Artificial & Computational Intelligence

AIML CLZG557

M1 : Introduction
&
M2 : Problem Solving Agent using Search

Indumathi V

Guest Faculty,

BITS - WILP

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

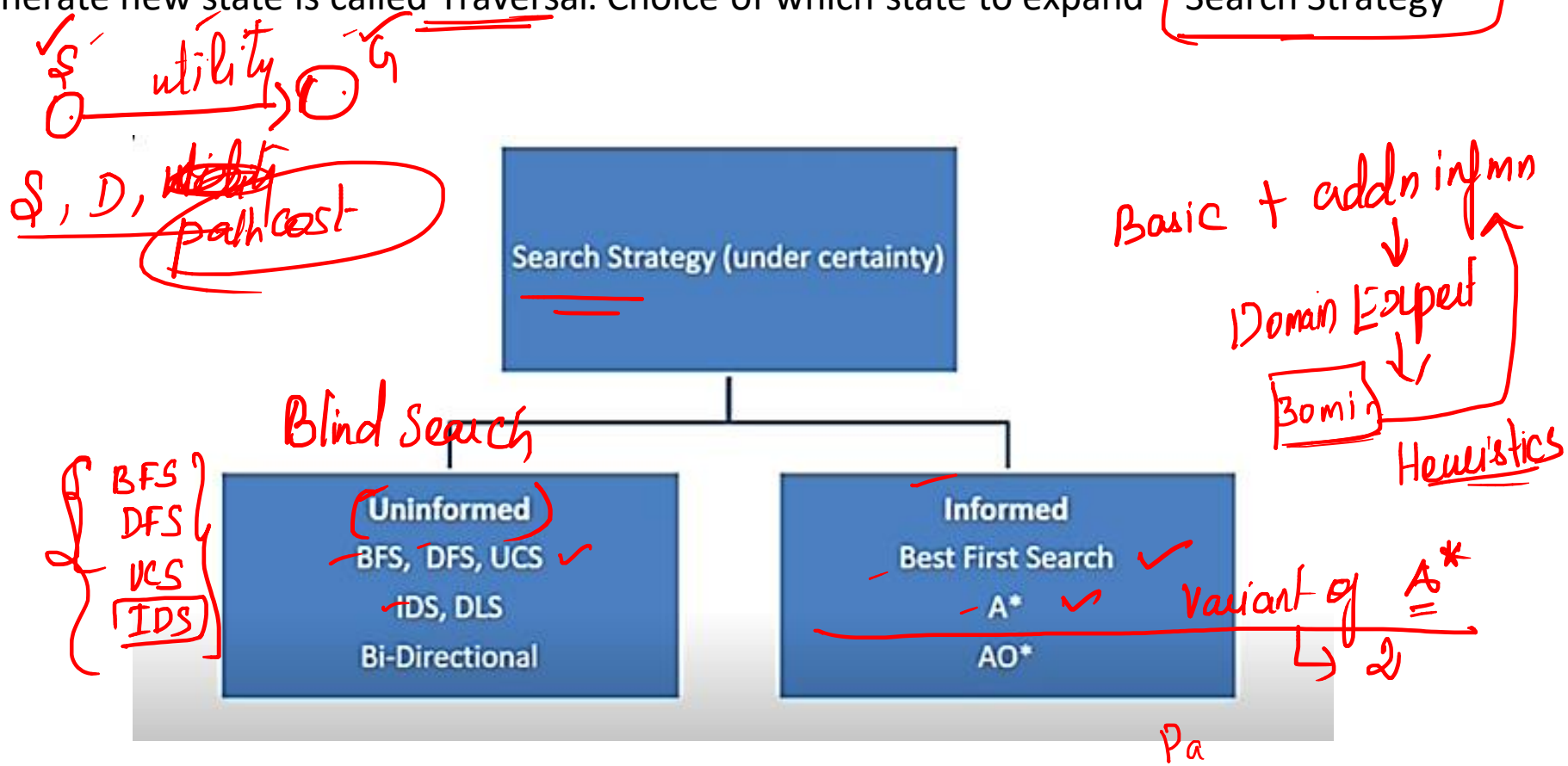
Learning Objective

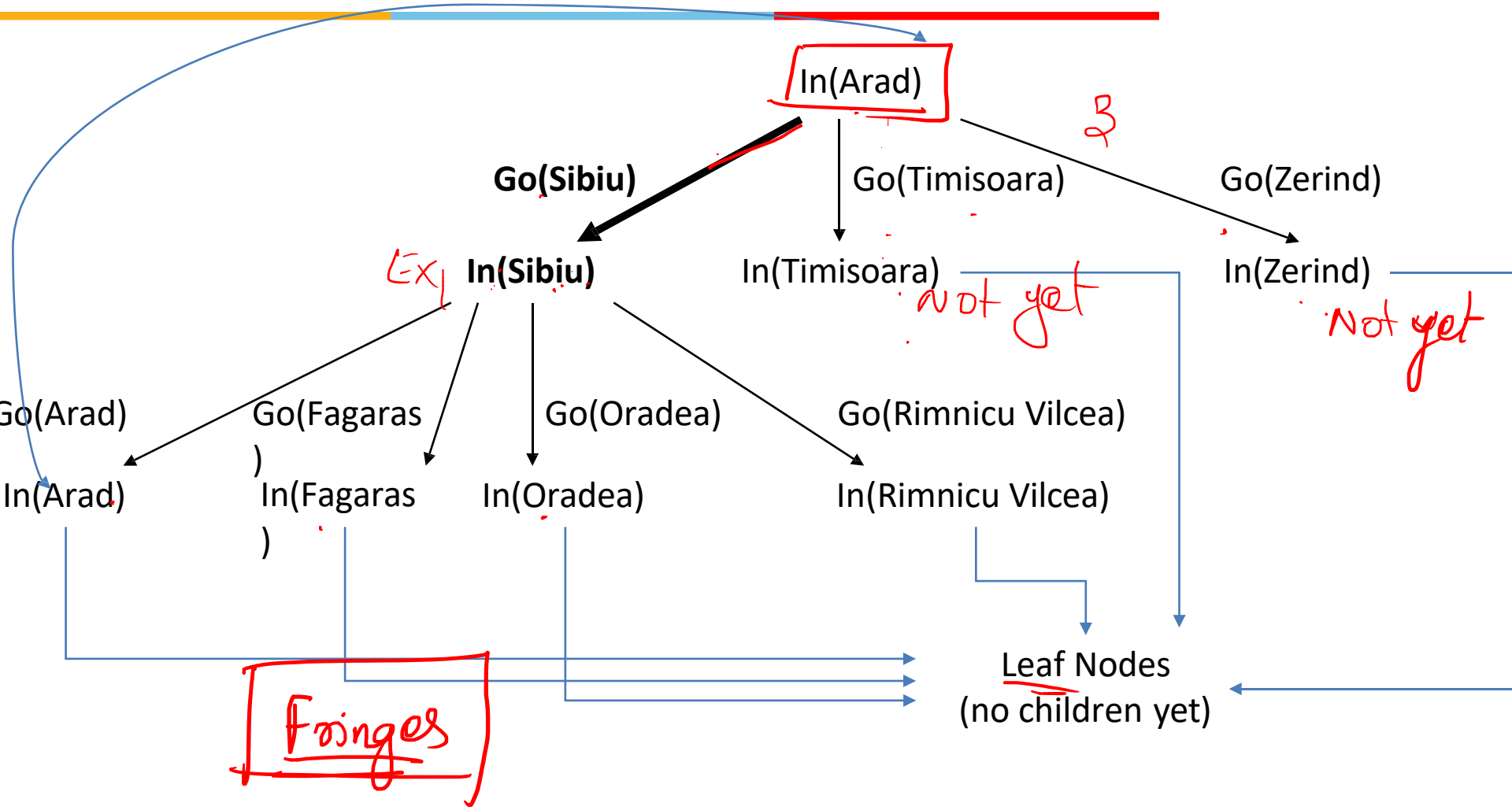
At the end of this class , students Should be able to:

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

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)
Ex
In(Sibiu)
In(Timisoara)
Not yet
In(Zerind)
Not yet
Go(Arad)
Go(Fagaras)
Go(Oradea)
Go(Rimnicu Vilcea)
In(Arad)
In(Fagaras)
In(Oradea)
In(Rimnicu Vilcea)
Leaf Nodes
(no children yet)
Fringes

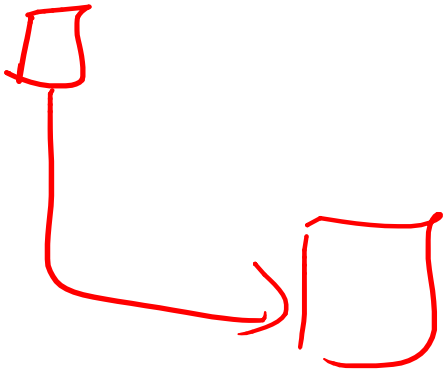
Application

Breadth First Search

- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph

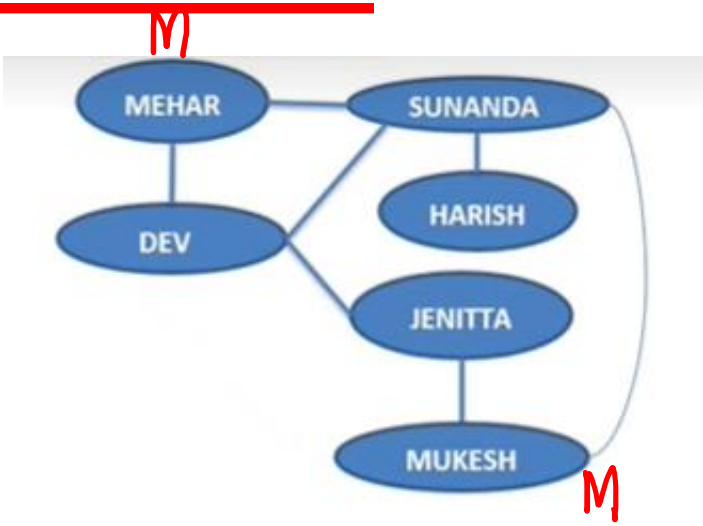
Depth First Search

- Find the Connectedness in a graph
- Topological Sorting



Application

Breadth First Search



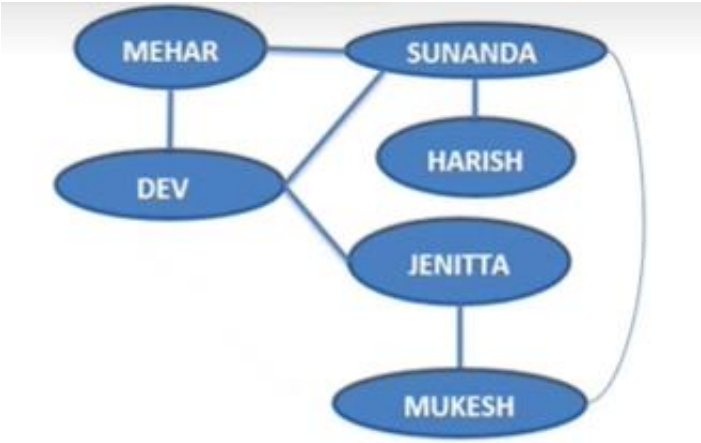
- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph

Depth First Search

- Find the Connectedness in a graph
- Topological Sorting

Application

Breadth First Search



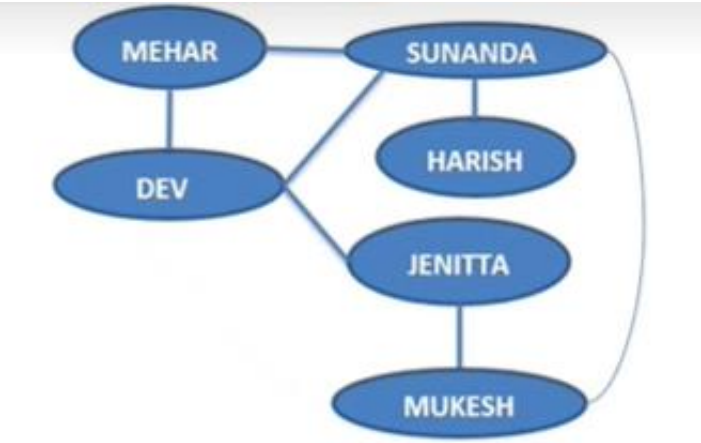
- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph

Depth First Search

- Find the Connectedness in a graph

1st level friend
2nd fr

- Topological Sorting



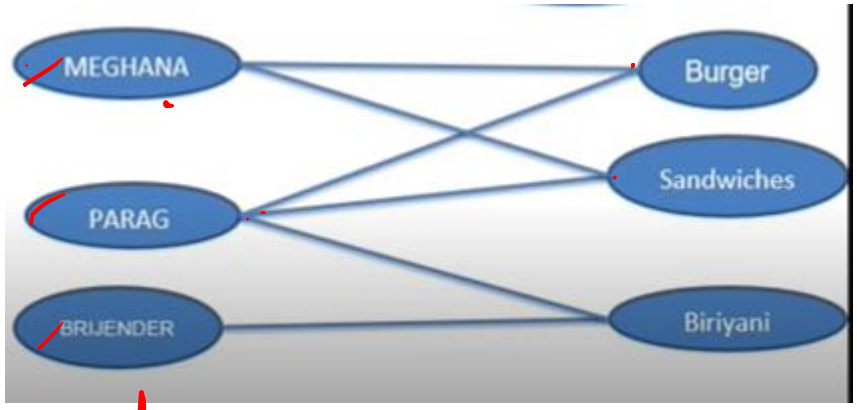
Application

Set 1

Names

Food Name

Breadth First Search



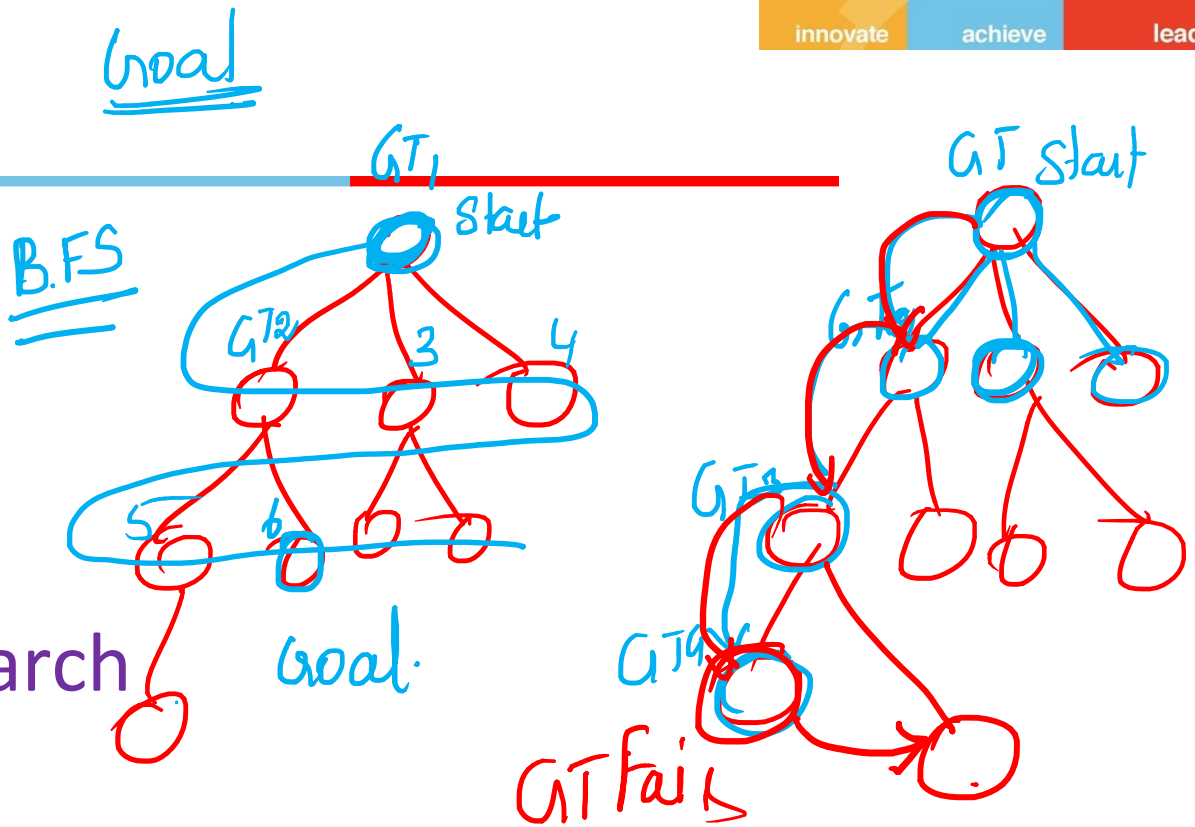
- Finding path in a graph (many solutions)
- Finding the Bipartitions in a graph

Recommend

Depth First Search

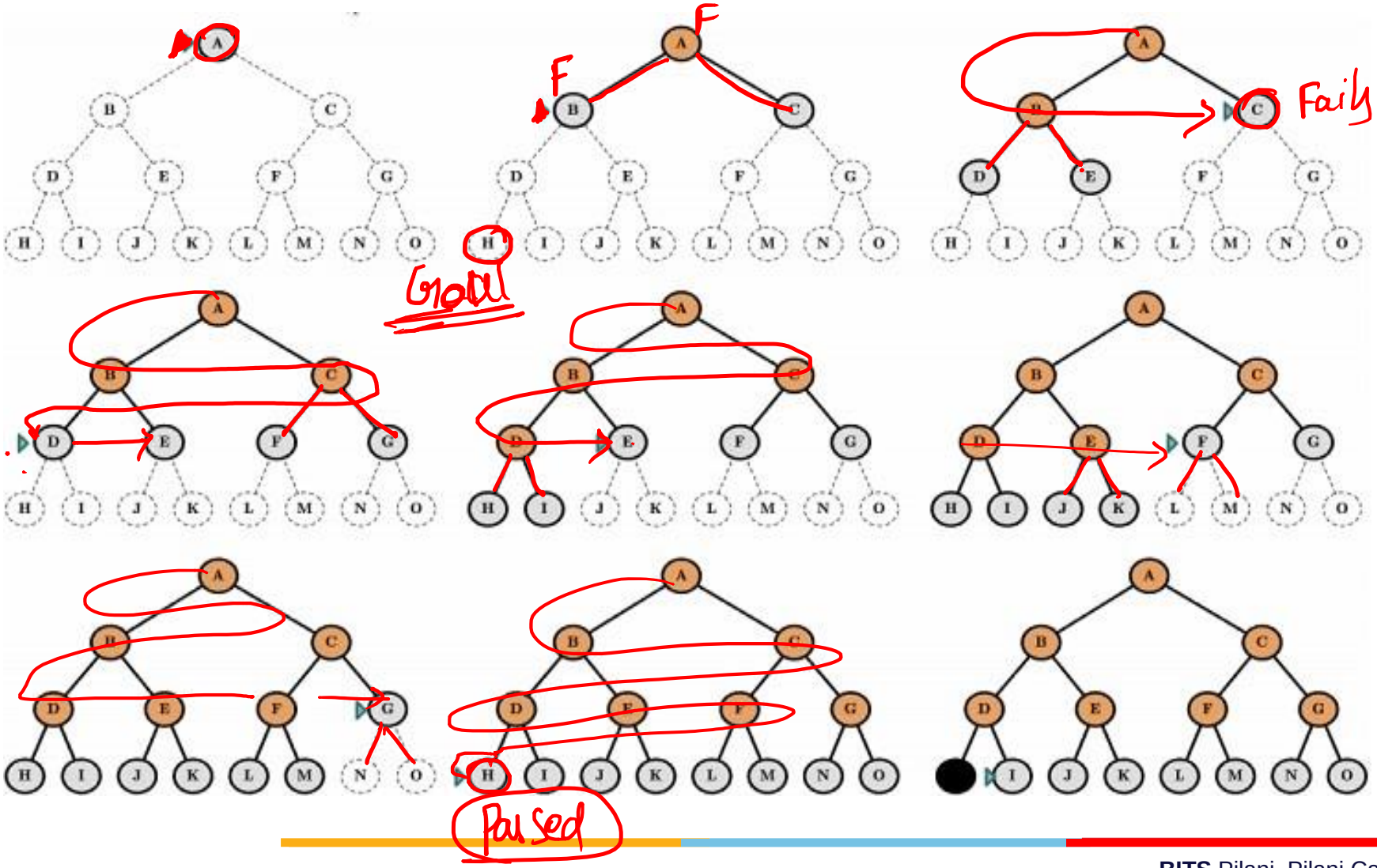
- Find the Connectedness in a graph
- Topological Sorting

Uninformed Search Overview



Breadth First Search (BFS)

① GTest : Fails
① Expand Successors. Child
levelwise



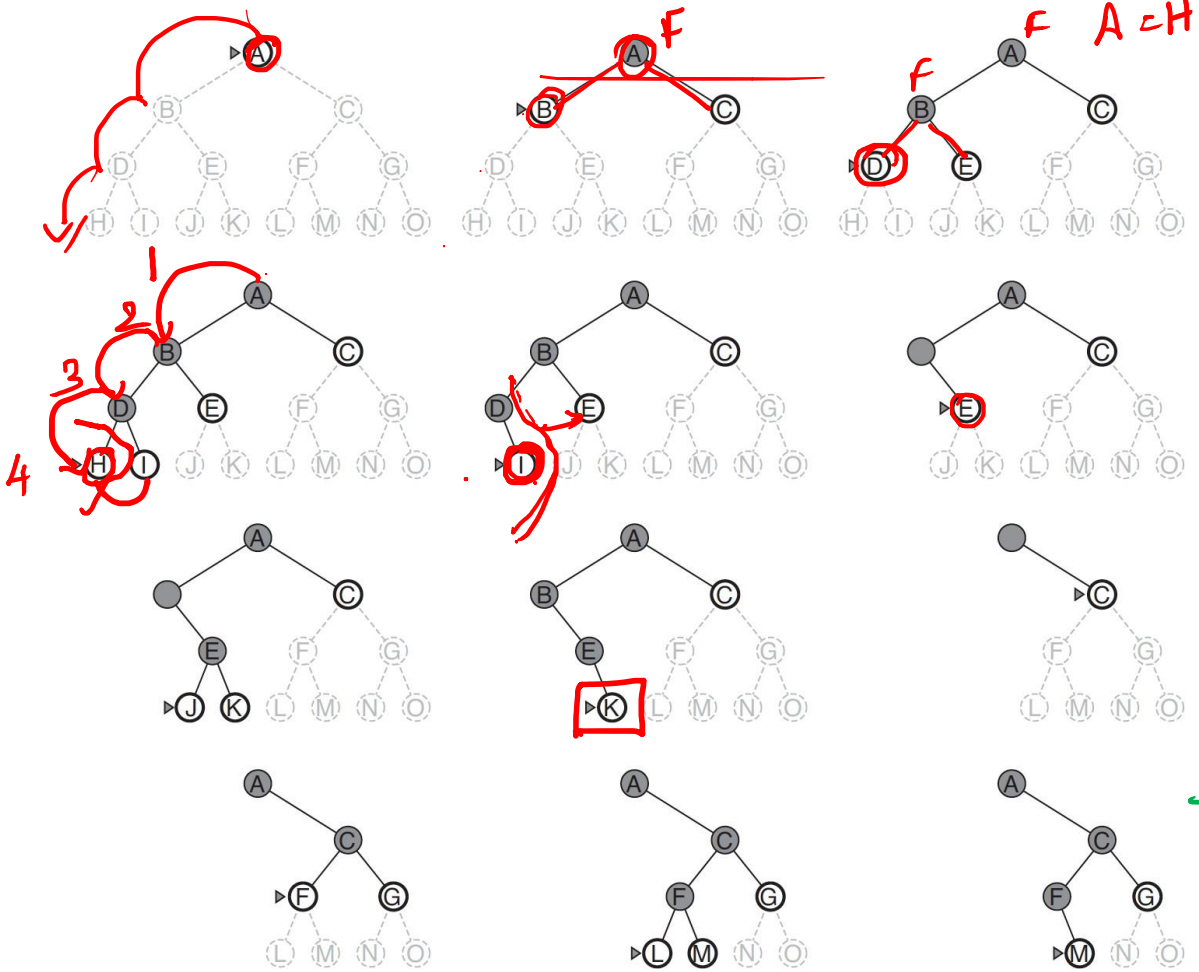
Depth First Search (DFS)

Start-start : A

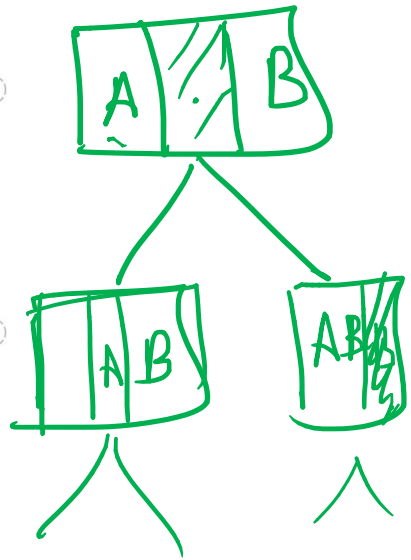
Goal : H

DFS

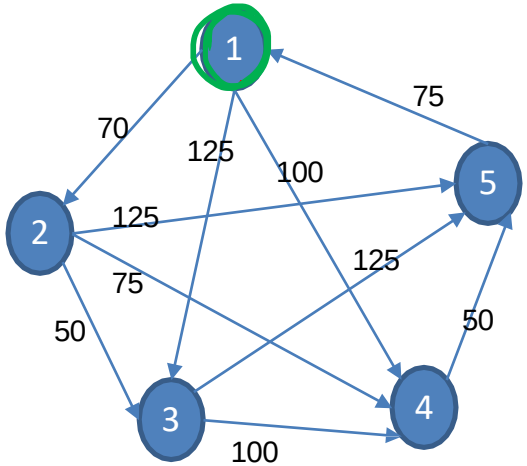
Memory efficient

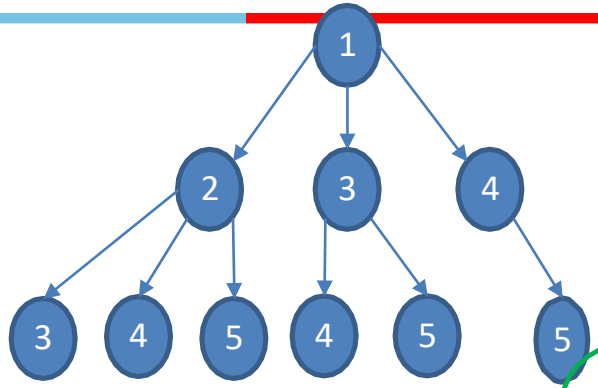


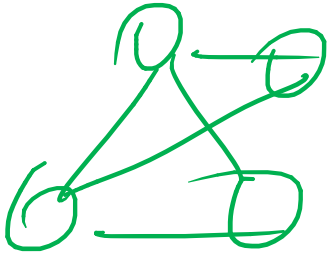
So



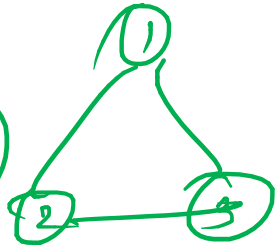
Search Tree – Sample Generation



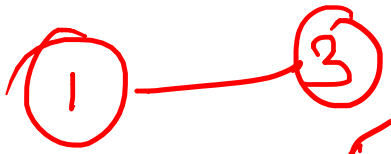




Path



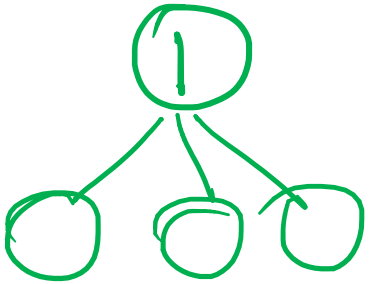
Each ~~NODE~~ in in the search tree denotes an entire PATH through the state space graph.



1 to 3

1-3 → Node

1-2-3 →



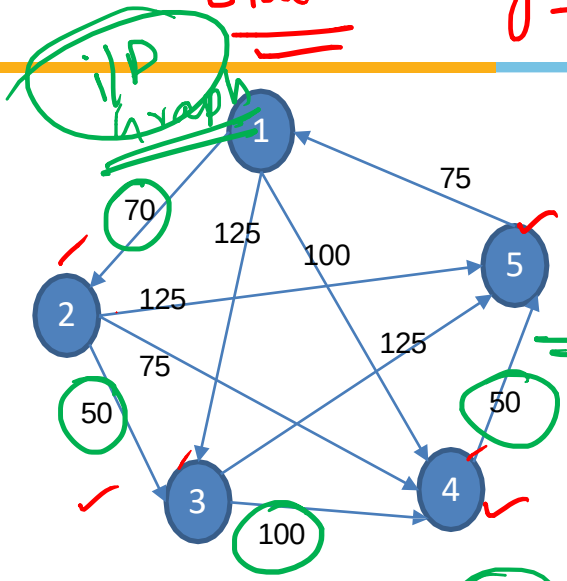
1 — 2,3

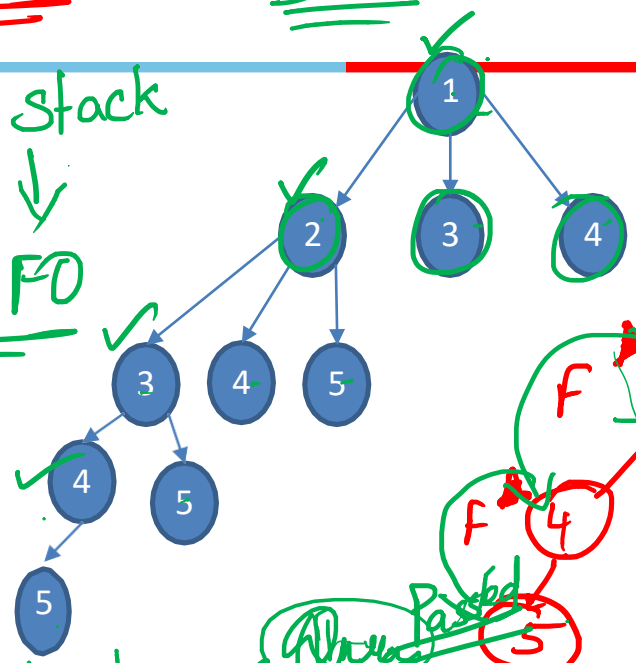
2 → 1,3

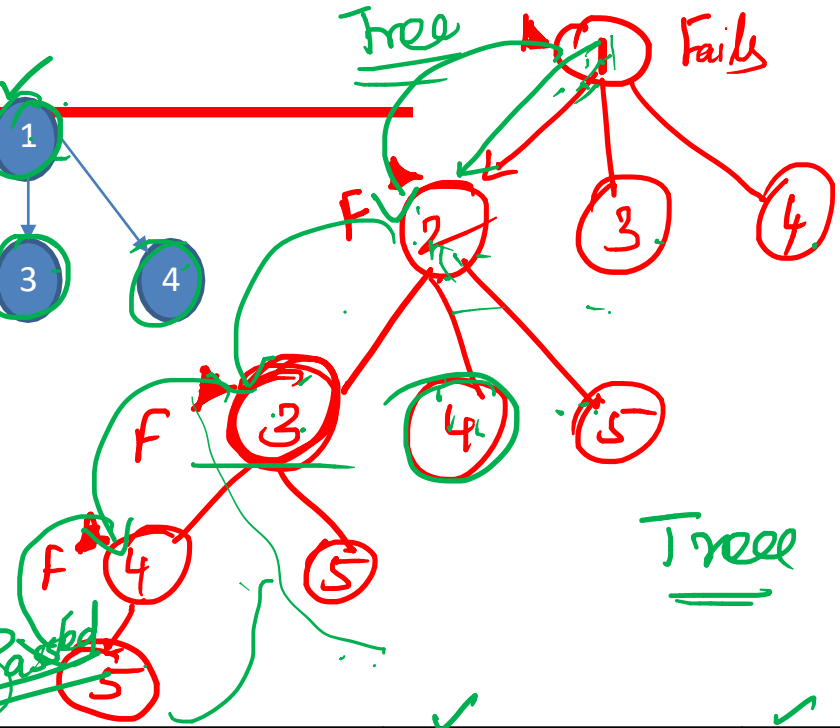
3 — 1,2

Search Algorithm – Uninformed Example - 1

Start : 1 goal : 5







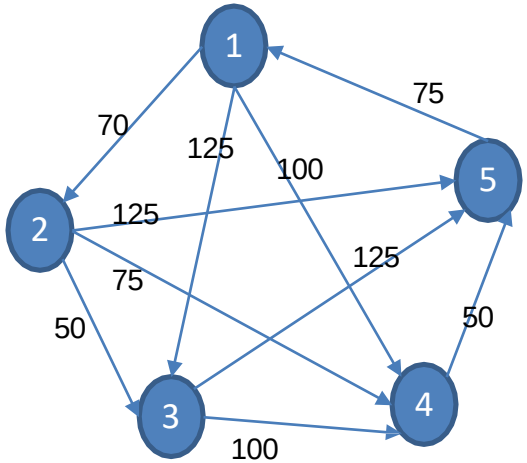
(1)
(1 2) (1 3) (1 4)
(1 2 3) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)
(1 2 3 4 5) (1 2 3 5) (1 2 4) (1 2 5) (1 3) (1 4)

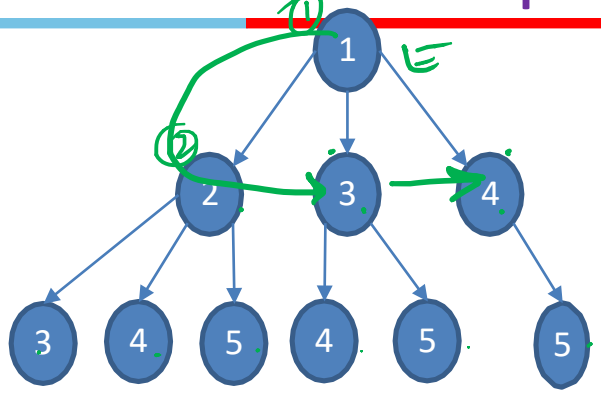
C(1-2-3-4-5) = 70 + 50 + 100 + 50 = 270
Expanded : 4
Generated : 10
Max Queue Length : 6

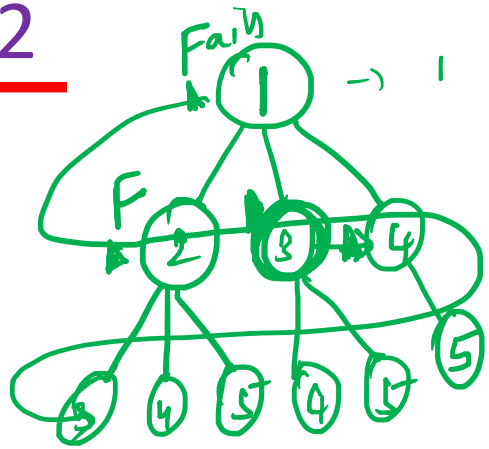
Iter	Open List / Frontiers / Fringes	Closed List	Goal Tes
1.	(1)		Fail on (1)
2.	(12) (13) (14)	(1)	Fail(12)
	(123) (124) (125) (13) (14)	(1) (12)	F
	(1234) (1235)	(1) (12) (123)	F
	(12345)	(1234)	P

BFS

Search Algorithm – Uninformed Example - 2







BFS

Queue

FIFO

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

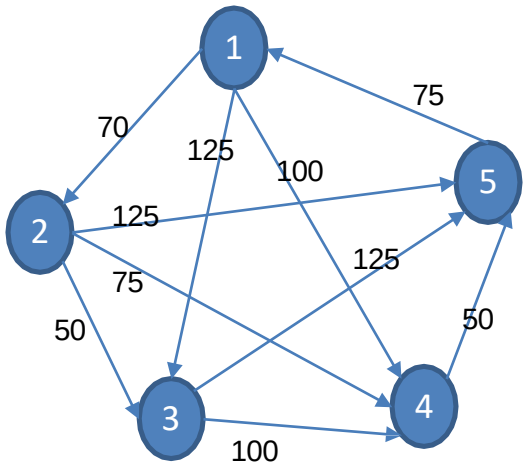
(1 3) (1 4) (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) (1 4 5)

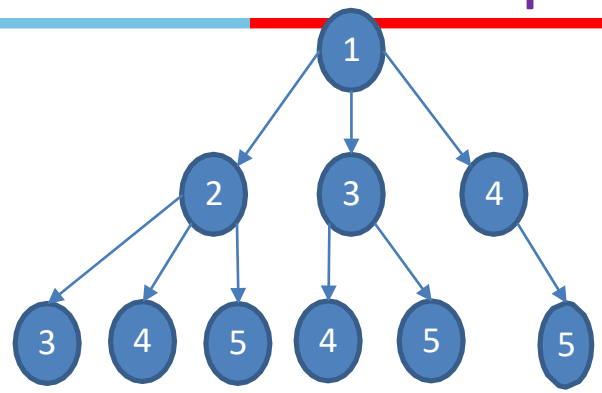
open	closed	Goal
(1)		
(12)(13)(14)		
(13)(14)	(12)	

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

BFS

Search Algorithm – Uninformed Example - 2





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

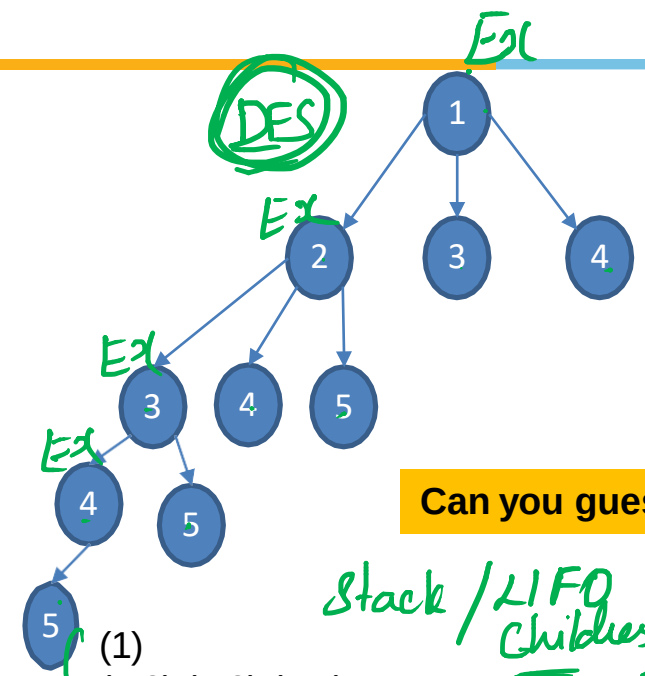
(1 3) (1 4) (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) (1 4 5)

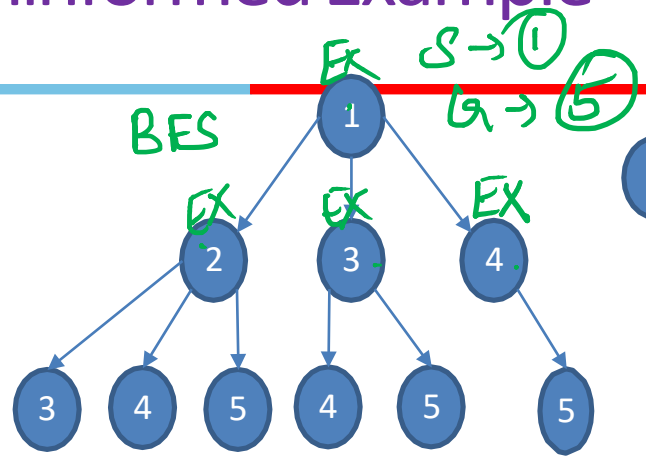
Iter	Open List / Frontiers / Fringes	Closed List	Goal Test
1.	(1)		Fail on (1)
2.			
3.			

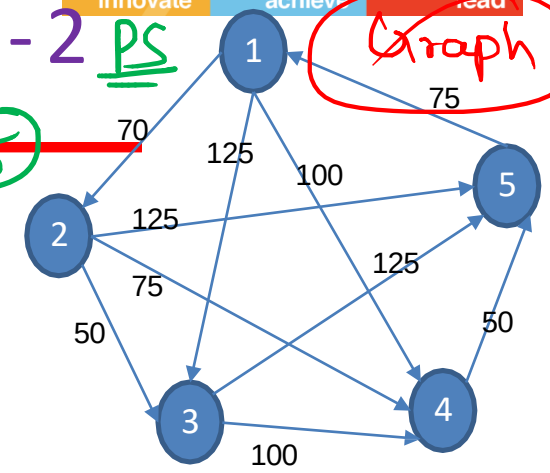
125

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 ?

Stack / LIFO
Children added top of stack

(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 = $\boxed{270}$
Expanded : 4
Generated : 10
Max Queue Length : 6

Queue / FIFO
Children are add at end of Queue

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

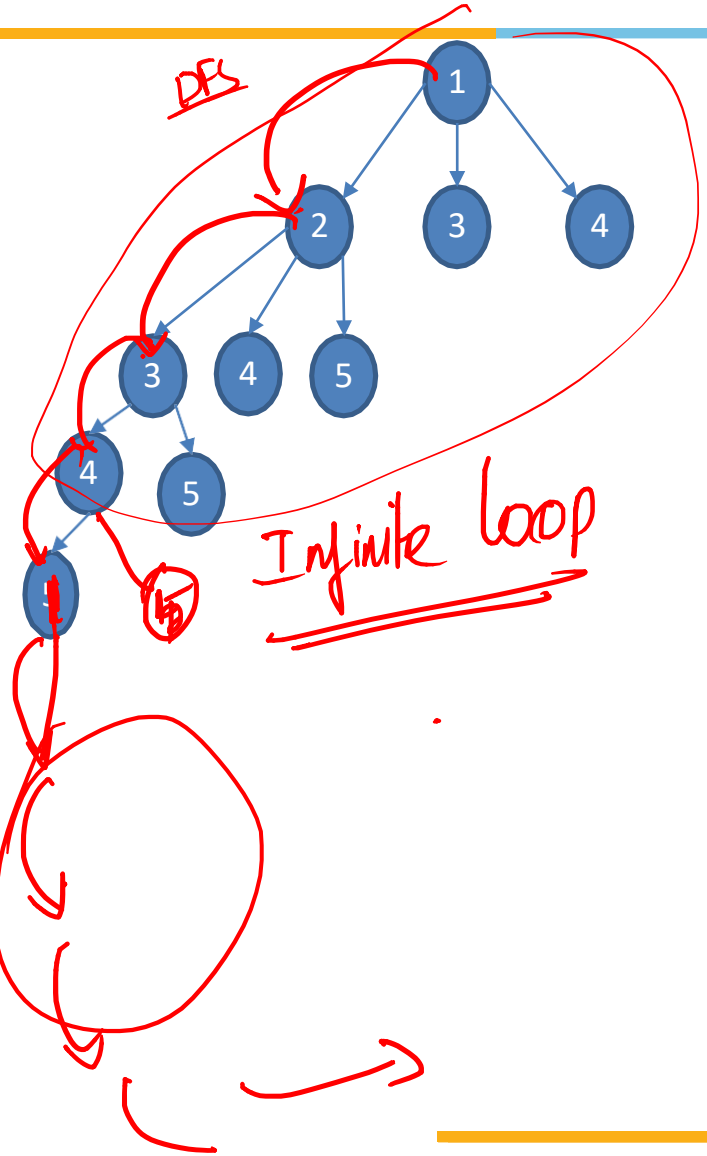
(1 3) (1 4) (1 2 3) (1 2 4) (1 2 5)
(1 2 3) (1 2 4) (1 2 5) (1 3 4) (1 3 5) (1 4 5)
TEST PASSED

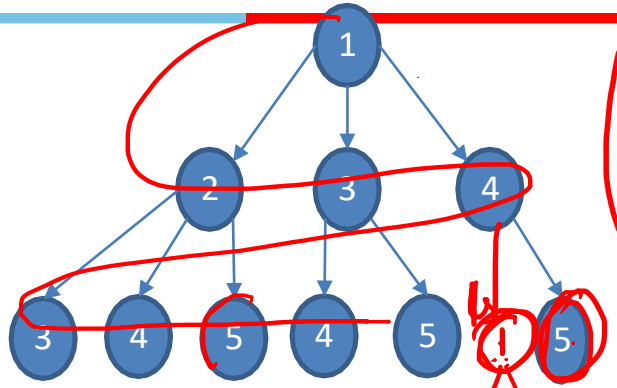
C(1-2-5) = 70 + 125 = $\boxed{195}$
Expanded : 4
Generated : 10
Max Queue Length : 6

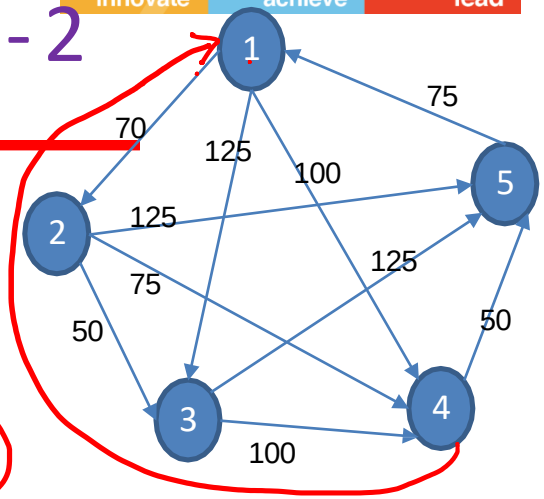
(1 4 5) ⇒ 150

Variant of BFS $\boxed{\text{UCS}}$

Search Algorithm – Uninformed Example - 2





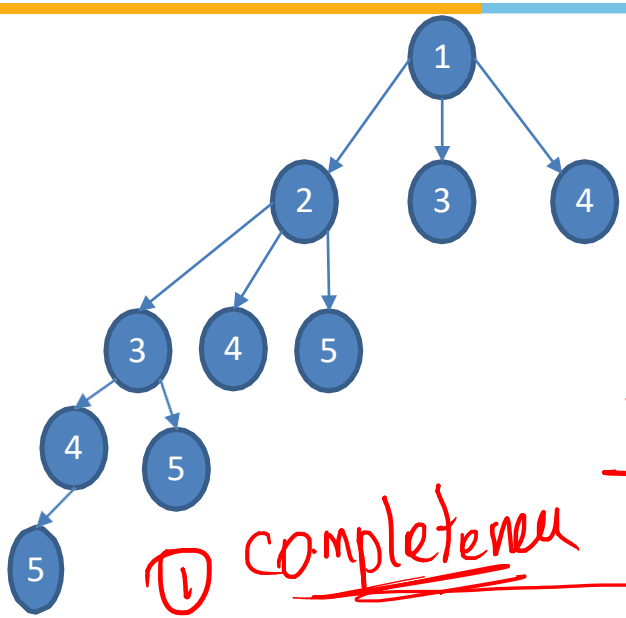


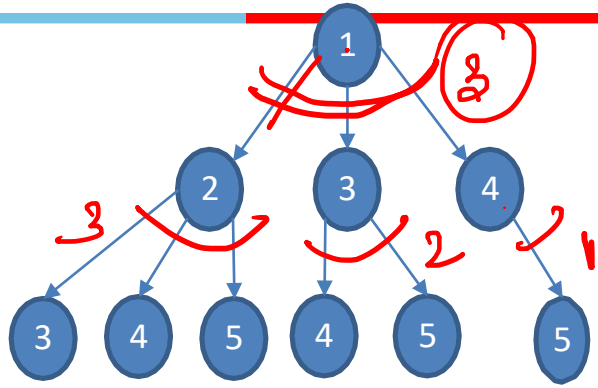
BFS

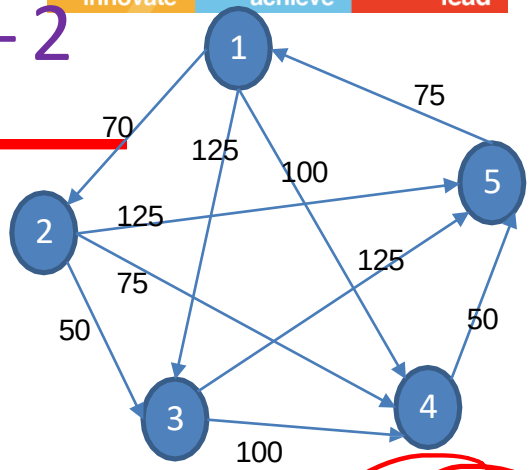
if bf = finite → complete

else → not complete

Search Algorithm – Uninformed Example - 2



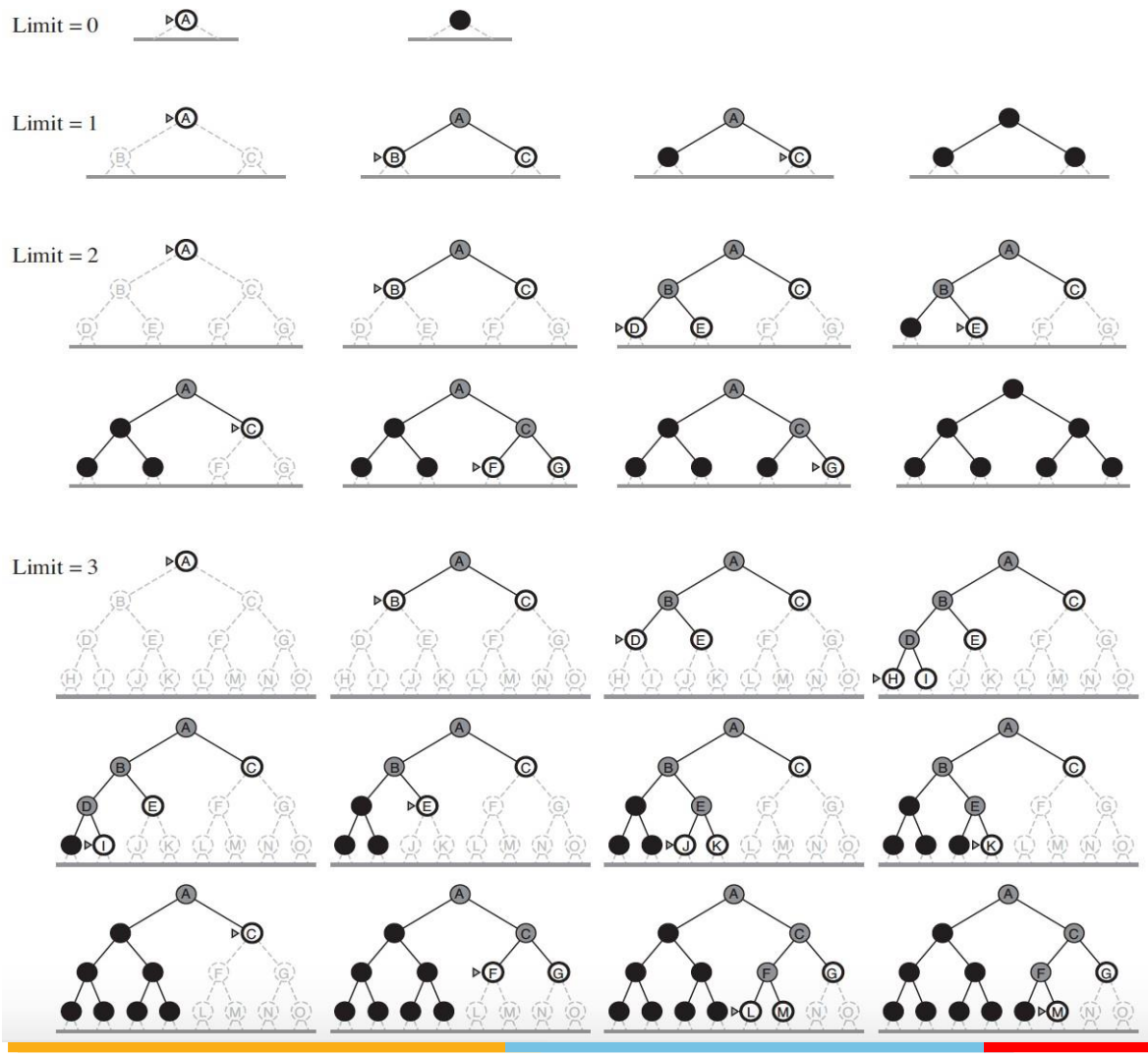




1-4-5

	DFS	BFS
1 completeness	Not comp	if branching factor finite → optimal else " " " not " → No optimal
2 optimality	Not opm	1-4-5 can be optimal
3 Time Complexity 4 Space complexity		

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.			

Variant BFS

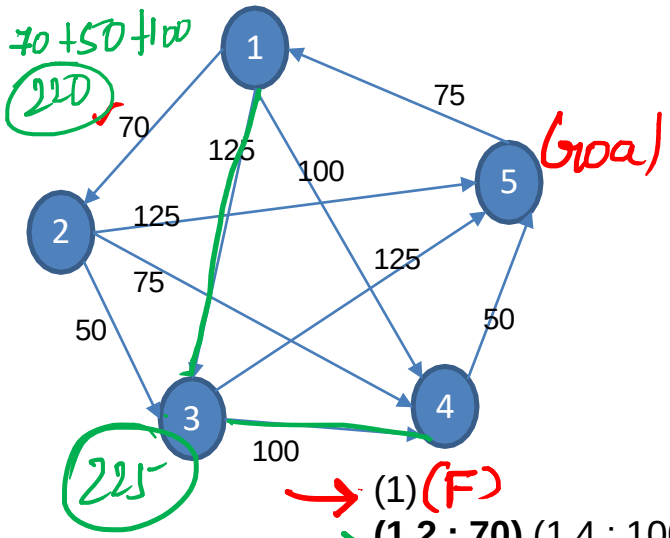
UCS → Uniform Cost Search

Priority Queue

Start

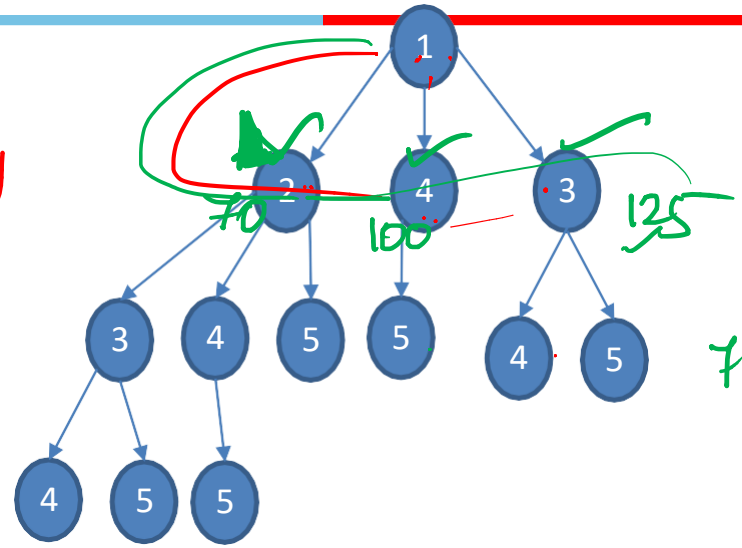
70+50+100

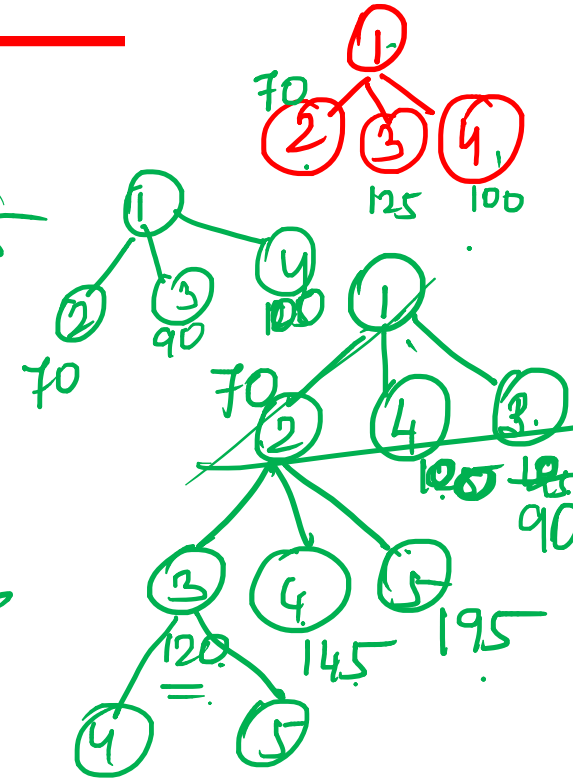
220



Goal

225





(1) (F)

(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

FIFO

Passed

Uniform Cost Search

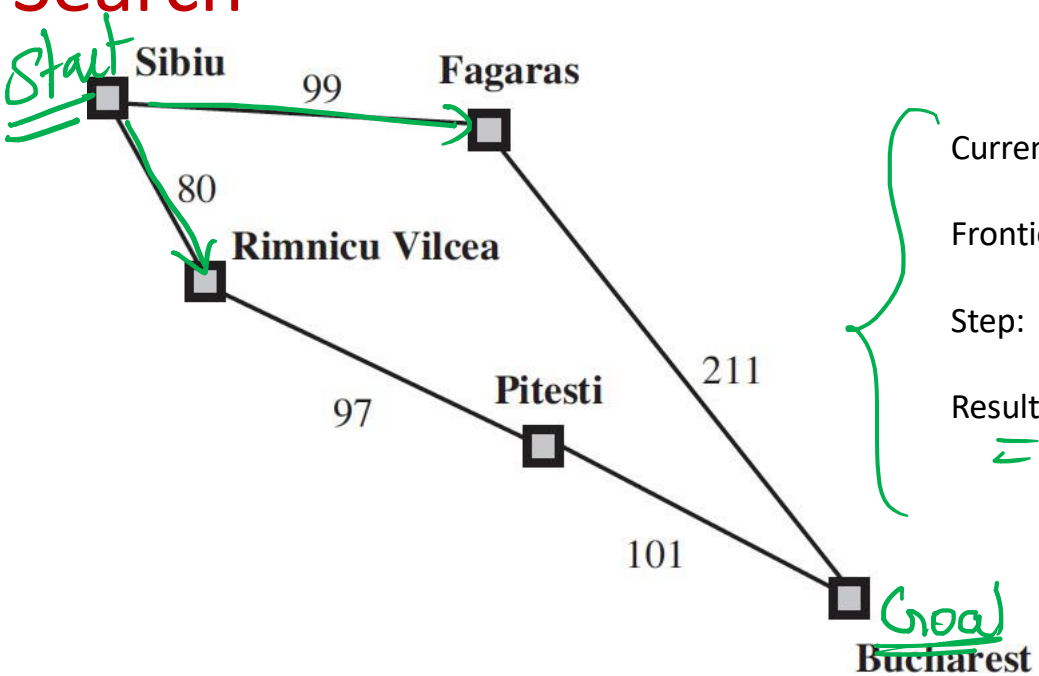
Instead of expanding the shallowest node, Uniform-Cost search expands the node n with the lowest path cost g(n)

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 G → Fails

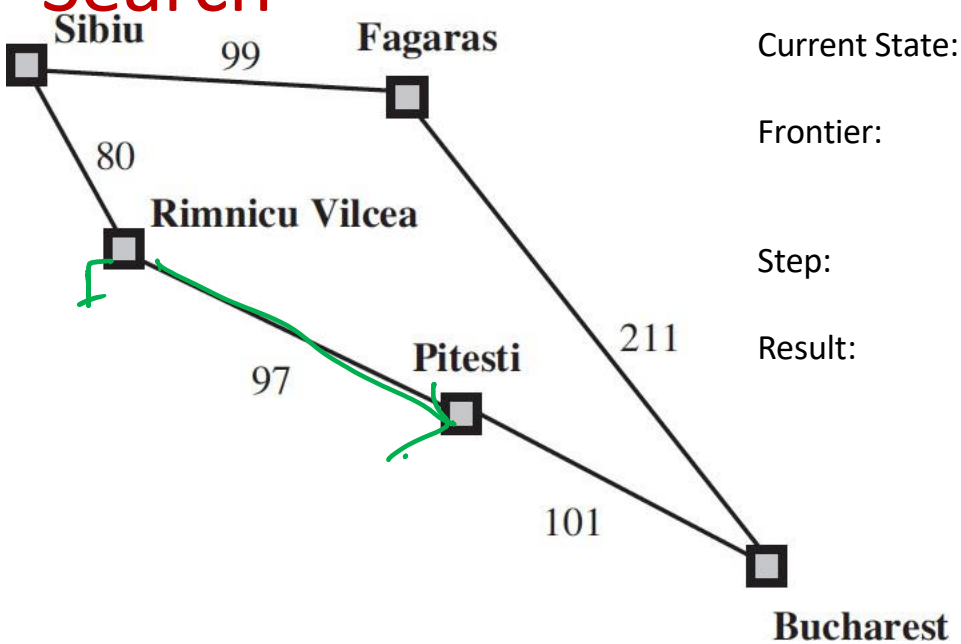
Frontier: []

Step: Expand Sibiu

Result: Generates ("Rimnicu Vilcea" 80)
("Fagaras", 99)
Add to Frontier

Initial State: Sibiu
Goal State: Bucharest

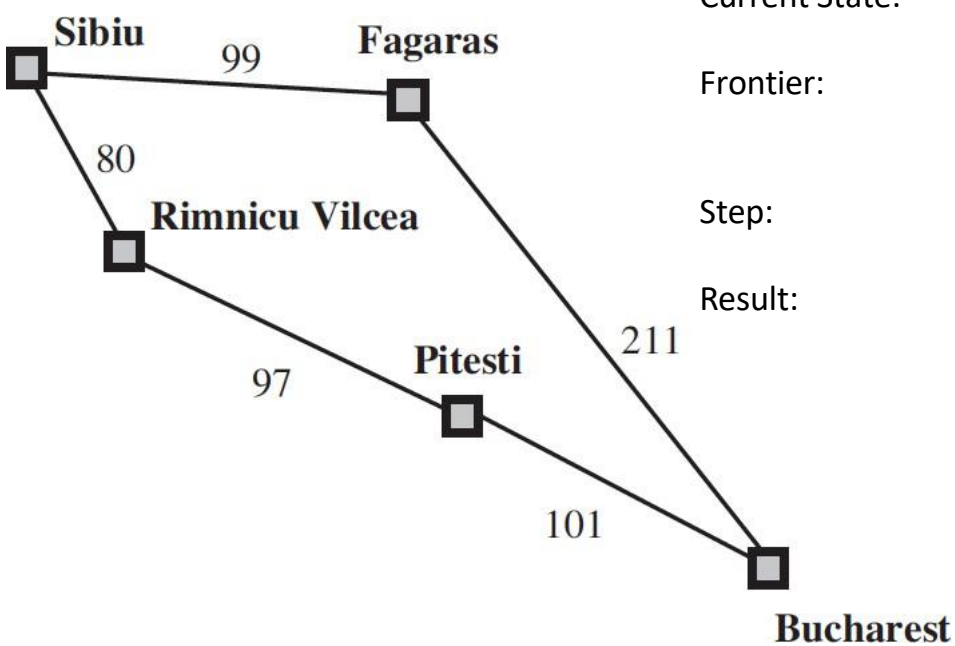
Uniform Cost Search



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

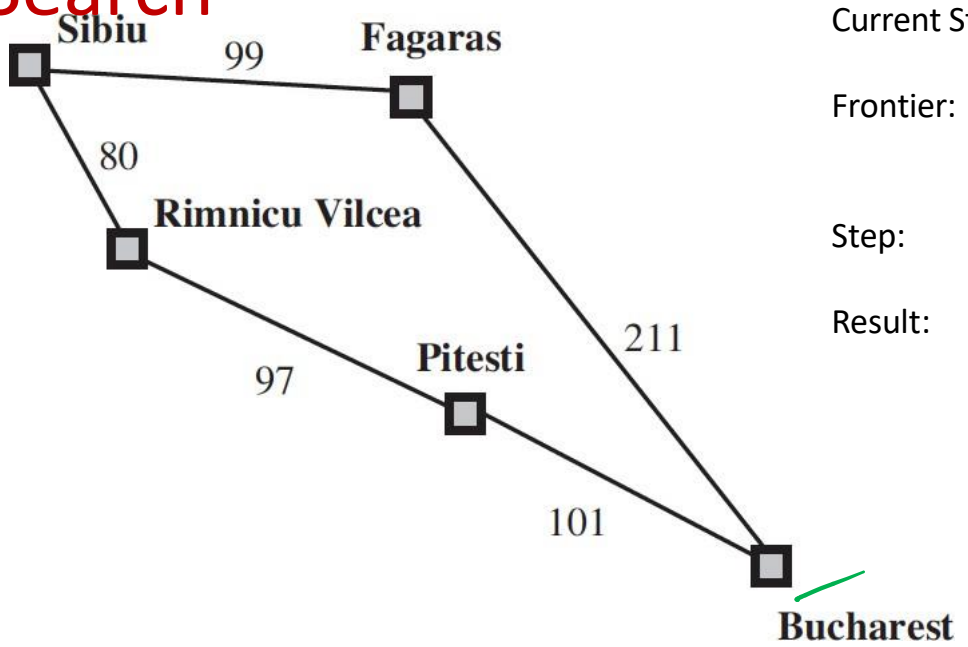
Uniform Cost Search



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

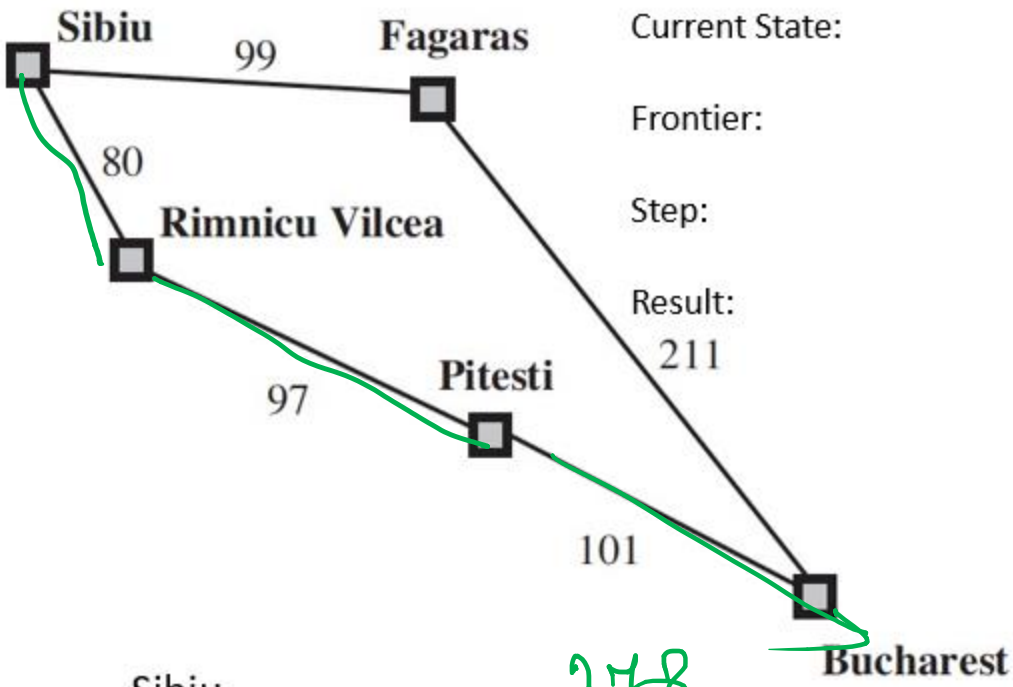
Uniform Cost Search



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



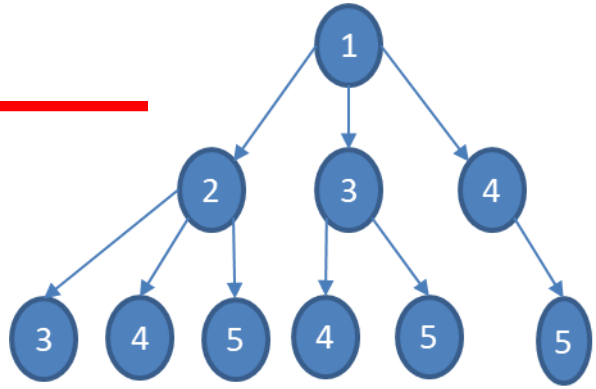
278

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

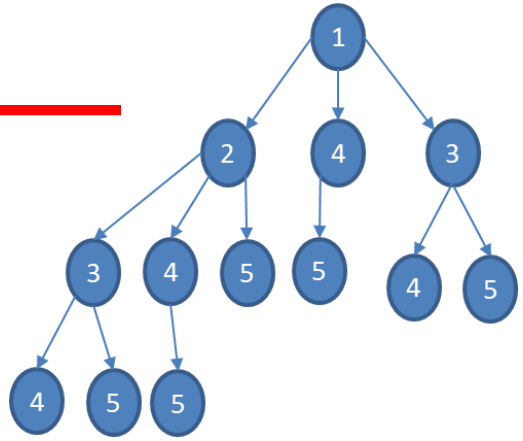
Time Complexity – $G(b^d)$ b - branching factor, d – depth

- 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

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

+ addition info

↳ H(n)

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

Learning Objective

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

1. Differentiate between uninformed and informed search requirements
2. Apply UCS, GBFS & A* algorithms to the given problem
3. Prove if the given heuristics are admissible and consistent
4. Design and compare heuristics apt for given problem
5. Apply A* variations algorithms to the given problem

Informed Search

1. Greedy Best First
2. A*

Informed /Heuristic Search

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

Straight line dist

Eucli dist

Manhandtan

$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$

366

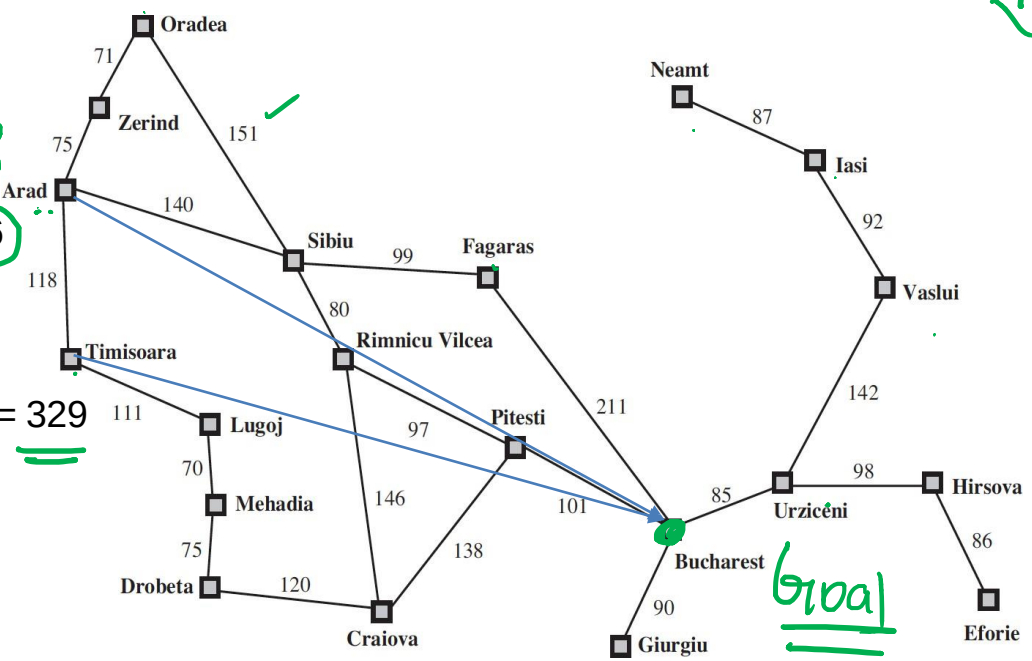
176

dist km

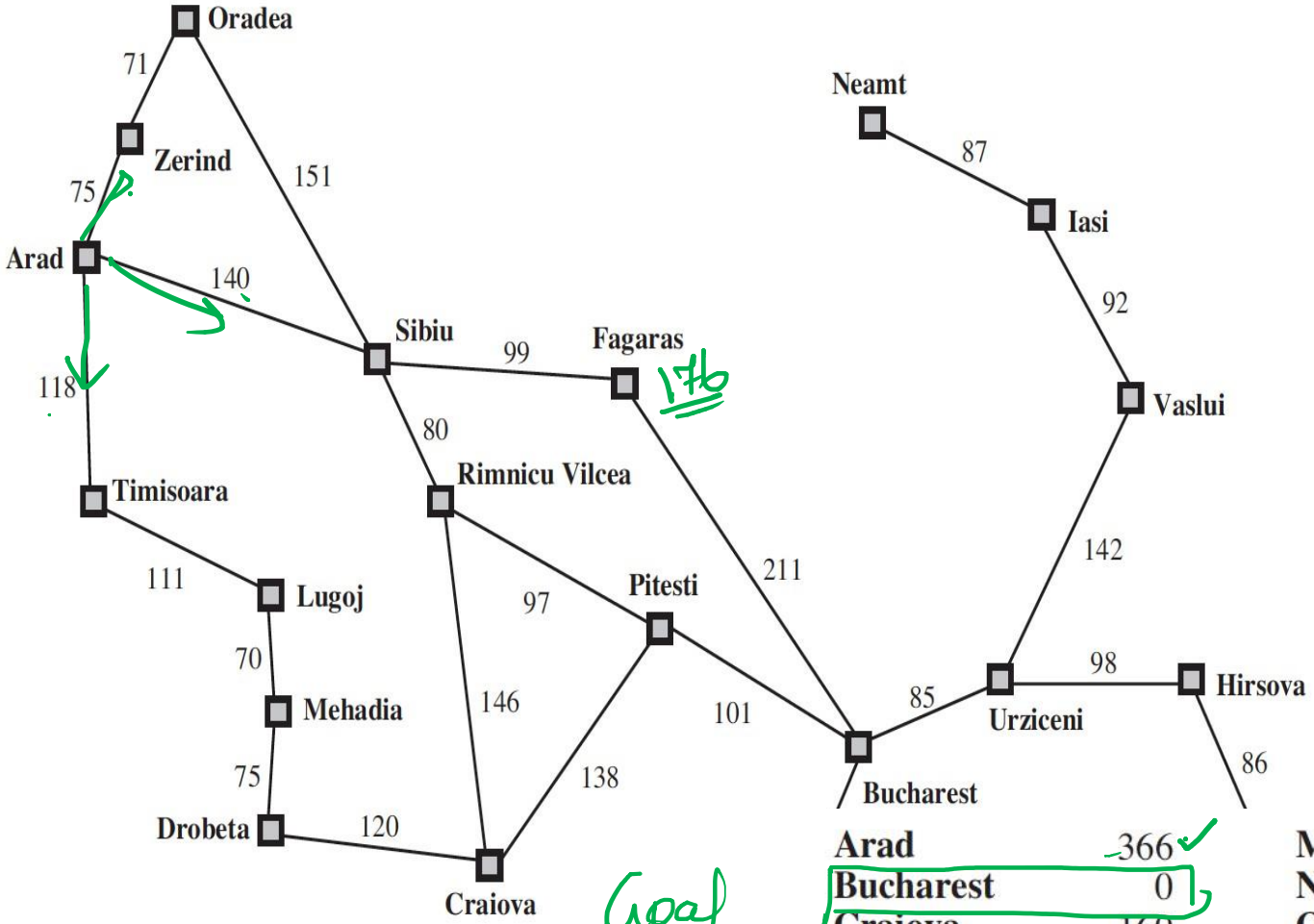
S

h(Arad) = 366

h(Timisoara) = 329



Goal



176

Goal

Goal

Domain expert

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

Path cost

Expands the node that is closest to the goal

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

h(n)

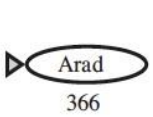
f(n) ⇒ h(n)

↓

evalfn

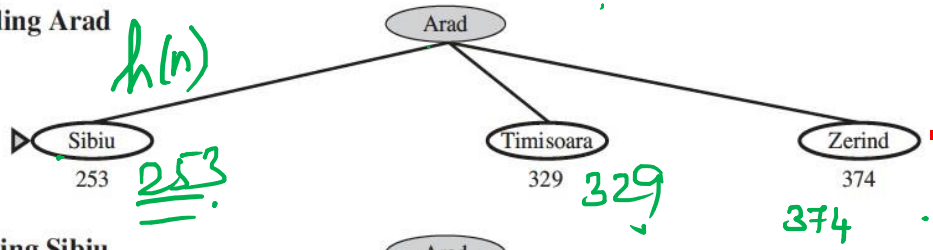
Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

(a) The initial state



Arad
366

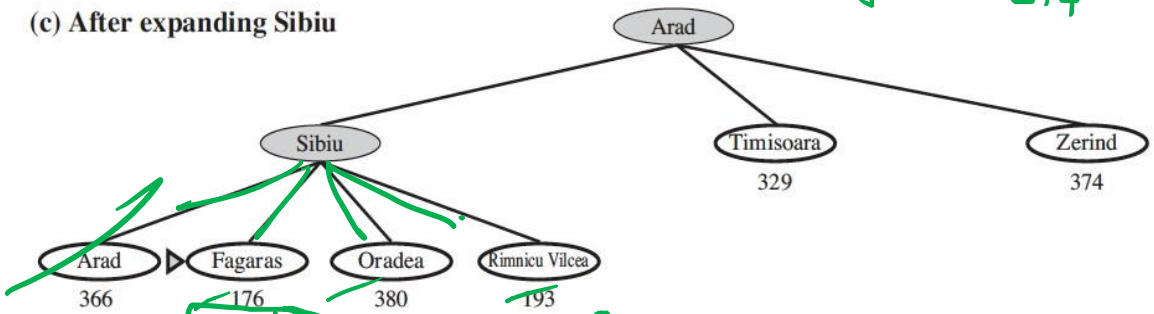
(b) After expanding Arad



h(n)

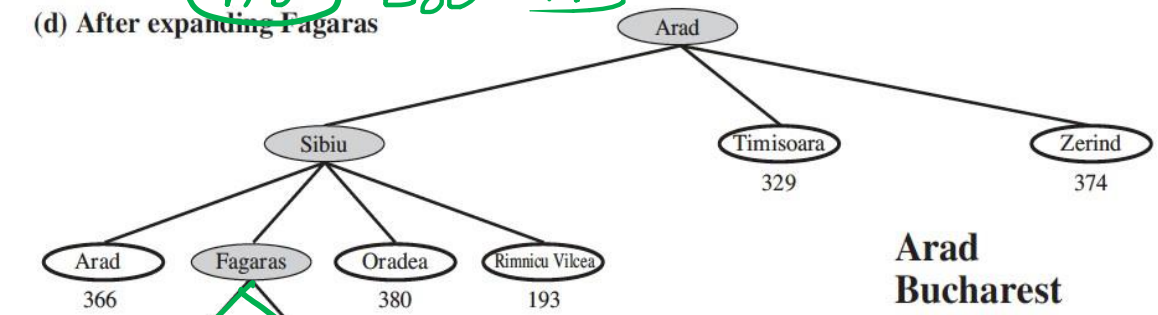
253 329 374

(c) After expanding Sibiu



176 380 193

(d) After expanding Fagaras

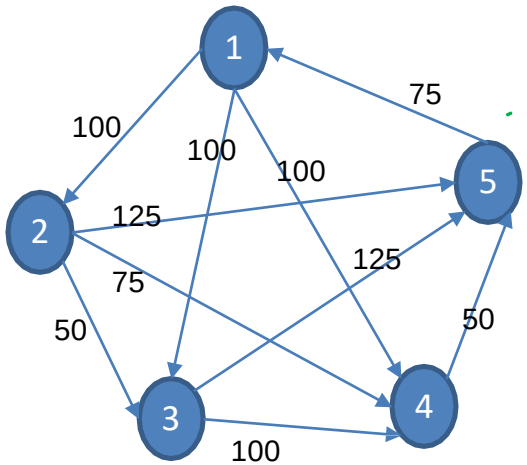


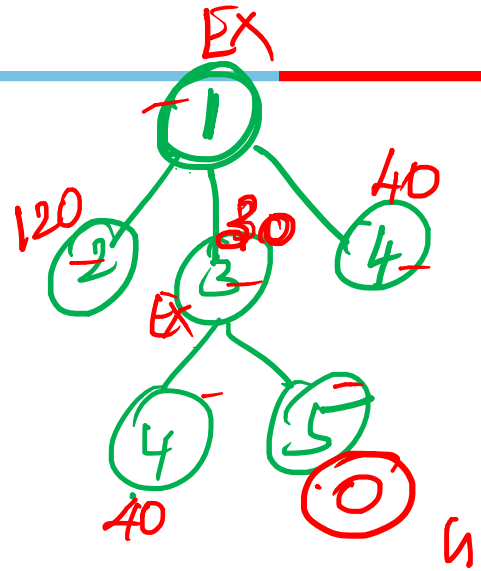
0 → Goal State

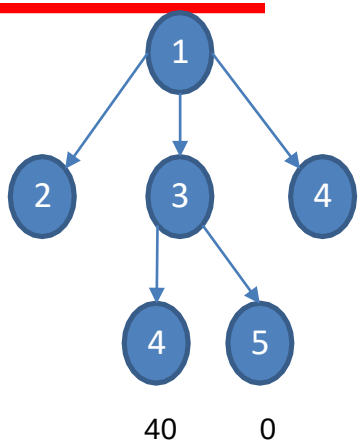
Given

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







Given

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

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

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

A* Search

Path

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

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

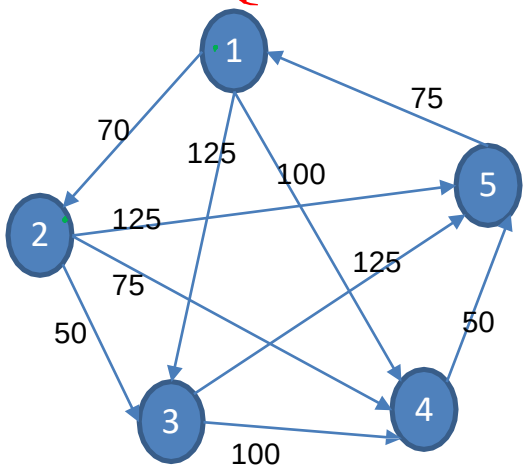
$f(n) = g(n) + h(2)$
$70 + 120$

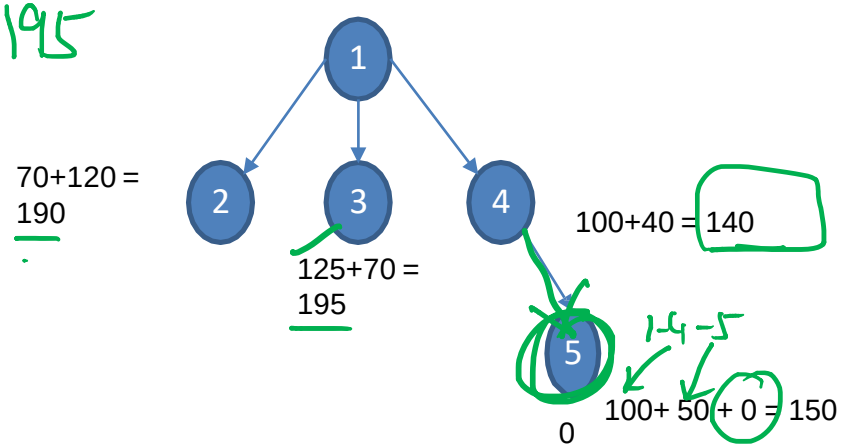
1

2 — 190

3 — $125 + 70 \Rightarrow 195$

4 — $100 + 40 \Rightarrow 140$





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

1-4-5

Expanded : 2
Generated : 5
Max Queue Length : 3

Required Reading: AIMA - Chapter #1, 2, 3.1, 3.2, 3.3

Thank You for all your Attention

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