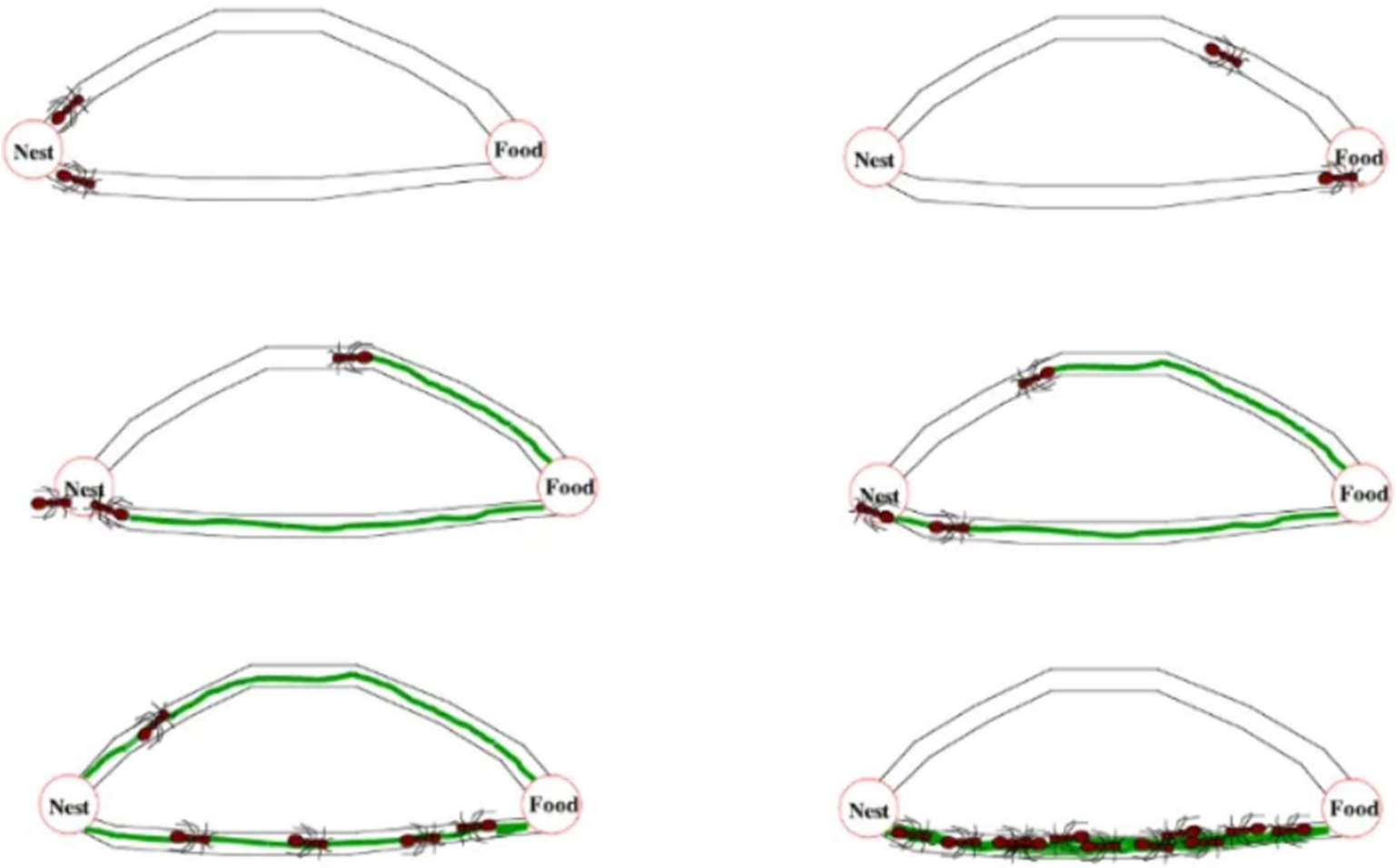
They leave chemical deposits called **Pheromone** to inform other ants about food, colonies, etc.

Using olfactory **stimulus** the insects follow the **trail** left behind by other ants

Other ants follow the pheromone trail and **deposit even more pheromone** leading to a positive feedback effect, **if the selected path is good.**

Ants **prefer** to follow a path with rich in pheromone

Ant Colony Optimization (Contd..)



Ant Colony Optimization

Problem Definition

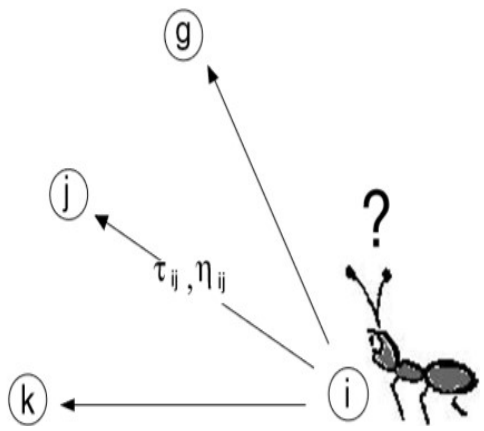
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones



m - No.of.Ants **n** - No.of.Nodes

Sk - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem
Cities = {i, j, k , g……}
$\Delta\tau_{ij}{}^{k}$ is the pheromone laid on edge (i, j) by ant k

Ants move through states of problem by applying a stochastic local decision policy based on two parameters, **trails and attractiveness**

When an ant completes a solution, or during construction phase, the solution is evaluated and trail values on components are modified

Ant Colony Optimization

Problem Definition

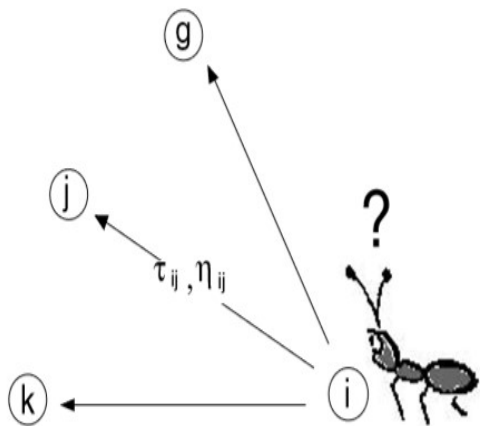
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones



m - No.of.Ants **n** - No.of.Nodes

S_k - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem
While real ants deposit chemical substance on the visited state, artificial ants change numerical information of problem state, stored locally when that state is visited

Ant Colony Optimization

Problem Definition

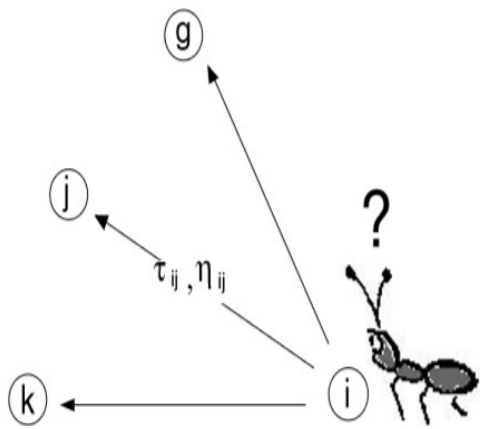
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones



m - No.of.Ants **n** - No.of.Nodes

Sk - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem
$\tau_0 = \frac{m}{C}$ $\tau_0 = \frac{1}{\rho C}$
ρ - In ACO and real ants it is important to slowly forget the history to avoid stacking in local extremes and move toward new regions using the evaporation rate

$$\Delta\tau_{ij}{}^{k} = \begin{cases} \frac{Q}{L_k} & if\ ant\ k\ used\ edge\ (i,j) \\ 0 & otherwise \end{cases}$$

where Q is a constant

L_k is the length of tour constructed by ant k

Ant Colony Optimization

Problem Definition

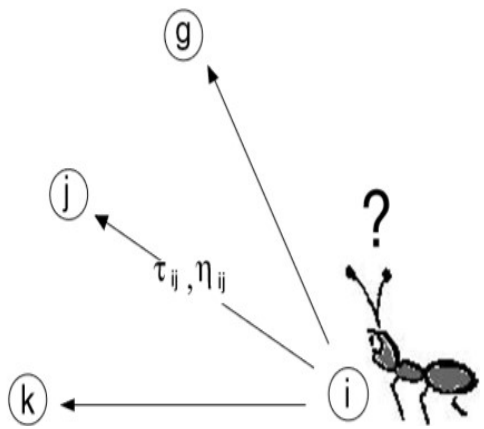
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones



m - No.of.Ants **n** - No.of.Nodes

S_k - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem
$n_{i,j}(S_k) = \frac{1}{d_{i,j}}$ d_{ij} is the distance between cities i and j $n_{i,j}$: To optimize for shortest paths, every state transition from i to j, is guided by the attractiveness as the heuristic

Ant Colony Optimization

Problem Definition

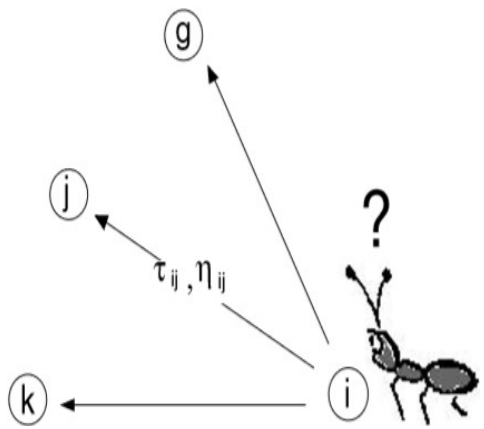
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution – Action Choice Rule

Update Pheromones



m - No.of.Ants **n** - No.of.Nodes

Sk - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}}{\sum_{c_{ij} \in N(s^p)} \tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}} & if\, c_{ij} \in N(s^p) \\ 0 & otherwise \end{cases}$$

Transition Policy: Decision making is based on probabilistic inference. It depends on local state information and existing pheromone trails

edge(i,j) where j is not yet visited $j \in \mathcal{N}_i^k$

Parameters α, β control relative importance of pheromone vs heuristic

Ant Colony Optimization

Problem Definition

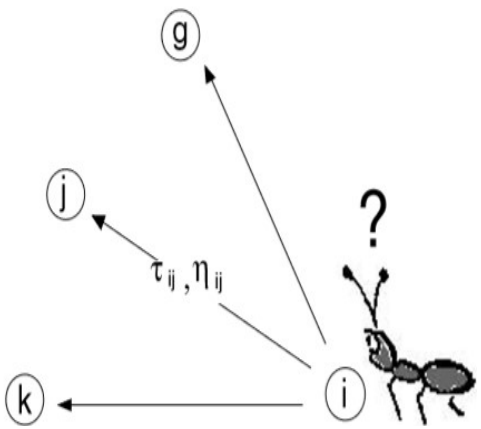
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones: Evaporation



m - No.of.Ants **n** - No.of.Nodes

Sk - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

Travelling Salesman Problem

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall (i,j) \in L$$

Local Update

$$\tau_{ij} = (1-\varphi) * \tau_{ij} + \varphi * \tau_0$$

A Local pheromone update is performed by all ants only to the last edge it traversed after each construction step

$\varphi \in (0,1)$ is pheromone decay coefficient

Global Update

$$\tau_{ij} \leftarrow \begin{cases} (1-\rho) * \tau_{ij} + \rho * \Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to best tour} \\ \tau_{ij} & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ij} = 1/L_{best}$$

Where L_{best} - length of tour of best ant either in current iteration or best so far

Applied at the end of iteration by only one ant (either iteration-best/best-so-far)

Ant Colony Optimization

Problem Definition

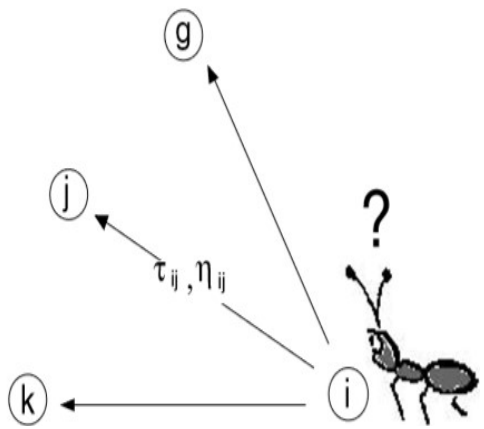
Create Construction Graph

Define & Initialize Pheromone Trails

Define Heuristics

Construct Solution

Update Pheromones: Deposit



m - No.of.Ants **n** - No.of.Nodes

Sk - Memory **α** - Influence of τ

β - Influence of n

ρ – Pheromone evaporation rate

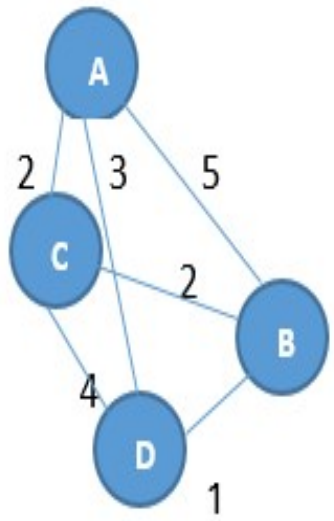
Travelling Salesman Problem
$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}^{best}$, where $\Delta\tau_{ij}^{best} = 1/C^{best}$.

Deposited amount of Pheromone: Depends on quality of discovered food source / cost of solution found so far

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	1	1	1	**A**
	0	1	1	**B**
		0	1	**C**
			0	**D**

A	B	C	D	
0	5	2	3	**A**
	0	2	1	**B**
		0	4	**C**
			0	**D**



Packet routing:

Optimal path finder for communication over network.

No.of.Ants : **m**= 5

Pheromone evaporation rate: **ρ** =0.5

α - Influence of τ = 1

β - Influence of n = 1

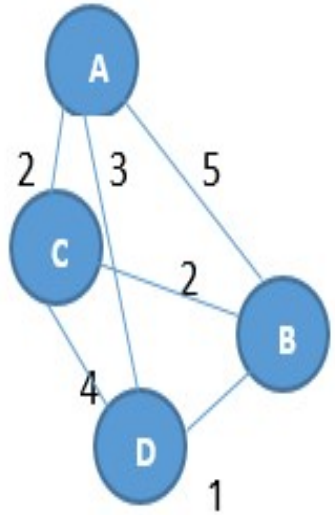
$\tau_{i,j}$ - Pheromone level = 1

1. Represent the solution space by a construction graph
2. Initialize ACO parameters
3. Generate random solutions from each ant's random walk
4. Update pheromone information
5. Go to 3 and repeat until convergence or a stopping criteria is met

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	1	1	1	**A**
	0	1	1	**B**
		0	1	**C**
			0	**D**

A	B	C	D	
0	5	2	3	**A**
	0	2	1	**B**
		0	4	**C**
			0	**D**



<u>Iteration 1: 5 packets randomly sent</u>

1: ABDCA → C_k = AB+BD+DC+CA = 12

2: ADBCA → $C_k = 8$

3: ABDCA → $C_k = 12$

4: ADBCA → $C_k = 8$

5: ACBDA → $C_k = 8$

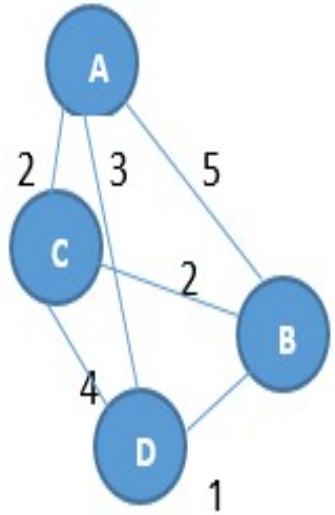
This is a random trail .
Typically the algorithm uses the probability value based Transition Policy for optimal search to construct the paths

1. Represent the solution space by a construction graph
2. Initialize ACO parameters
3. Generate random solutions from each ant's random walk
4. Update pheromone information
5. Go to 3 and repeat until convergence or a stopping criteria is met

Ant Colony Optimization - Sample Problem

A	B	C	D	
0	~~1~~ **0.58**	1	1	**A**
	0	1	1	**B**
		0	1	**C**
			0	**D**

A	B	C	D	
0	5	2	3	**A**
	0	2	1	**B**
		0	4	**C**
			0	**D**



Iteration 1: 5 packets

1: **AB**DCA → $C_k = 12$

2: ADBCA → $C_k = 8$

3: **AB**DCA → $C_k = 12$

4: ADBCA → $C_k = 8$

5: ACBDA → $C_k = 8$

$$\Delta \tau_{ij} = \frac{1}{Ck}$$

Pheromone Update

AB:

2 ants with $C_k = 12$

$ant\ 1: \Delta \tau_{ij} = \frac{1}{12}$

$ant\ 3: \Delta \tau_{ij} = \frac{1}{12}$

AB: $\tau_{ij} \leftarrow (1 - \rho) * \tau_{ij} + \rho \sum_{k=1}^{m} \Delta \tau_{ij}{}^{k}$

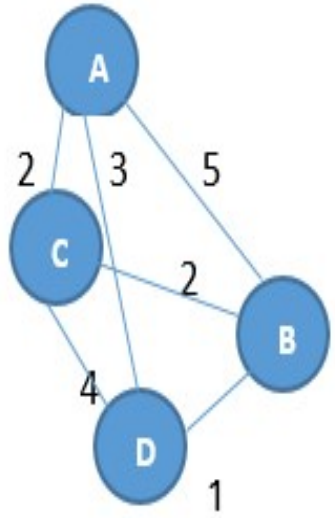
$$\tau_{ij} \leftarrow (1 - 0.5) * 1 + \left(\frac{1}{12} + \frac{1}{12}\right) * 0.5$$

$\boldsymbol{\tau_{ij}} \leftarrow \mathbf{0.58}$

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	0.58	1.04	0.875	A
	0	0.875	1.04	B
		0	0.75	C
			0	D

A	B	C	D	
0	5	2	3	A
	0	2	1	B
		0	4	C
			0	D



Iteration 1: (Above table is a random values filled for rest. Students should try to work out for the right values.)

1: ABDCA → C_k =12

2: ADBCA → C_k = 8

3: ABDCA → C_k = 12

4: ADBCA → C_k = 8

5: ACBDA → C_k = 8

$$\Delta \tau_{ij} = \frac{1}{Ck}$$

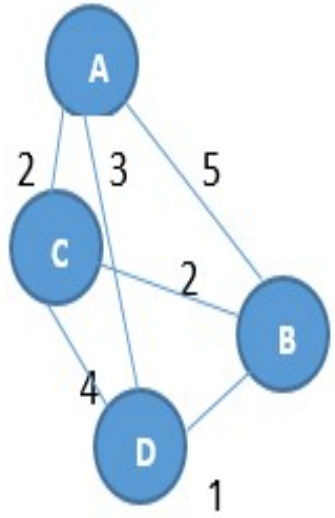
$$\tau_{ij} \leftarrow (1-\rho) * \tau_{ij} + \rho \sum_{k=1}^{m} \Delta \tau_{ij}^{k}$$

1. Represent the solution space by a construction graph
2. Initialize ACO parameters
3. Generate random solutions from each ant's random walk
4. Update pheromone information
5. Go to 3 and repeat until convergence or a stopping criteria is met

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	0.58	1.04	0.875	A
	0	0.875	1.04	B
		0	0.75	C
			0	D

A	B	C	D	
0	5	2	3	A
	0	2	1	B
		0	4	C
			0	D



Iteration 2: 3 packets randomly sent

1: BCADB→ C_k =8

2: BDACB → C_k = 8

3: DBACD → C_k = 12

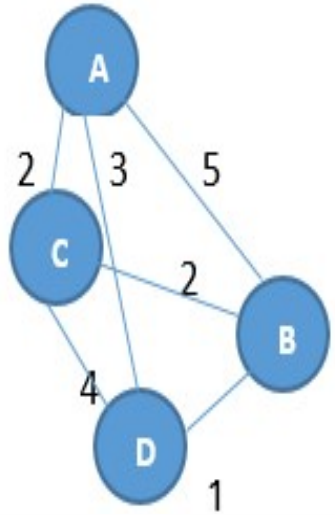
$$\tau_{ij} \leftarrow (1-\rho) * \tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k}$$

$$\Delta\tau_{ij} = \frac{1}{Ck}$$

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	0.58	~~1.04~~ 0.687	0.875	A
	0	0.875	1.04	B
		0	0.75	C
			0	D

A	B	C	D	
0	5	2	3	A
	0	2	1	B
		0	4	C
			0	D



Iteration 2: 3 packets

1: BCADB→ $C_k = 8$

2: BDACB → $C_k = 8$

3: DBACD → $C_k = 12$

$$\Delta \tau_{ij} = \frac{1}{Ck}$$

Pheromone Update

AC:

3 ants with $C_k = 12, 8, 8$

$ant\ 1: \Delta \tau_{ij} = \frac{1}{8}$

$ant\ 2: \Delta \tau_{ij} = \frac{1}{8}$

$ant\ 3: \Delta \tau_{ij} = \frac{1}{12}$

AC: $\tau_{ij} \leftarrow (1 - \rho) * \tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k}$

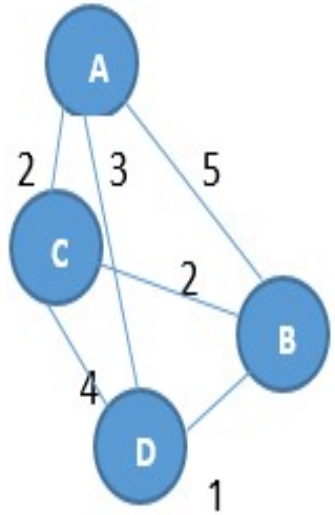
$$\tau_{ij} \leftarrow (1 - 0.5) * 1.04 + 0.5\left(\frac{1}{8} + \frac{1}{8} + \frac{1}{12}\right)$$

$$\boldsymbol{\tau_{ij}} \leftarrow \mathbf{0.687}$$

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	**0.413**	**0.853**	**0.687**	**A**
	0	**0.687**	**0.853**	**B**
		0	**0.458**	**C**
			0	**D**

A	B	C	D	
0	5	2	3	**A**
	0	2	1	**B**
		0	4	**C**
			0	**D**



Iteration 2: 3 packets

1: BCADB→ C_k =8

2: BDACB → C_k = 8

3: DBACD → C_k = 12

(Above table is a random values filled for rest. Students should try to work out for the right values.)

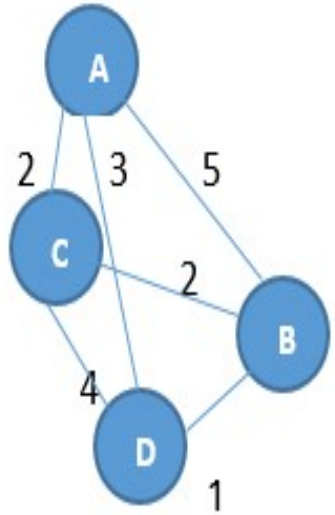
$$\tau_{ij} \leftarrow (1-\rho) * \tau_{ij} + \rho \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

$$\Delta\tau_{ij} = \frac{1}{Ck}$$

Ant Colony Optimization – Sample Problem

A	B	C	D	
0	**0.413**	**0.853**	**0.687**	**A**
	0	**0.687**	**0.853**	**B**
		0	**0.458**	**C**
			0	**D**

A	B	C	D	
0	5	2	3	**A**
	0	2	1	**B**
		0	4	**C**
			0	**D**



Solution Choice:

A new packet arrives at router C . Which is the next hop ?

$$CA := \frac{\tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}}{\sum_{c_{ij}\in N(s^p)} \tau_{ij}^{\alpha}\,\eta_{ij}^{\beta}} = \frac{0.853*\left(\frac{1}{2}\right)}{\left(0.853*\frac{1}{2}+0.687*\frac{1}{2}+0.458*\frac{1}{4}\right)} \sim 60-70\%$$

Similarly calculate for below edges as well before deciding the next node to send packets to.

CB:=

CD:=

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in\mathcal{N}_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, \quad \text{if } j\in\mathcal{N}_i^k$$

Required Reading: Some Reading materials are uploaded in the canvas pages

Research papers & web resources

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Artificial & Computational Intelligence

AIML CLZG557

M3 : Game Playing

Dr. Sudheer Reddy

BITS Pilani
Pilani Campus

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time

M7 Ethics in AI

Module 3 : Searching to play games

A. Minimax Algorithm

B. Alpha-Beta Pruning

C. Making imperfect real time decisions

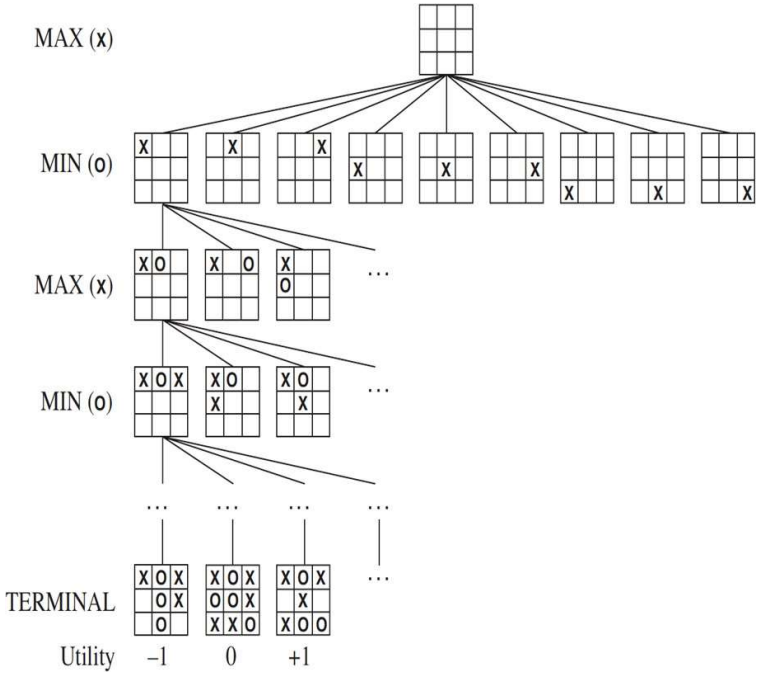
Learning Objective

At the end of this class , students should be able to:

1. Convert a given problem into adversarial search problem
2. Formulate the problem solving agent components
3. Design static evaluation function value for a problem
4. Construct a Game tree
5. Apply Min-Max
6. Apply and list nodes pruned by alpha pruning and nodes pruned by beta pruning

Task Environment

Phases of Solution Search by PSA



Assumptions – Environment :

Static

Observable

Discrete

Deterministic

Number of Agents

Game Problem

Study & design of games enables the computers to model ways in which humans think & act hence simulating human intelligence.

<u>AI for Gaming:</u>

- Interesting & Challenging Problem
- Larger Search Space Vs Smaller Solutions
- Explore to better the Human Computer Interaction

<u>Characteristics of Games:</u>

- Observability
- Stochasticity
- Time granularity
- Number of players

<u>Adversarial Games:</u>

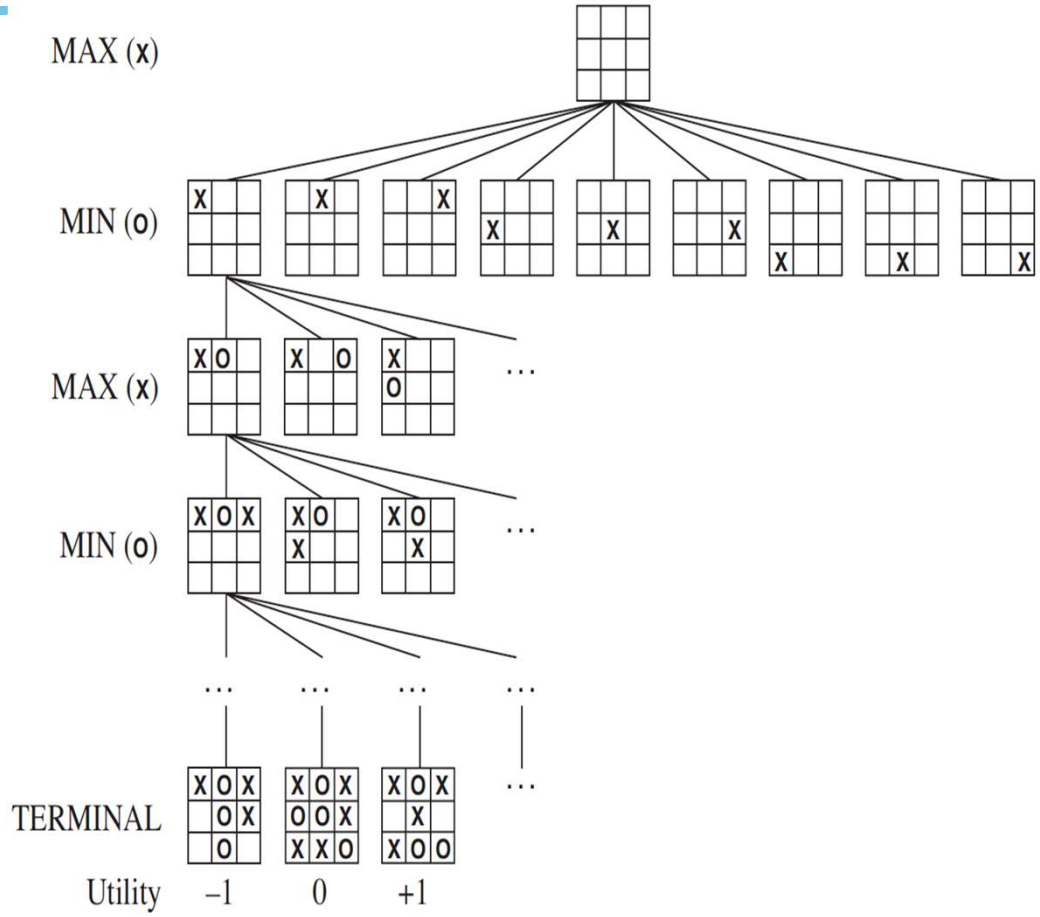
Goals of agents are in conflict where one's optimized step would reduce the utility value of the other.

Games as Search Problem

PSA : Representation of Game:

- INITIAL STATE: S0
- PLAYER(s)
- ACTIONS(s)
- RESULT(s, a)
- TERMINAL-TEST(s)
- UTILITY(s, p)

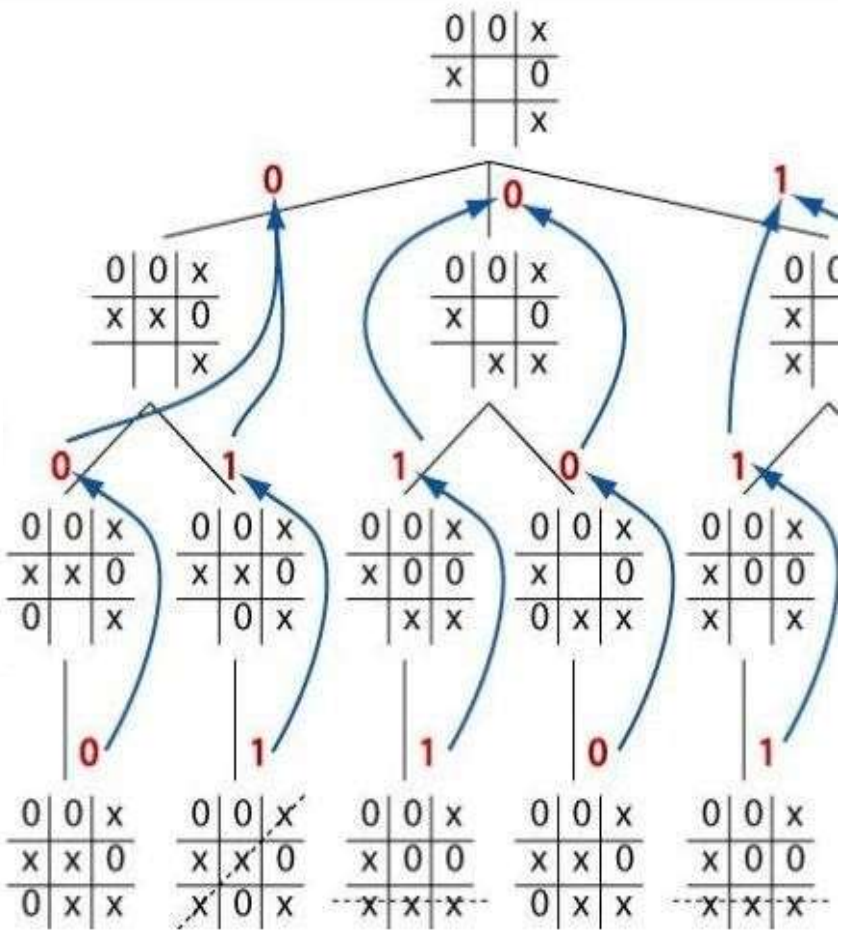
Eg., Tic Tac Toe



Min-Max Algorithm

<u>Idea:</u> Uses Depth – First search exploration to decide the move

Let
start Player = MAX
Depth m =3



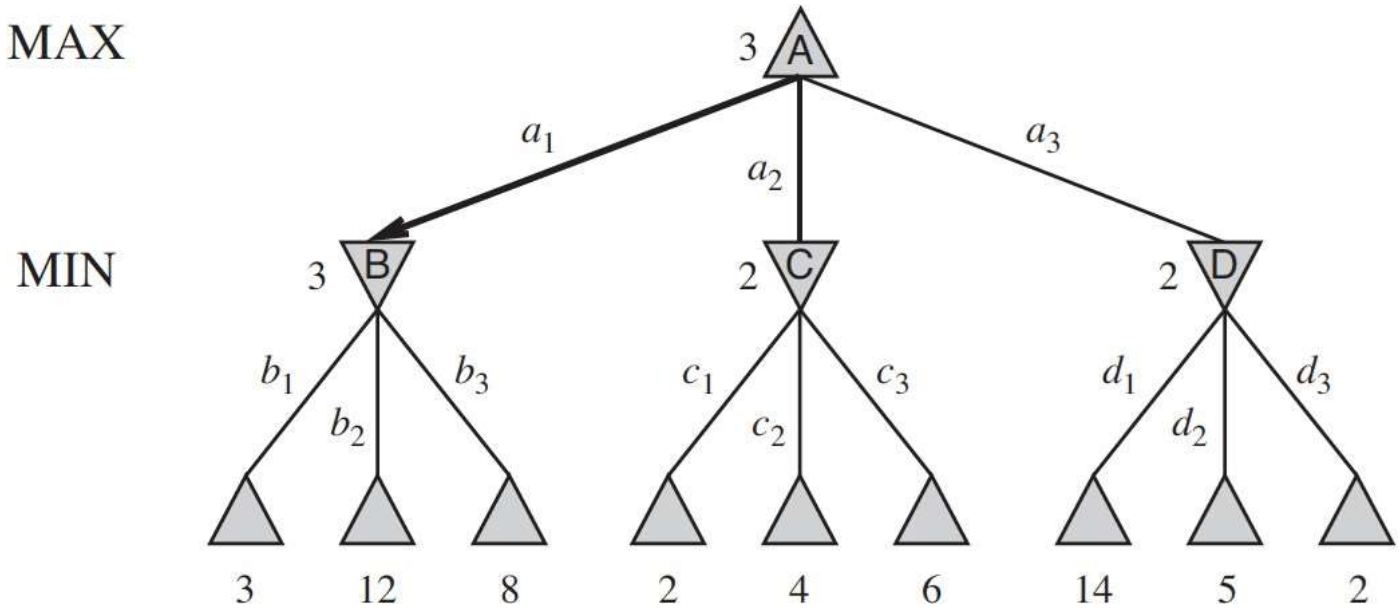
Min-Max Algorithm

```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))
```

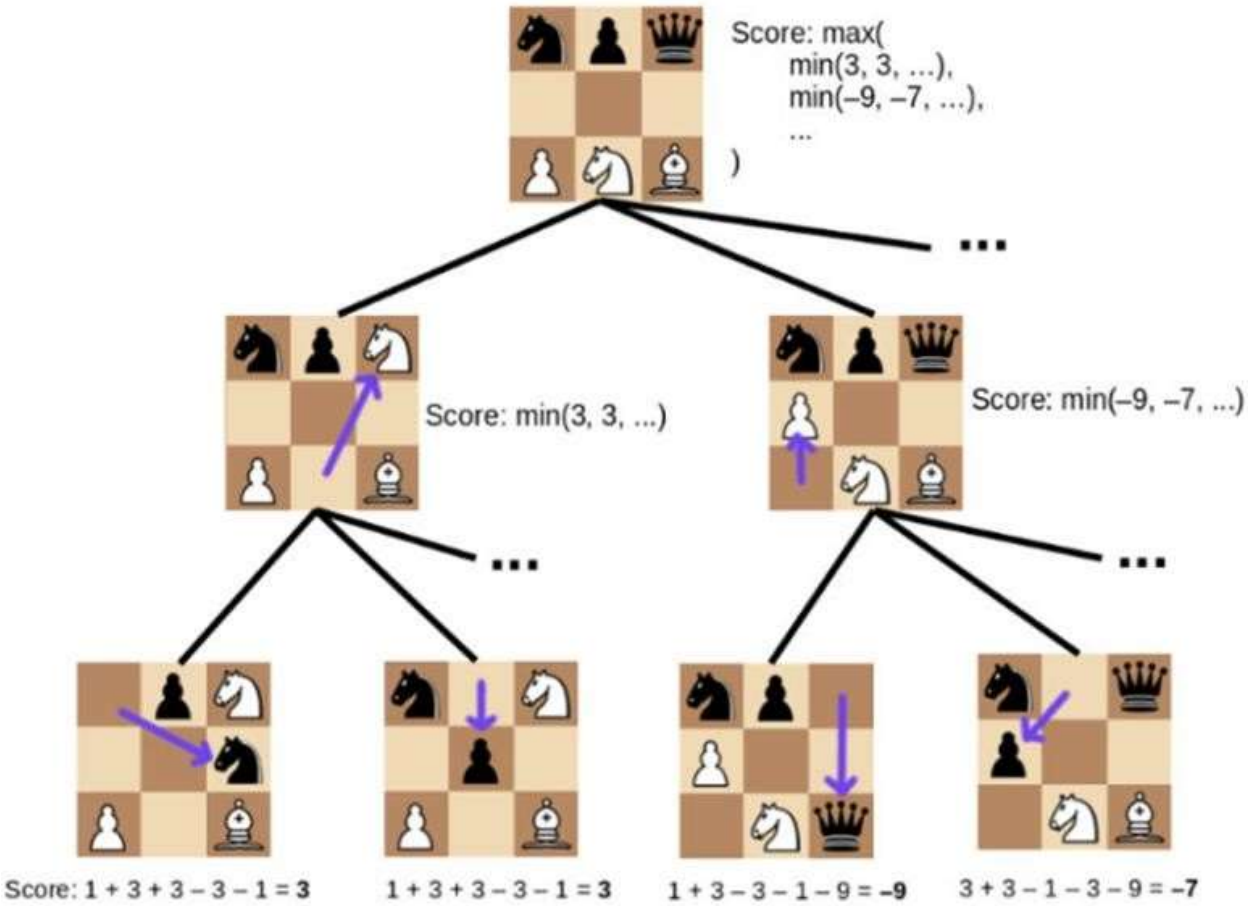
```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

Book Example

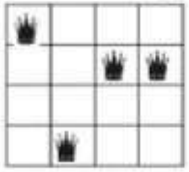


Design of Static Evaluation Values



Design of Static Evaluation Values

N-Queens



1	4	2	2	4

Tic-Tac-Toe

0	0	x
x		0
		x

Max's Share	2
Min's Share	1
Board Value	1

N-Tile

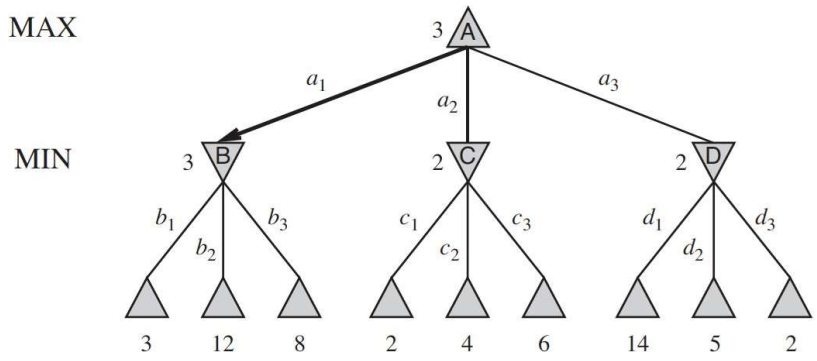
2	8	3
1	6	4
7		5

1	2	3
8		4
7	6	5

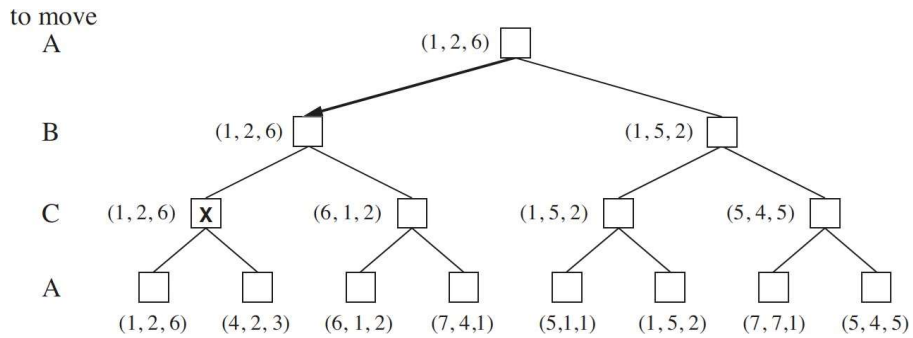
No.of.Tiles Out of Place	5

$$\text{Eval}(S) = w_1 f_1(s) + w_2 f_2(s) + \ldots\ldots + w_n f_n(s)$$

$$= 0.6\,(\text{MaxChance} - \text{MinChance}) + 0.4\,(\text{MaxPairs} - \text{MinPairs})$$



Two Player Game : 1-Ply Game

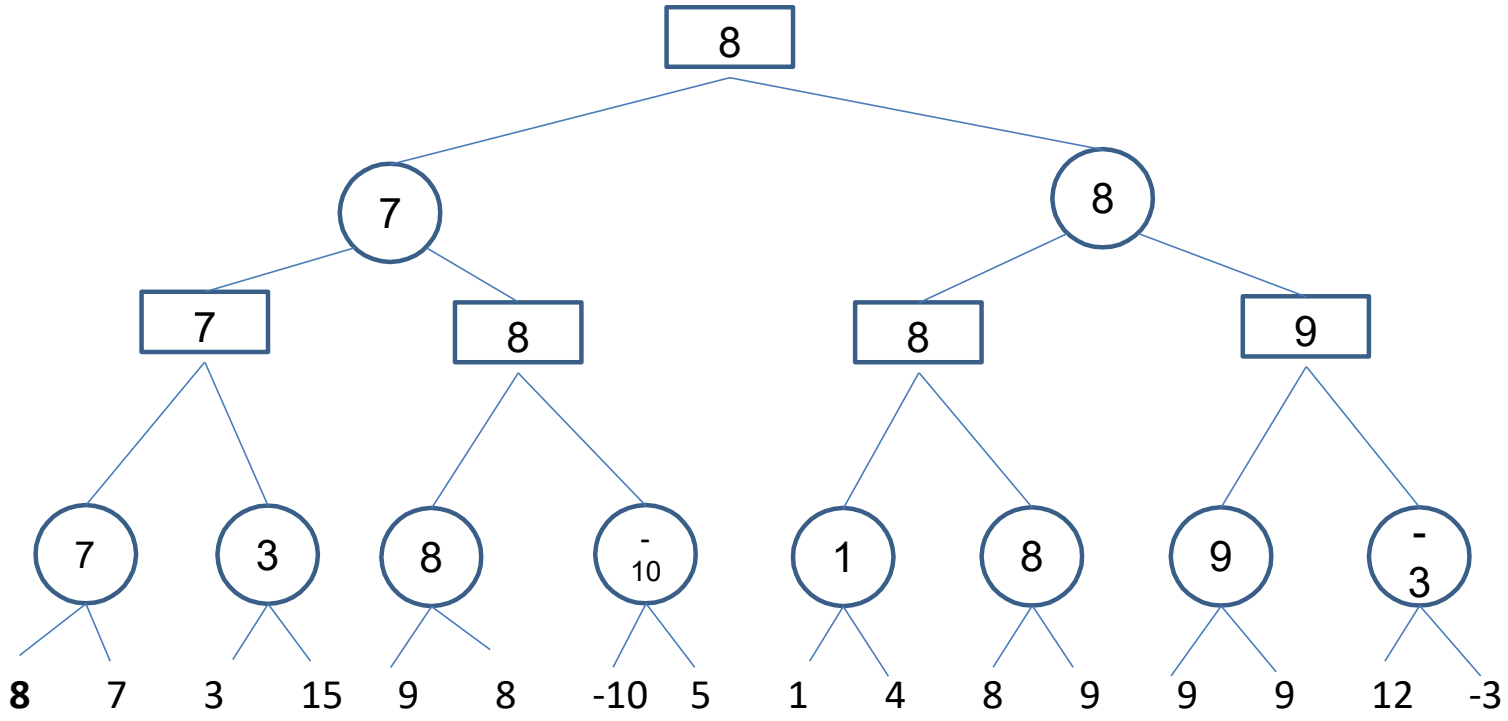


Multiplayer Game

Min-Max Algorithm – Example -1

Squares represent MAX nodes

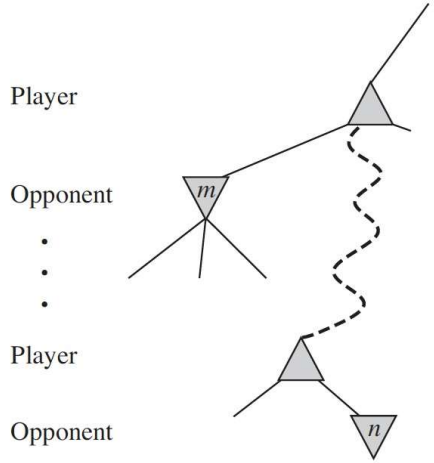
Circles represent MIN nodes

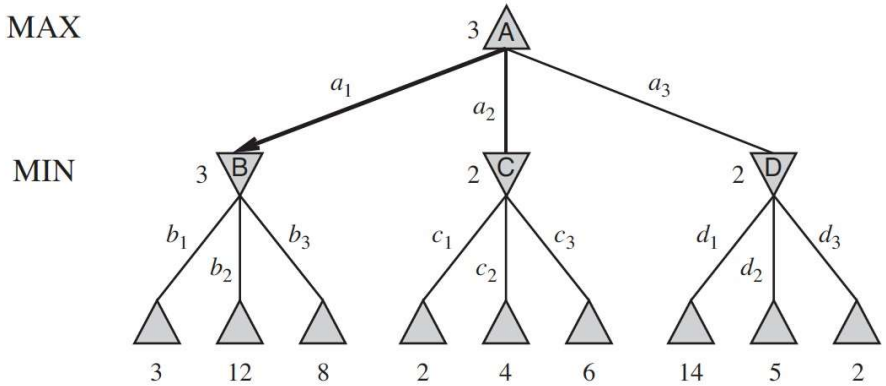


Alpha – beta Pruning

General Principle:

At a node ***n*** if a player has better option at the parent of ***n*** or further up, then ***n*** node will never be reached .Hence the entire subtree pruned

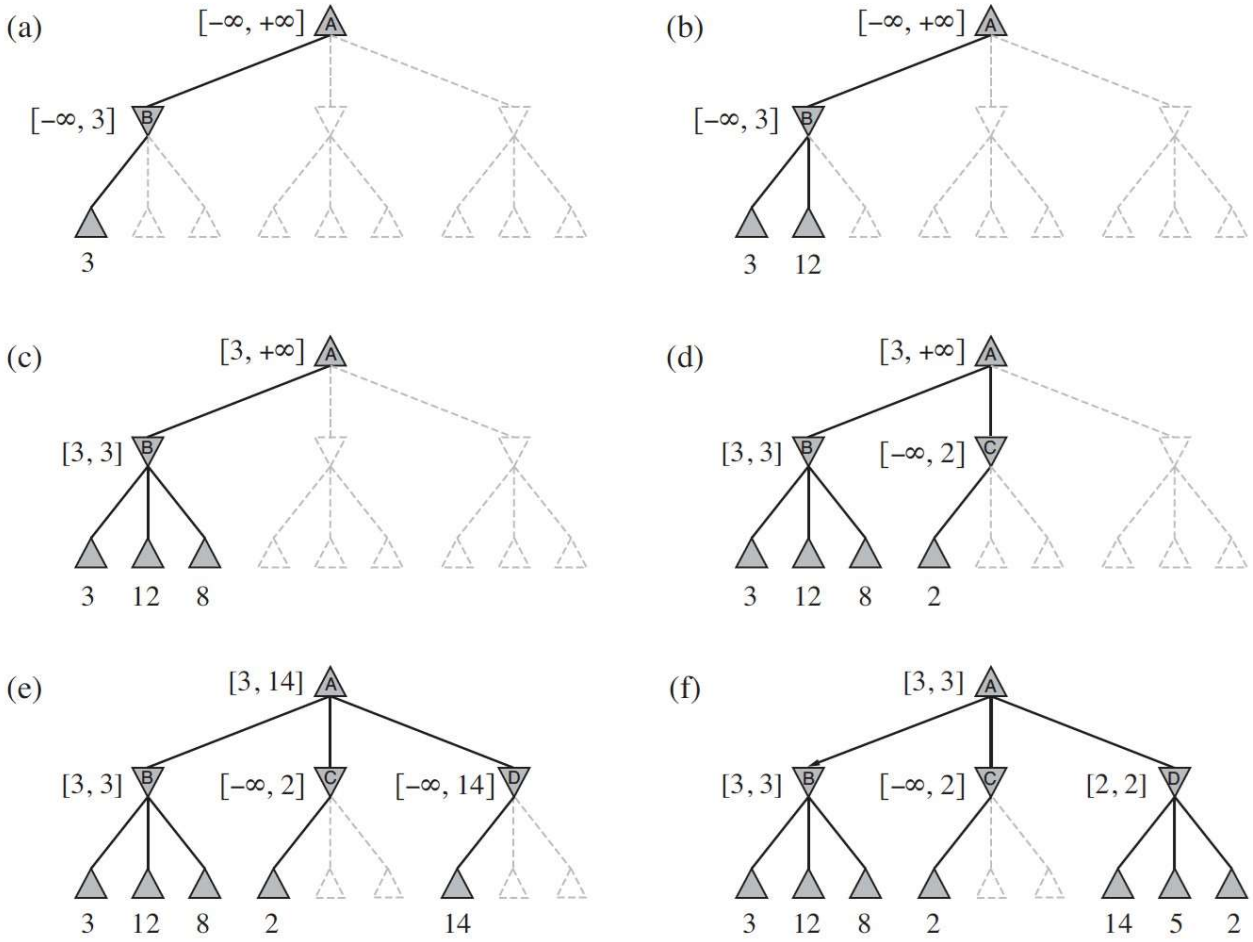




$$
\begin{aligned}
\text{MINIMAX}(root) &= \max(\min(3,12,8), \min(2,x,y), \min(14,5,2)) \\
&= \max(3, \min(2,x,y), 2) \\
&= \max(3, z, 2) \qquad \text{where } z = \min(2,x,y) \le 2 \\
&= 3.
\end{aligned}
$$

Alpha Beta Pruning

Book Example



Min-Max Algorithm

Alpha beta Modifications

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

```
function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

Is it possible to compute the minimax decision for a node without looking at every successor node?

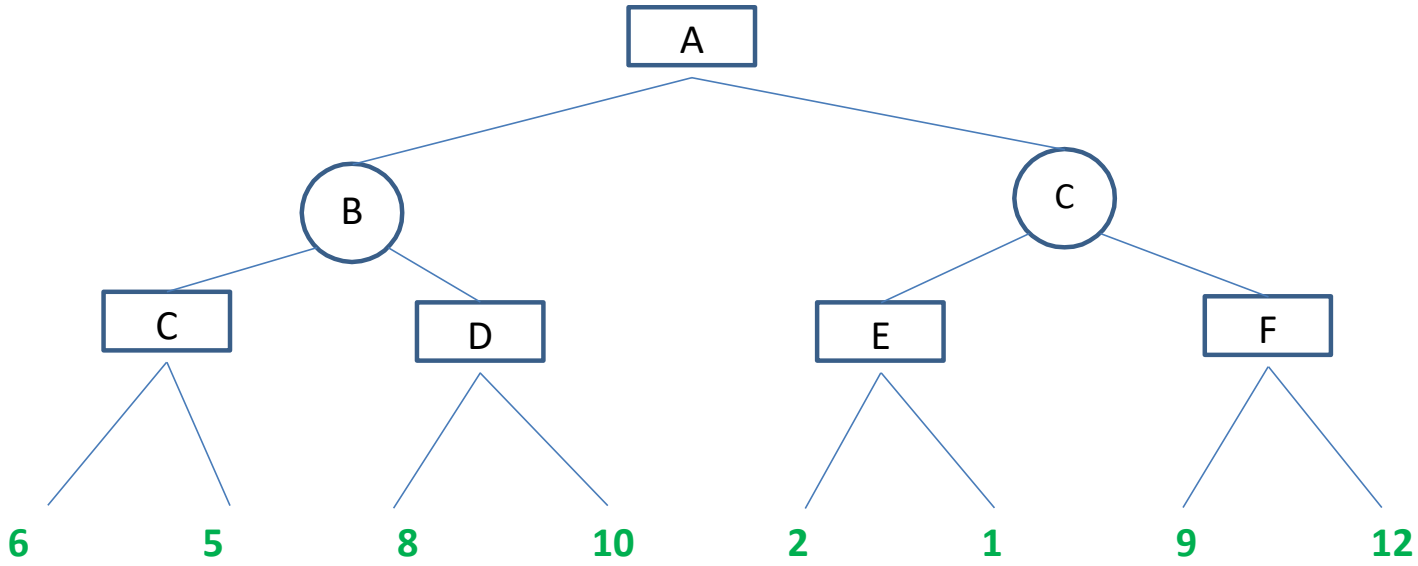
Alpha – beta Pruning

Steps in Alpha – Beta Pruning:

1. At root initialize alpha = -∞ and beta = + ∞. This is to set the worst case boundary to start the algorithm which aims to increase alpha and decrease beta as much as optimally possible
2. Navigate till the depth / limit specified and get the static evaluated numeric value.
3. For every value VAL being analyzed : Loop till all the leaf/terminal/specified state level nodes are analyzed & accounted for OR until **beta <= alpha.**
    1. If the player is MAX :
        1. If VAL > alpha
        2. then reset alpha = VAL
        3. also check **if** beta <=alpha **then** tag the path as unpromising (TO BE AVOIDED) **and** prune the branch from game tree. Rest of their siblings are not considered for analysis
    2. Else if the player is MIN:
        1. If VAL < beta
        2. then reset beta = VAL
        3. also check **if** beta <=alpha **then** tag the path as unpromising (TO BE AVOIDED) **and** prune the branch from game tree. Rest of their siblings are not considered for analysis

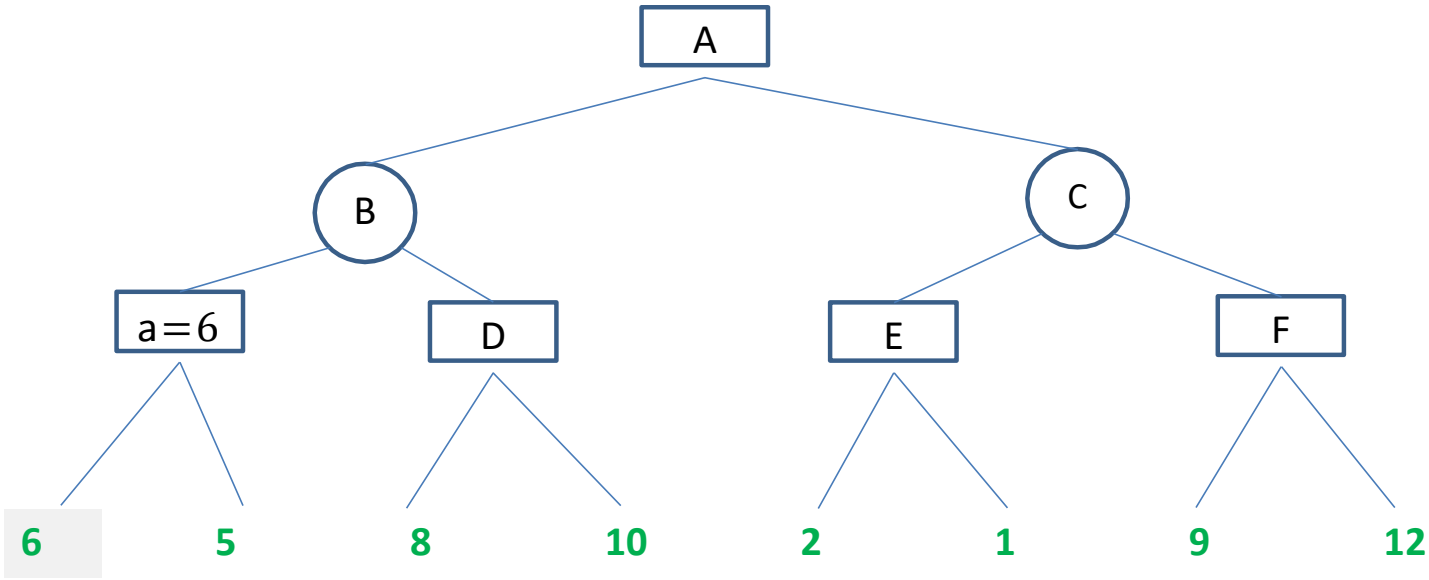
Alpha Beta Pruning - Another Example

Idea –Pruning



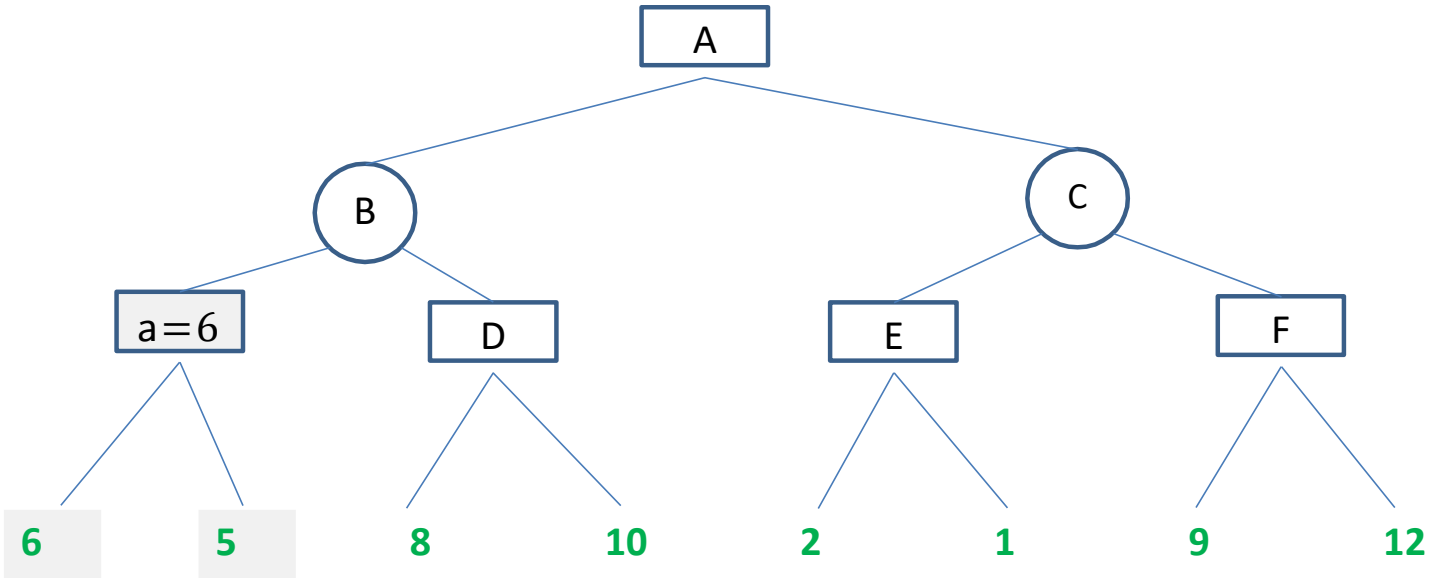
Alpha Beta Pruning

Idea –Pruning



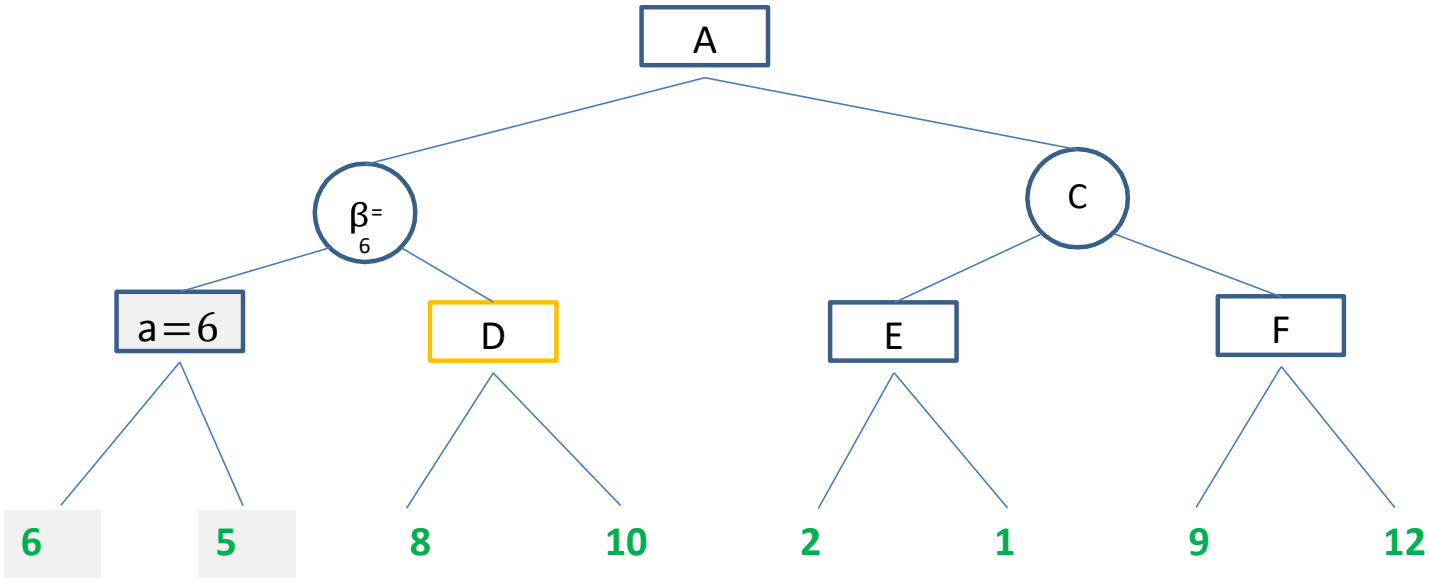
Alpha Beta Pruning

Idea –Pruning



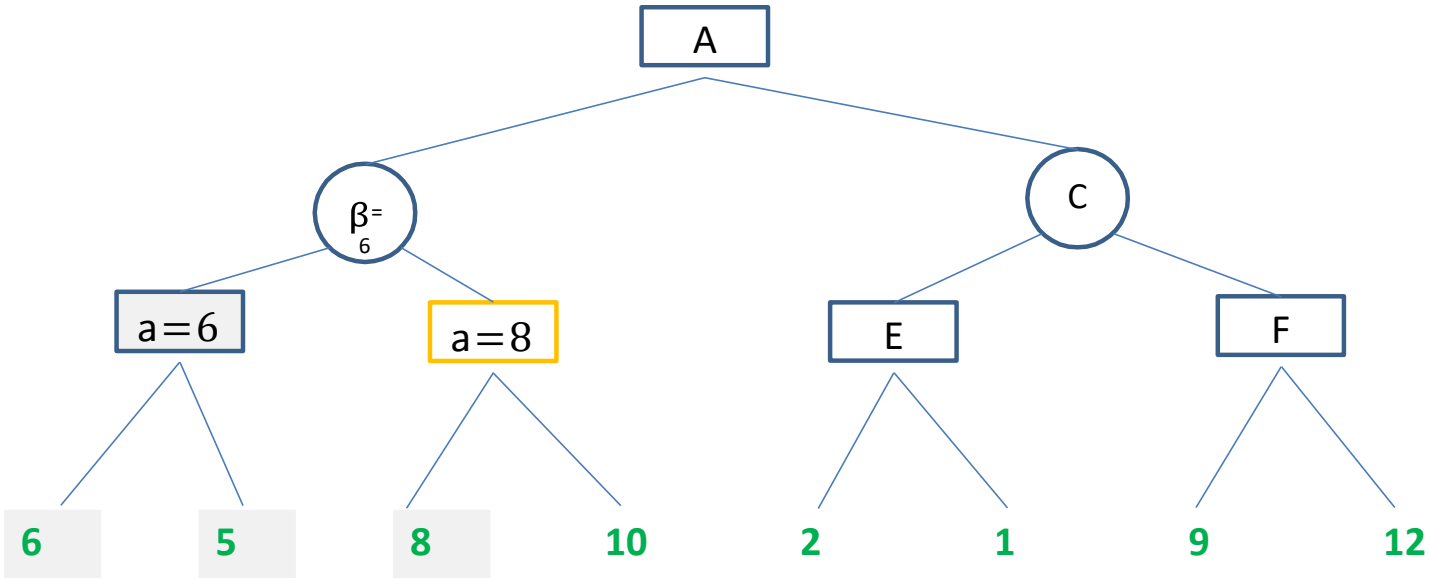
Alpha Beta Pruning

Idea –Pruning



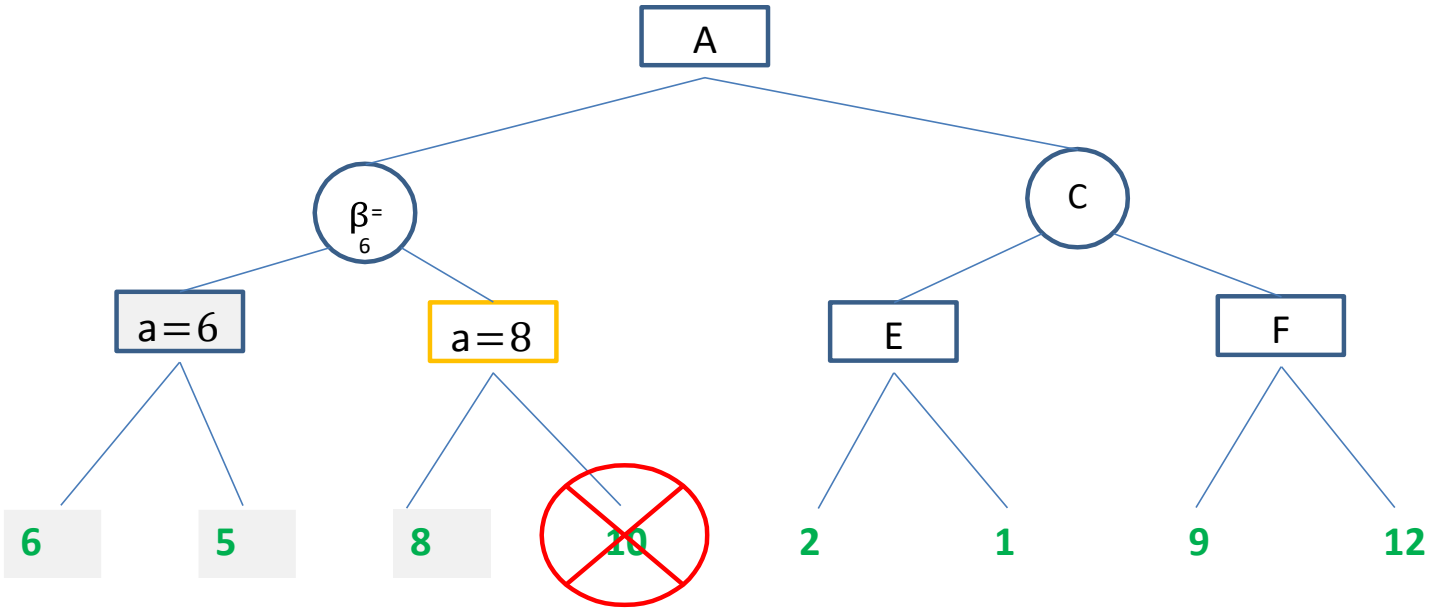
Alpha Beta Pruning

Idea – Alpha Pruning



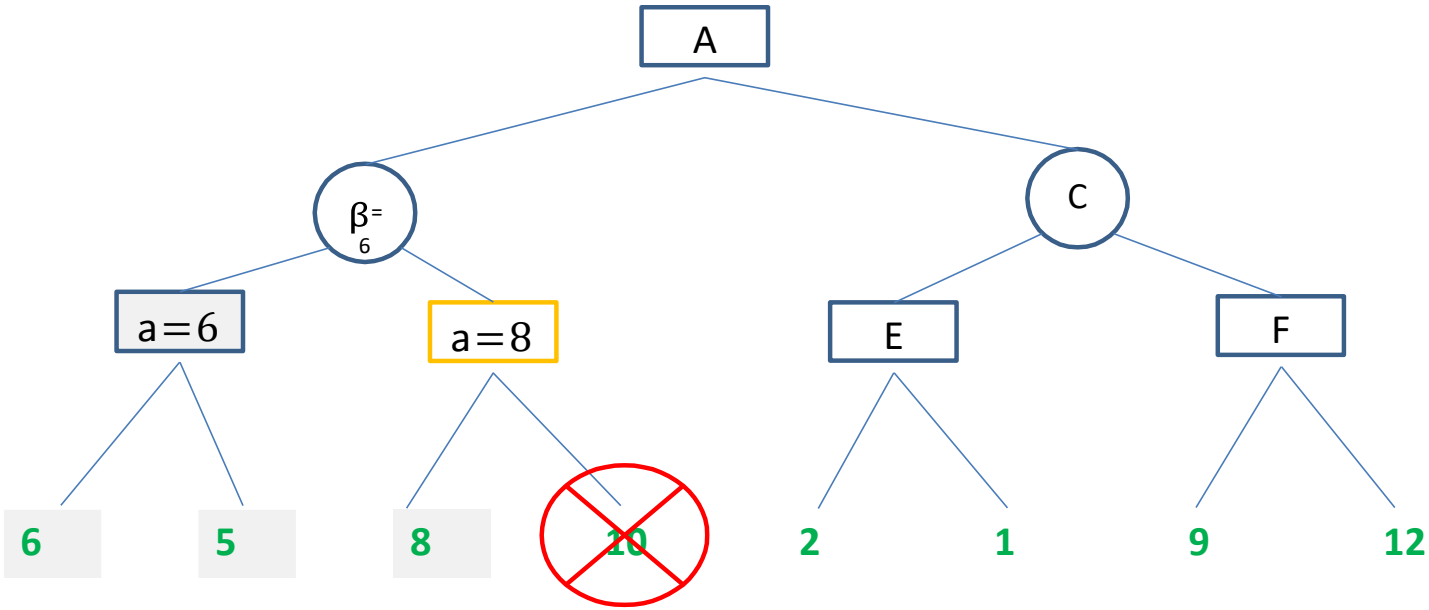
Alpha Beta Pruning

Idea – Beta Pruning



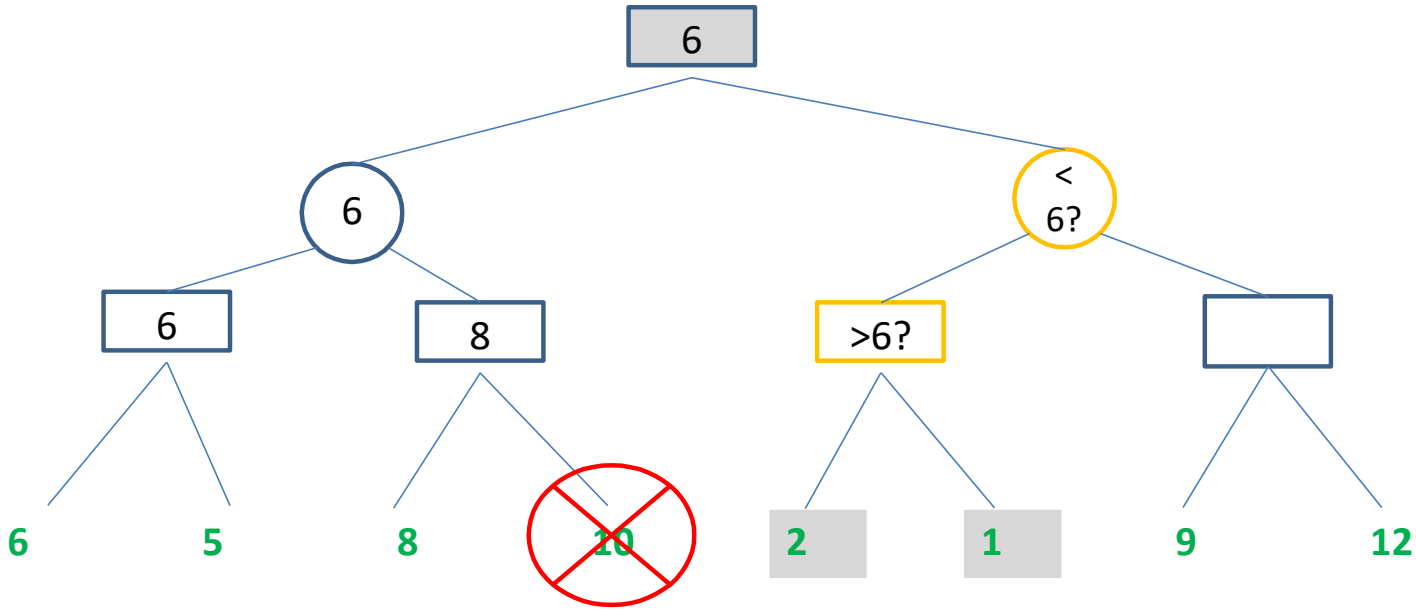
Alpha Beta Pruning

Idea –Pruning



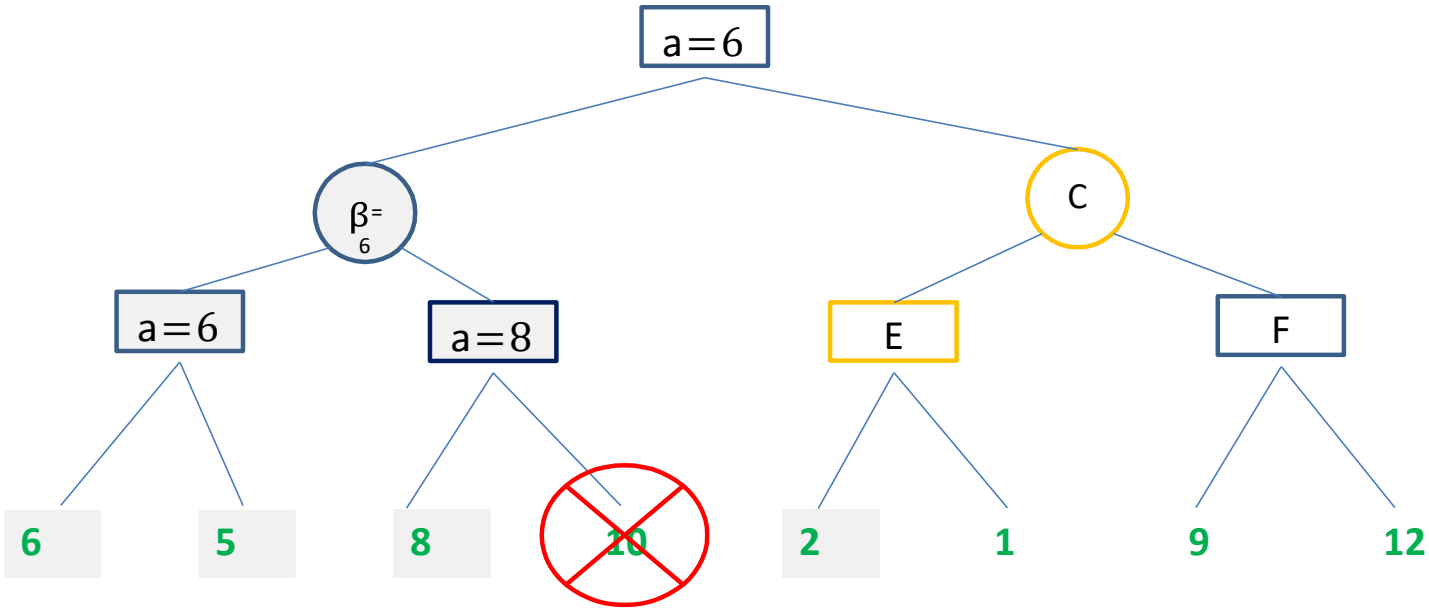
Alpha Beta Pruning

Idea –Pruning



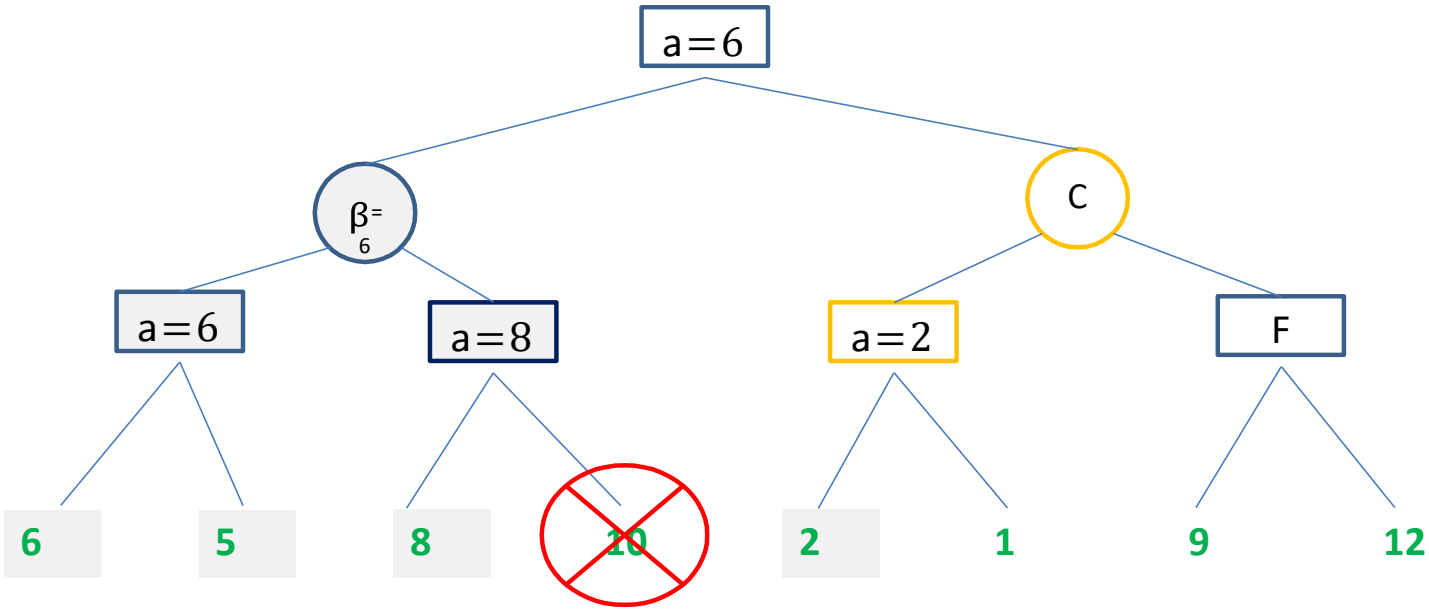
Alpha Beta Pruning

Idea –Pruning



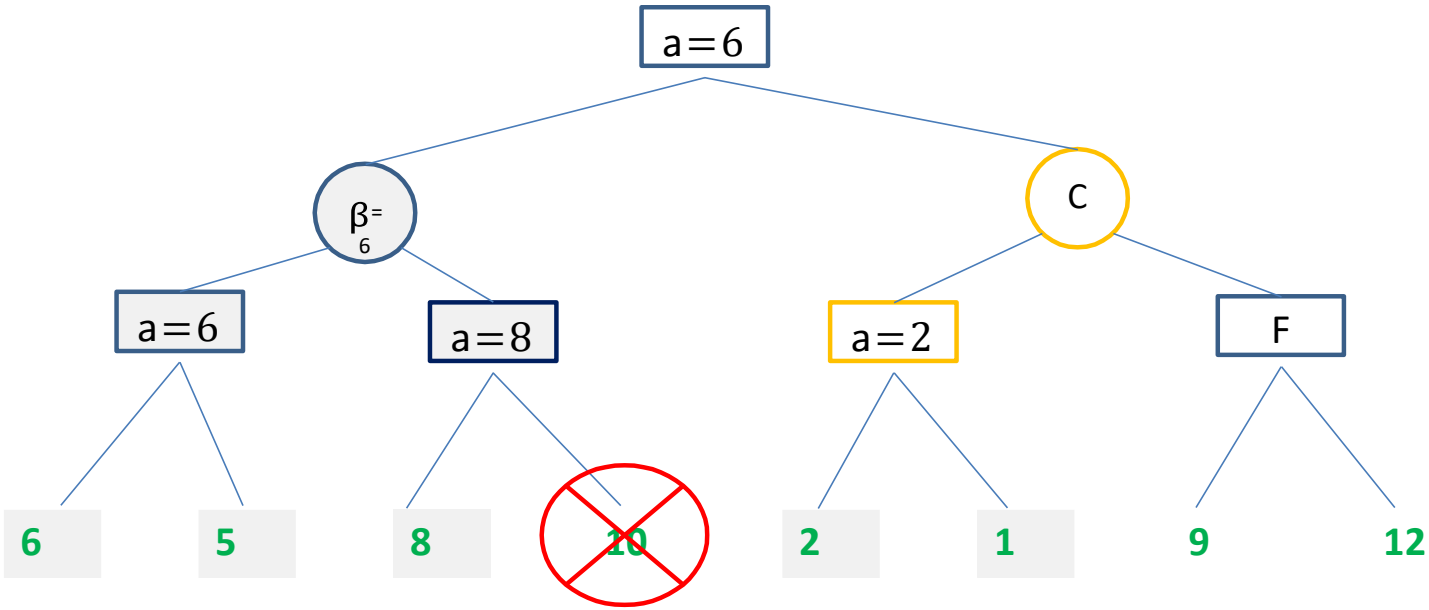
Alpha Beta Pruning

Idea –Pruning



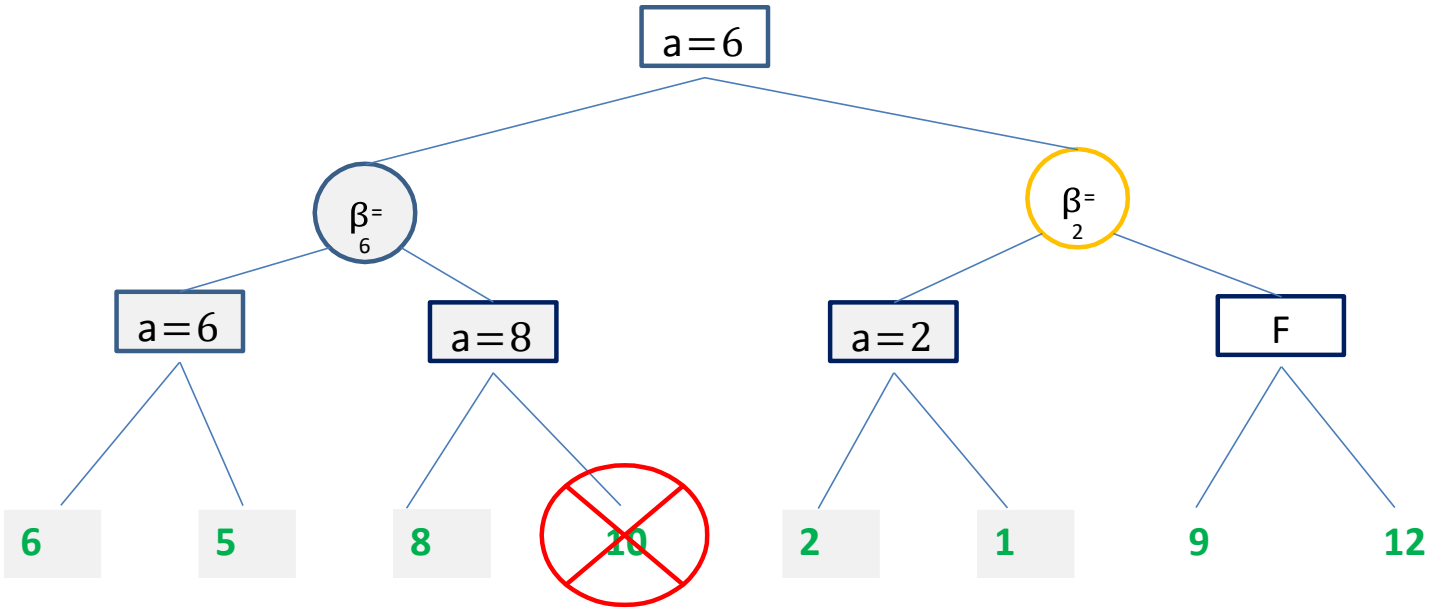
Alpha Beta Pruning

Idea –Pruning

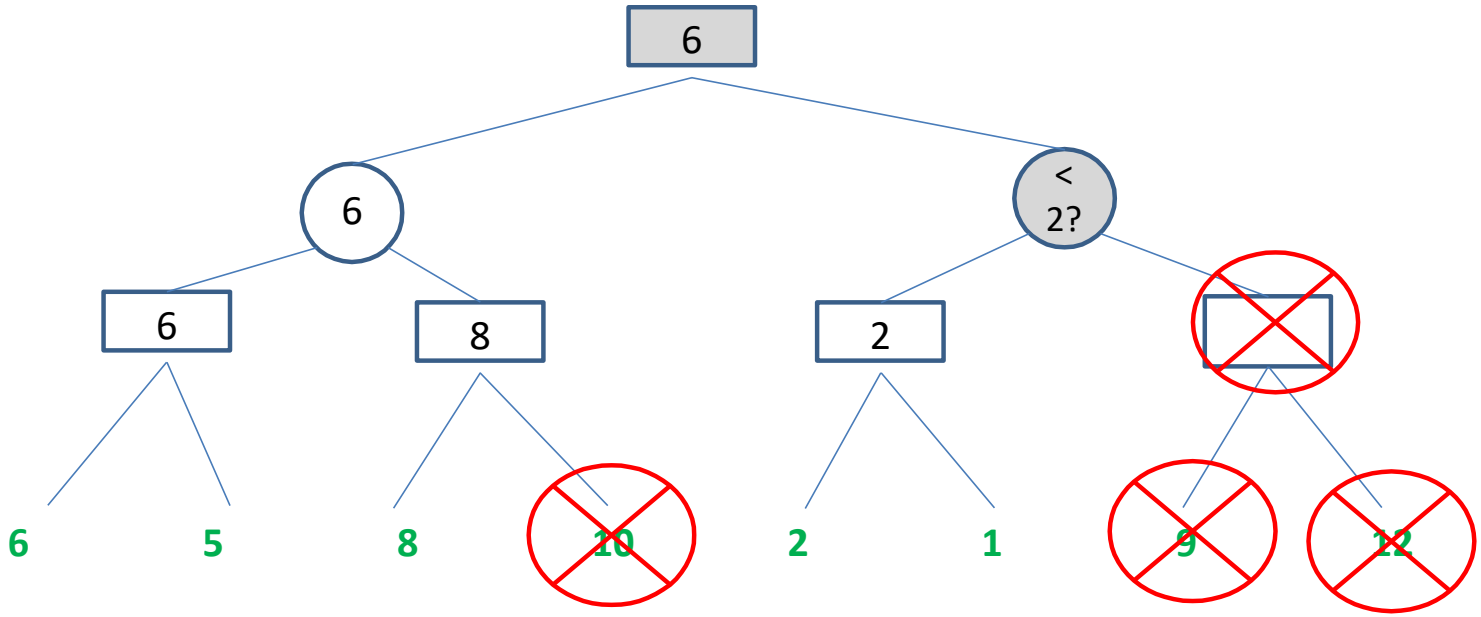


Alpha Beta Pruning

Idea –Pruning



Idea – Alpha Pruning



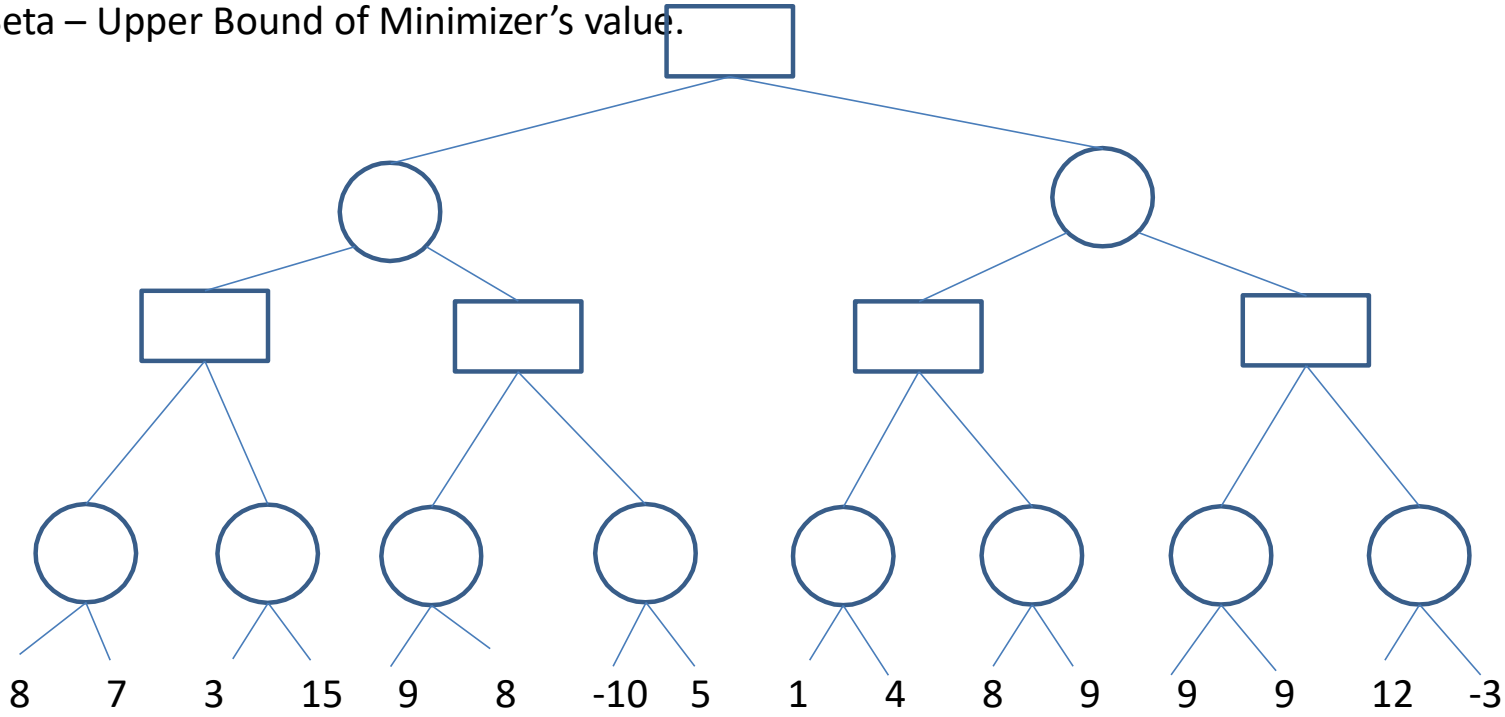
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to against a competitive Minimizer

Alpha – beta Pruning – Example -4

Do for practice.

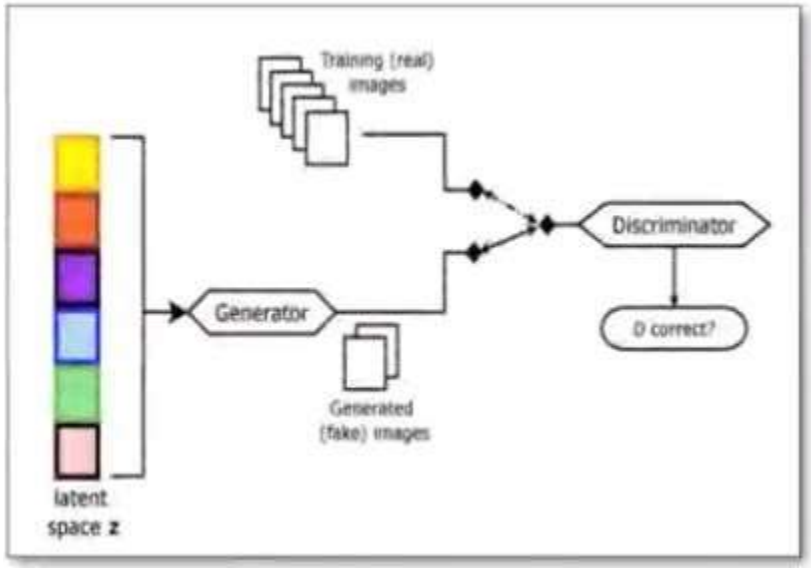
Alpha – Lower bound of Maximizer's value. Perceived value that Maximizer hopes to get with a competitive Minimizer

Beta – Upper Bound of Minimizer's value.



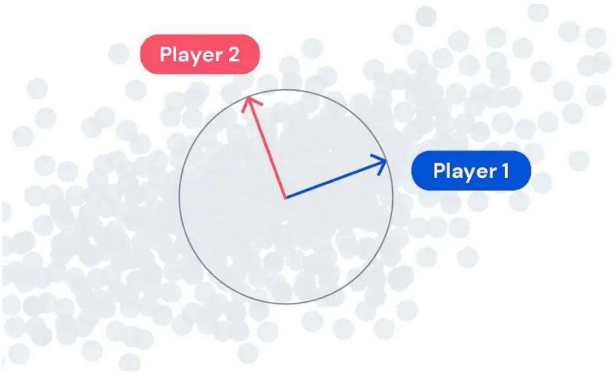
Game Playing
(Interesting Case Studies)

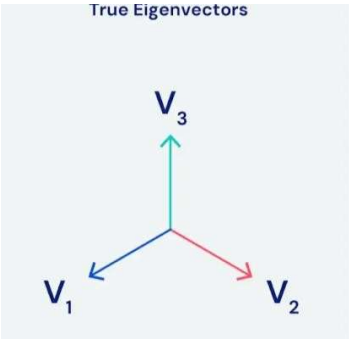
Games in Image Processing

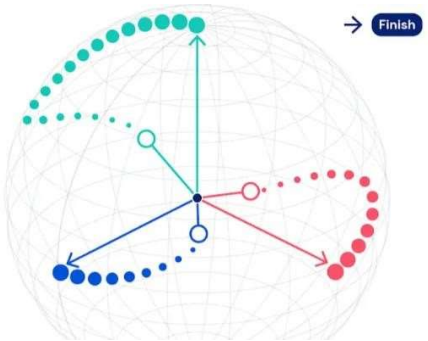


Source Credit:
[2019 - Analyzing and Improving the Image Quality of StyleGAN]
[Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila]
[https://thispersondoesnotexist.com/]

Games in Feature Engineering







Source Credit:
https://deepmind.com/blog/article/EigenGame
2021 - EigenGame: PCA as a Nash Equilibrium , Ian Gemp, Brian McWilliams, Claire Vernade, Thore Graepel

Games in Feature Engineering

$$\mathtt{Utility}(v_i \mid v_{j<i}) = \boxed{\mathtt{Var}(v_i)} - \sum_{j<i} \boxed{\mathtt{Align}}(v_i, v_j)$$

Source Credit:
https://deepmind.com/blog/article/EigenGame
2021 - EigenGame: PCA as a Nash Equilibrium , Ian Gemp, Brian McWilliams, Claire Vernade, Thore Graepel

Gaming – Sample Question

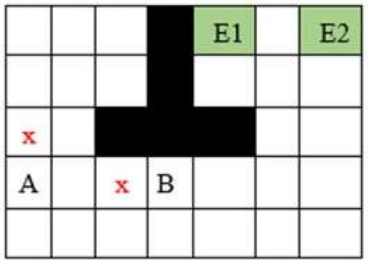
Two Robots A and B competes to leave the maze through either of exits: E1 and E2, as shown in the diagram. At each time step, each Robot move to an adjacent free square. Robots are not allowed to enter squares that other robots are moving into. The same exit cannot be used by both robots at once, but either robot may use either exit. A poisonous gas is left behind when a robot moves. No robot may enter the poisonous square for the duration of the gas's 1-time-step presence (ie., If the square is left free for one game round, the poison evaporates and is no longer dangerous). The poisonous squares are represented as ×'s in the diagram. For utility calculation consider the below assumptions.

H(n) = (Max player's ability to win in the board position) – (Min player's ability to win in the board position)

Note: *Player "Z's" ability to win is given by below:*

Utility = Minimum (manhanttanDistance(Player "Z", Exit E1), manhanttanDistance(Player "Z", Exit E2)) + Penalty

Penalty = Number of unsafe cells (blockage+ poisonous squares) with 4 degree of freedom(Up, Down, Right, Left) in the immediate neighborhood of the Player "Z's" position.

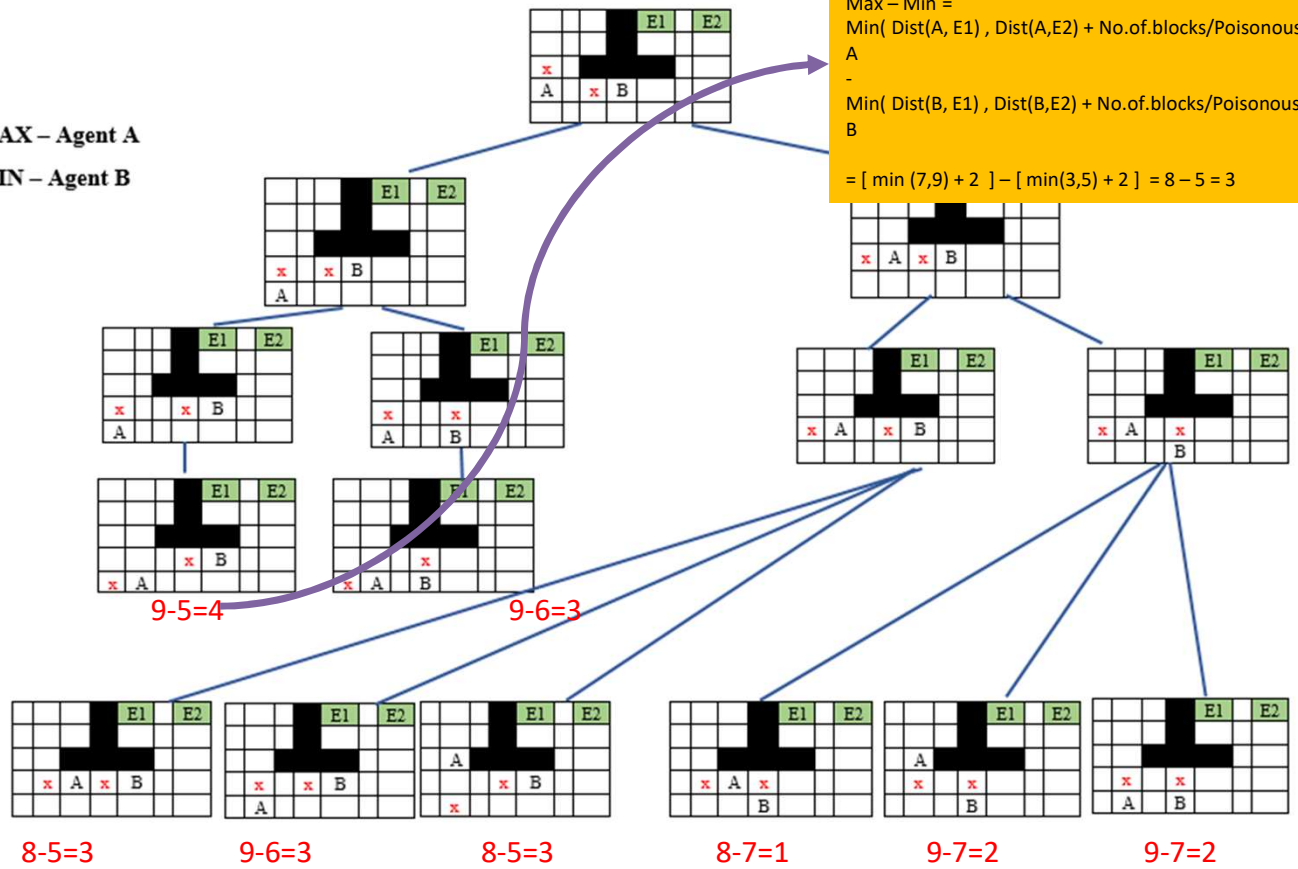


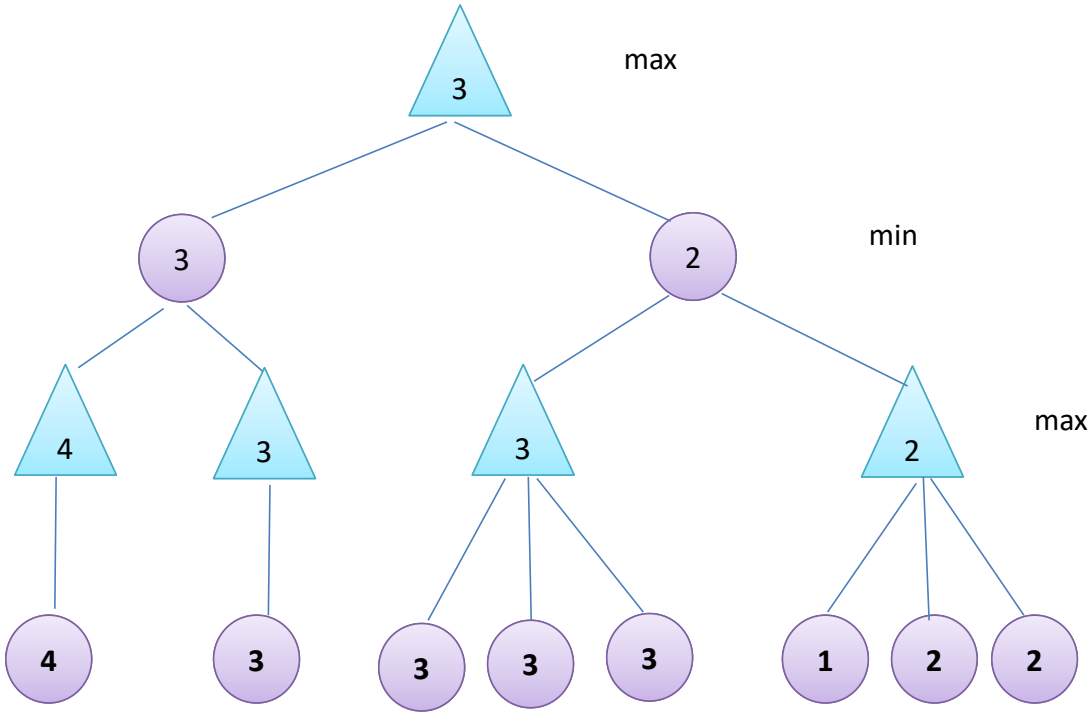
Max – Min =
Min(Dist(A, E1) , Dist(A,E2) + No.of.blocks/Poisonous cells around A
-
Min(Dist(B, E1) , Dist(B,E2) + No.of.blocks/Poisonous cells around B

= [min (7,9) + 2] – [min(3,5) + 2] = 8 – 5 = 3

MAX – Agent A

MIN – Agent B




max
3
min
3
2
max
4
3
3
2
4
3
3
3
3
1
2
2

Required Reading: AIMA - Chapter # 5.1, 5.2, 5.3

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials

Birla Institute of Technology & Science, Pilani
Work-Integrated Learning Programmes Division
Second Semester 2018-19
Mid Semester Examination(Makeup)

Course No. : DSECL ZG557
Course Title : Artificial & Computational Intelligence
Nature of Exam : Closed Book
Weightage : 30%
Duration : 1.5 Hours
Date of Exam : 05 / 01 / 2020, AN

No. of Pages = 4
No. of Questions = 4

Time of Exam: 2:00 PM to 3:30 PM

Note:

1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. ***Questions must be answered only in the page numbers mentioned against each question. In case you require more pages to answer a question, you can continue from page #21***
4. Assumptions made if any, should be stated clearly at the beginning of your answer.

Answer all the questions

Question -1 [1.5 + 1.5 + 1.5 + 2 +1.5 = 8 M] **[Page 03 - 09]**

(1-a) Provide one point for and against Turing Test to determine if a software piece is intelligent

For questions (1-b) and (1-c), answer if forward search (data driven search) or backward search (goal driven search) will be more appropriate with one good reason for the scenarios given:

(1-b) You have met a person Dr. Rajiv who claims to be your distant cousin. You need to verify his claim by finding some common ancestor who is a few generations older.

(1-c) A system that will help human to classify plants by species

For questions (1-d) and (1-e) consider the following scenario:

A farmer with his wolf, goat, and cabbage come to the edge of a river they wish to cross. There is a boat at the river's edge, but, of course, only the farmer can row. The boat also can carry only two things (including the rower) at a time. If the wolf is ever left alone with the goat, the wolf will eat the goat; similarly, if the goat is left alone with the cabbage, the goat will eat the cabbage. Devise a sequence of crossings of the river so that all four characters arrive safely on the other side of the river.

(1-d) Devise a concise representation of a state (considering all possible uses of this in search algorithms taught to you). As per the representation devised provide write down the representation of a start state and the goal state.

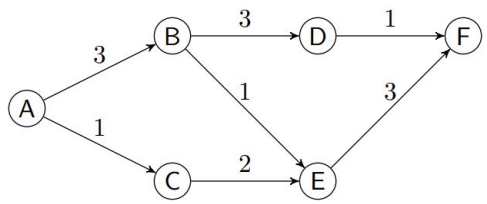
Provide the data structure description of a state. Specify the fields (with details of what goes into), pointers to be kept in the node. Answers both in pseudo code, pictorial rep, in description are all allowed. [1.5 M]
State diagram, solution to the problem are not expected here

(1-e) Which of DFS or BFS is most preferred for this problem? Why.

DFS is recommended. This will lead to the lease number of nodes being expanded. This will become obvious when the state diagram is drawn.

Question -2 [4 + 2 + 1 = 7 M] **[Page 10 - 15]**

(2-a) Consider the following state space and the heuristic function associated. The initial state is A and the goal state is F.



Show the result of Uniform cost search algorithm on this search tree.

(2-b) Is the uniform cost search (as presented in class and Text Book) a tree search or a graph search? Explain why should it be this way?

(2-c) When the heuristic function h(n) $\geq \boldsymbol{\varepsilon}$ (where $\boldsymbol{\varepsilon}$ is a small positive constant) the states expanded by A* algorithm will be a subset of uniform cost search. Why or why not?

Question -3 [3+2+ 3 +2 + 1 = 11 M] **[Page 16 - 22]**

The *sliding-tile puzzle* consists of three black tiles, three white tiles, and an empty space in some order. The goal is to arrive at the goal configuration shown below by legal moves [B- Black tile, W-White tile].

B	B	B		W	W	W

The puzzle has two legal moves with associated costs

Move #1: A tile may move into an adjacent empty location. This has a cost of 1.

Move #2: A tile can hopover one or two other tiles into the empty position. This has a cost equal to the number of tiles jumped over

(2-a) What are measures using which the complexity of a state space can be measured. List each of them and provide the corresponding expression / value for this problem.

Listing of space, brancing ,- 1Mark, Corresponding values for this problem - 2M

(2-b) Write down an admissible heuristic function h(n) for this problem with a formal description. Explain why this proposed heuristic is admissible.

1Mark for heuristic function and 1Mark for explaining

h(n) = # of white tiles on the wrong side of each black tile

- Breadth-first search is admissible, monotonic and very uninformed.
- Counting tiles out of the goal position is a simple useful heuristic that is admissible.
- A form of this heuristic would be to count all white tiles on the wrong side of each black tile.
- For example for the following configuration, there are 3 Ws on the right side of 1st B, and 2 Ws on the right side of both the 2nd and 3rd Bs.
- Therefore h(n) = 3+2+2 = 7 B_WBBWW
- Counting tiles out of place is not monotonic, but is more informed than breadth-first search. An easy way to prove that a heuristic is non-monotonic is to watch its value change as it goes along a search path.

(2-c) Generate the A* search tree (using tree search) starting with the following start state for depth up to 3 or until the goal state is reached, whichever is earlier.

W	B	W	B	B	W	

3Marks for : Annotate each node and edges with appropriate values f(n), h(n) and g(n) for better understanding. A search tree without such annotations will not get full credit here.

(2-d) Argue formally the optimality of the solution obtained.
Valid argument - 2M, Good attempt in right direction - 1M
(2-e) Will the solution obtained be different if the heuristic is also consistent in addition to being admissible. Why or why not?
0.5 , 0.5 for reasoning.

Question -4 [4M] **[<u>Page 22 - end</u>]**
You have been asked to solve below linear equation problem with multiple variables:

$$2a + 7b - 5c + d = 0$$

where a, b,c, and d are integers in the range [-20,20] (equation can have more than one solution)
You have decided to solve this question using genetic algorithm. Show all the steps involved in solving this using Genetic Algorithm. Number each of your steps, and provide appropriate title to those steps. (No need to get the final solution, just the first iteration of each step involved in the algorithm)

Most likely representation of chromosome could be a list with 4 integers with each integer representing variables a,b,c or d. For example [3,2,3,1] would imply a=3,b=2,c=3,d=1.

Generate random population (list of lists)

Fitness function - use a fitness function (most likely f(x) = 2a + 7b -5c +d) and calculate the fitness of each individual

Apply some selection criteria based on fitness value (could be roulette or the best fit etc.) to come up with initial population

Apply crossover and mutation (how students chose to incorporate crossover and mutation may differ but it should be consistent)