Artificial and Computational Intelligence

DSECLZG557

Contributors & Designers of document content : Cluster Course Faculty Team

M2 : : Problem Solving Agent using Search

BITS Pilani

Pilani Campus

Presented by
V Indumathi –Guest Faculty –BITS-W
indumathi.p@wilp.bits-pilani.ac.in

Artificial and Computational Intelligence

Disclaimer and Acknowledgement

- Few content for these slides may have been obtained from prescribed books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online.
- .I have provided source information wherever necessary
- This is not a full fledged reading materials. Students are requested to refer to the textbook w.r.t detailed content of the presentation deck that is expected to be shared over e-learning portal - taxilla.
- I have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation

- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof.Raja vadhana, Prof. Indumathi, Prof.Sangeetha
- From BITS Oncampus & External : Mr.Santosh GSK

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

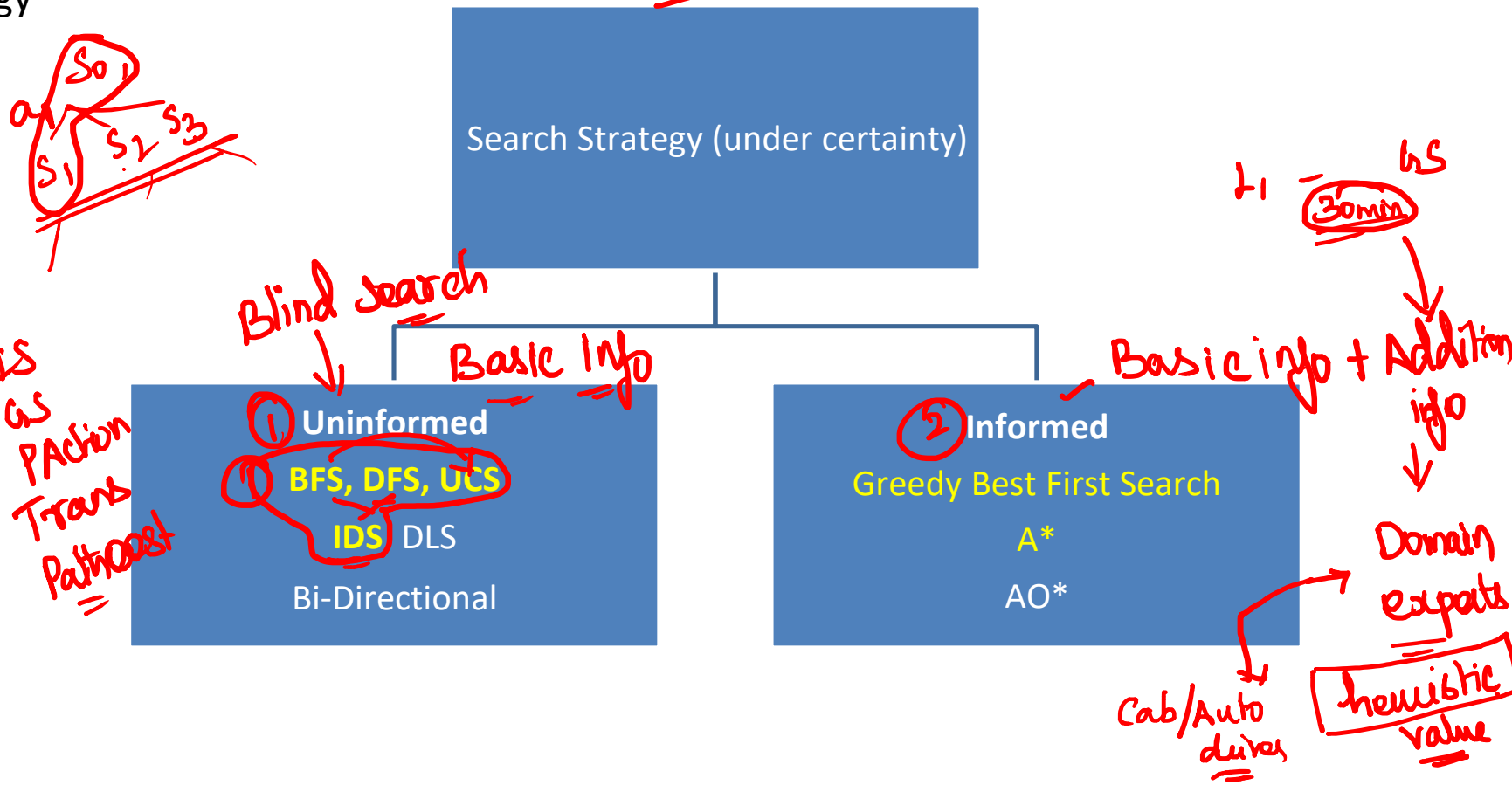
At the end of this class , students Should be able to:

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.**
6. **Differentiate between Tree and Graph search**

Problem Formulation

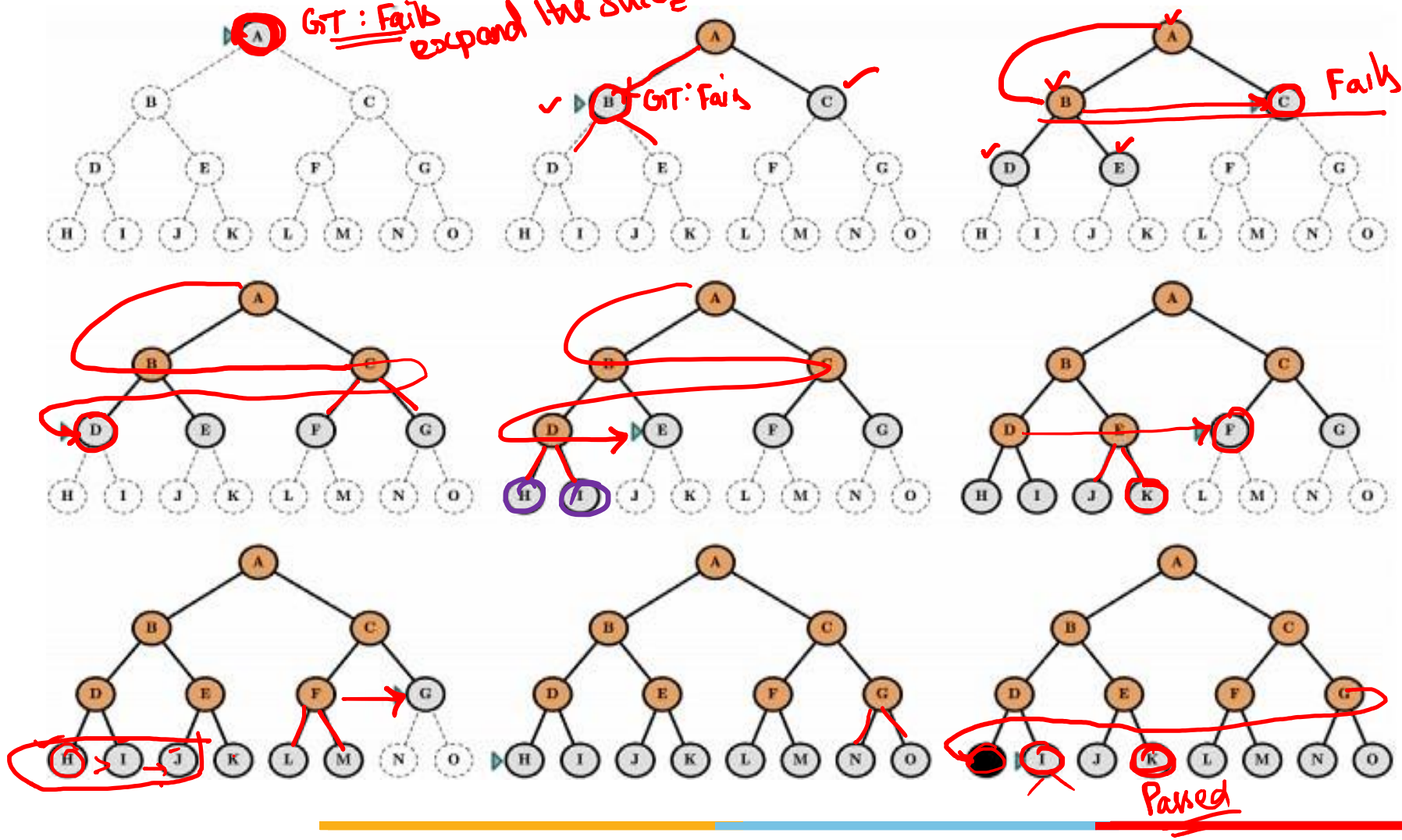
Searching for Solutions

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



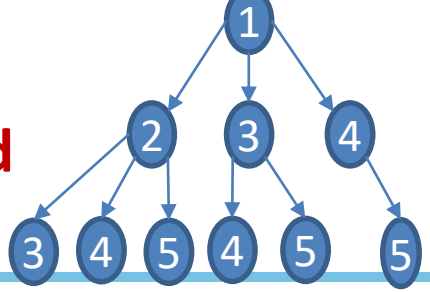
Uninformed Search – BFS & its Variant

Breadth First Search (BFS)



FIFO – Queue

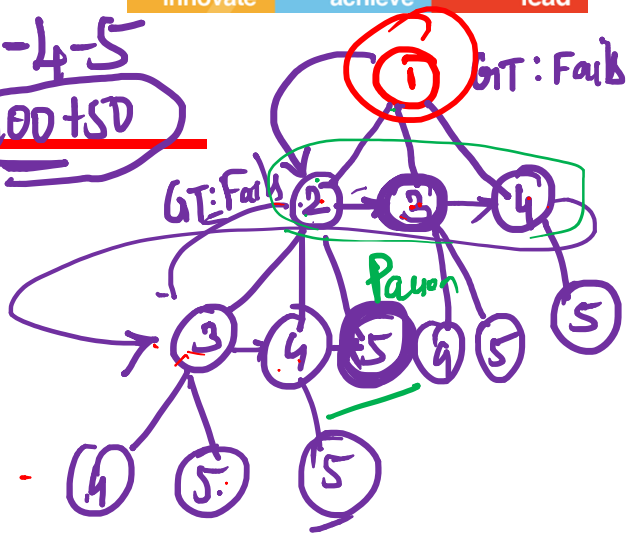
BFS – Uninformed

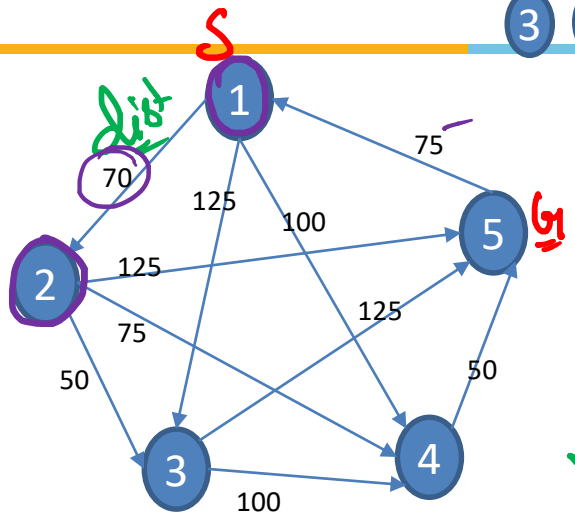


optimal. 1-4-5

cost: 100+50

Y?





P1

125

1-2-5

70+125 => 195

	open list	closed list	Goal test
1	(1)		F(1)
2	(12) (13) (14) ←	(1)	F(12)
8	(13) (14) (123) (124) (125)	(1)(12)	F(13)
6	(14) (123) (124) (125) (134) (135)	(1) (12) (13)	F(14)
5	(123) (124) (125) (134) (135) (145)	(1) (12) (13) (14)	F(123)
7	(124) (125) " " " (1234) (1235)	" (123)	F(124)

OL

(125) .....

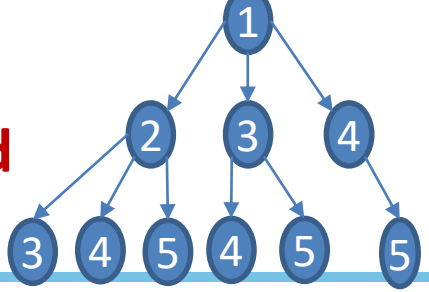
7 (1234)(1235) (1245)

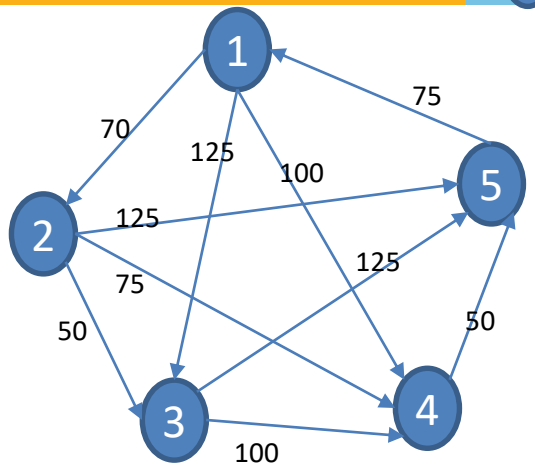
#nodes/Successor Generated: 13

max Queue length: 7

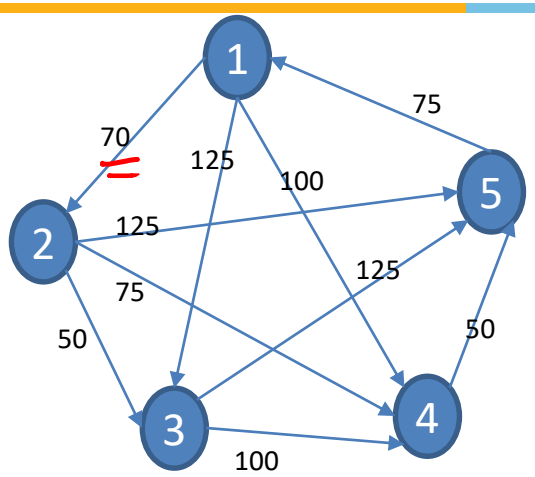
Expanded: 6//

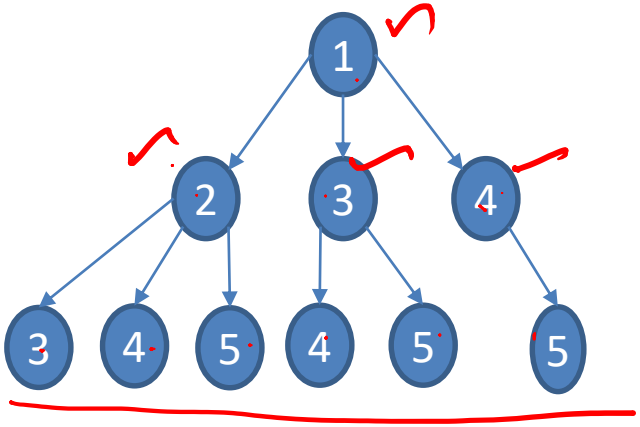
BFS – Uninformed





BFS – Uninformed





(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

Breadth First Search – Evaluation

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 – $\mathcal{O}(b^d)$ 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 $\mathcal{O}(b^{d+1})$

Space Complexity – $\mathcal{O}(b^d)$

- $\mathcal{O}(b^{d-1})$ in explored set
- $\mathcal{O}(b^d)$ in frontier set

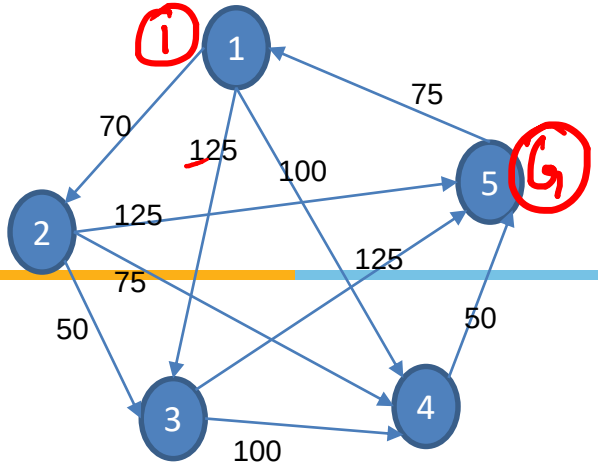
Uniform Cost Search

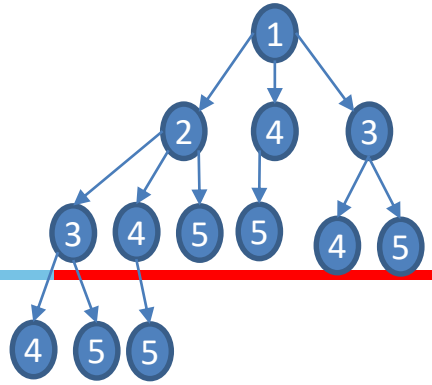
- Instead of expanding the shallowest node, Uniform-Cost search expands the node n with the lowest path cost g(n)

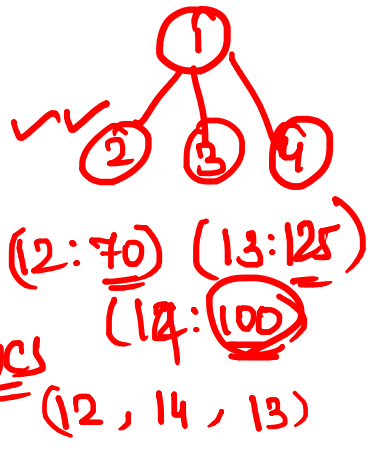
- Sorting the Frontier as a priority queue ordered by g(n)

- 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

UCS

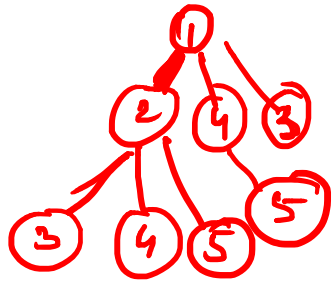






Table

open list	closed list	Goal
(1)		F(1)
(2:70) (4:100) (3:125)	(1)	F(2)
(4:100) (23:120) (3:125) (24:145) (125:195)	(1) (2)	F(4)
(23:120) (3:125) (24:145) (145:180) (125:195)	(1) (2) (4)	

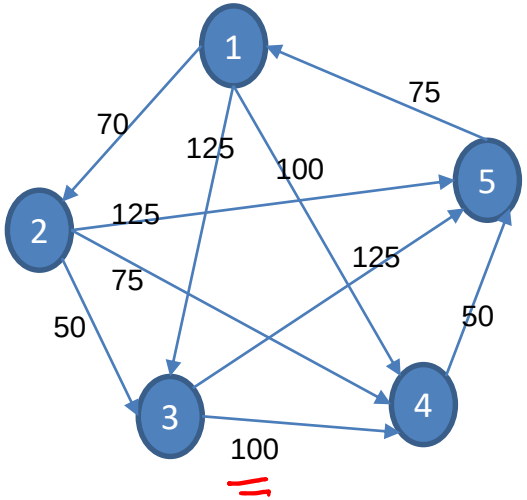


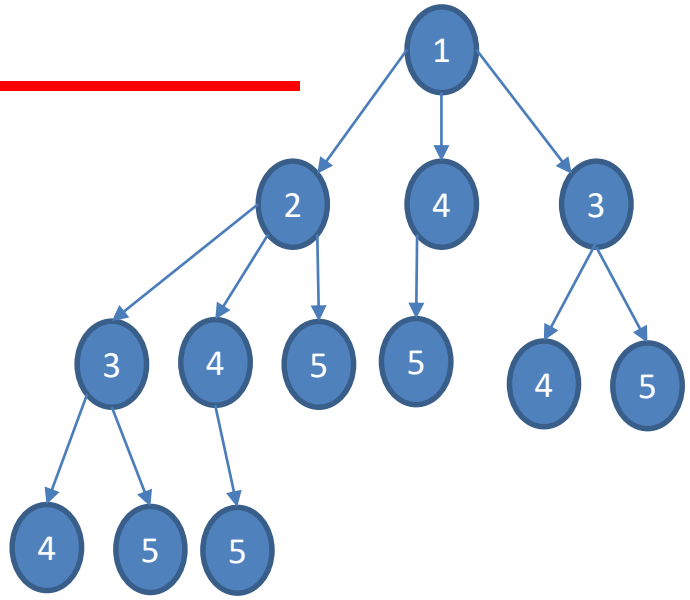
(23:120) (24:145)

(125:195)

145:180

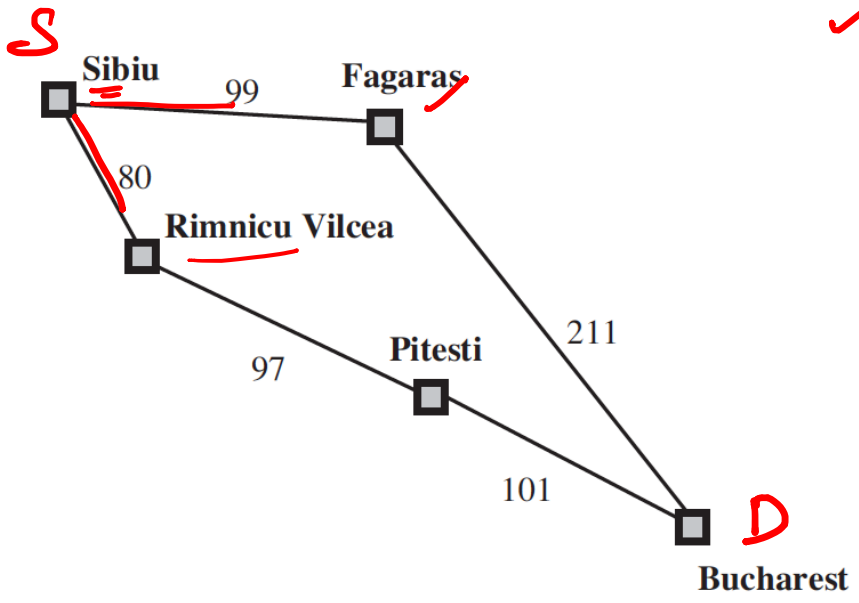
UCS





(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 5 : 195) (1 2 3 4:220) (1 2 3 5 : 245)
TEST-F
(1 2 4 : 145) (1 4 5: 150) (1 2 5 : 195) (1 2 3 4:220) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST-F
(1 4 5: 150) (1 2 4 5 : 195) (1 2 5 : 195) (1 2 3 4:220) (1 3 4 : 225) (1 2 3 5 : 245) (1 3 5 : 250)
TEST - P

Uniform Cost Search



Current State: Sibiu

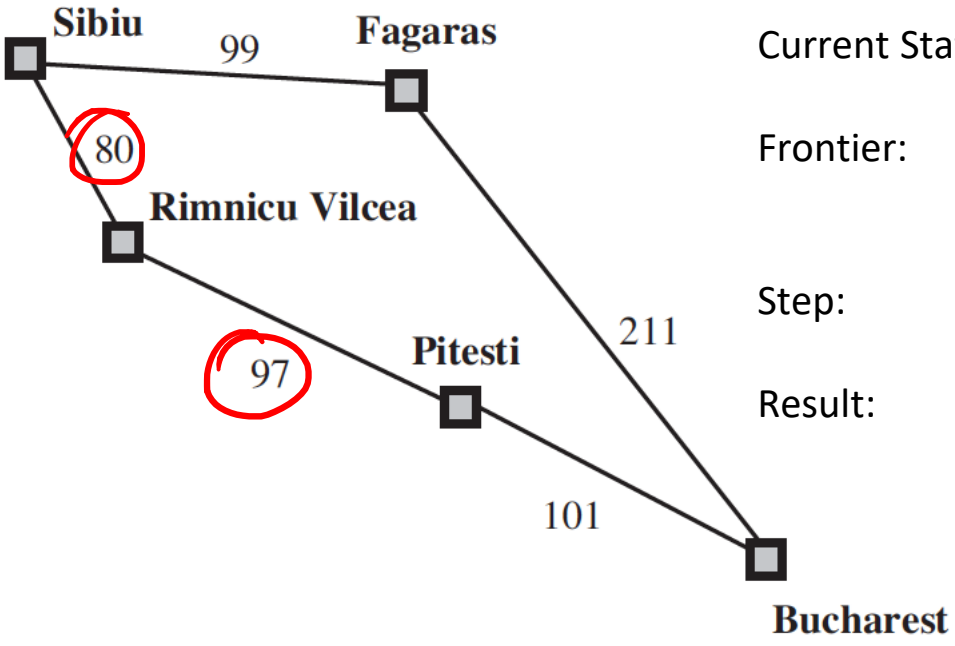
Frontier: []

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

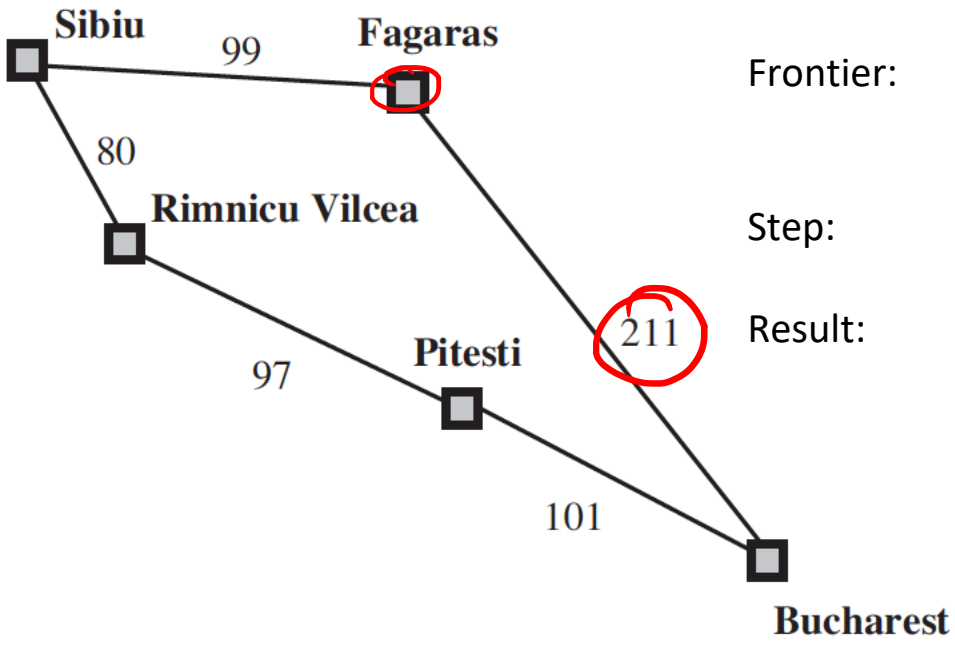
Frontier: [("Rimnicu Vilcea", 80)
("Fagaras", 99)]

Step: Expand "Rimnicu Vilcea" (least cost)

Result: Generates ("Pitesti", 177)
Add to Frontier

Initial State: Sibiu
Goal State: Bucharest

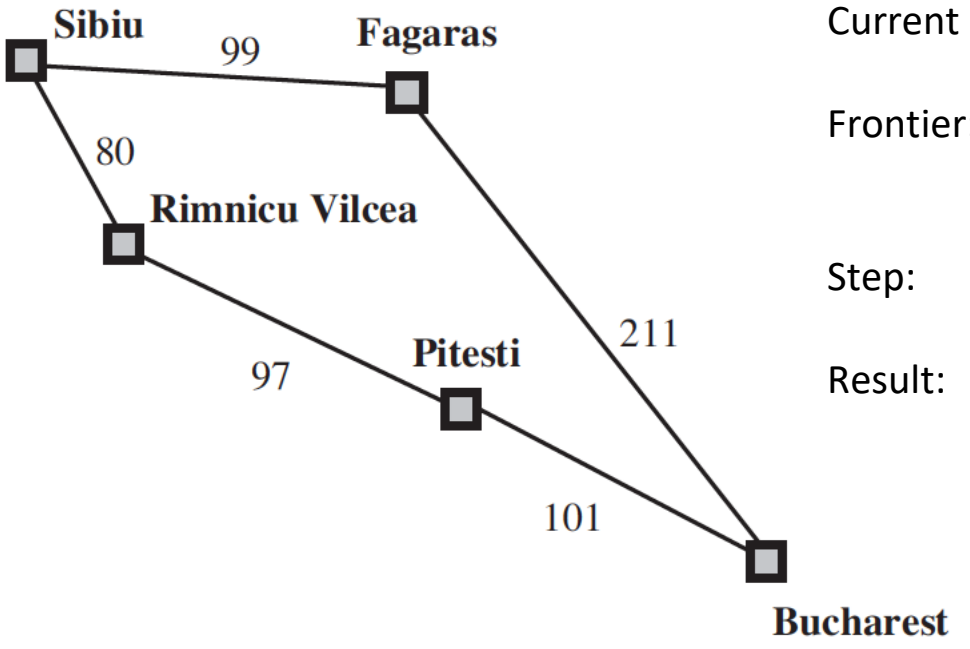
Uniform Cost Search



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

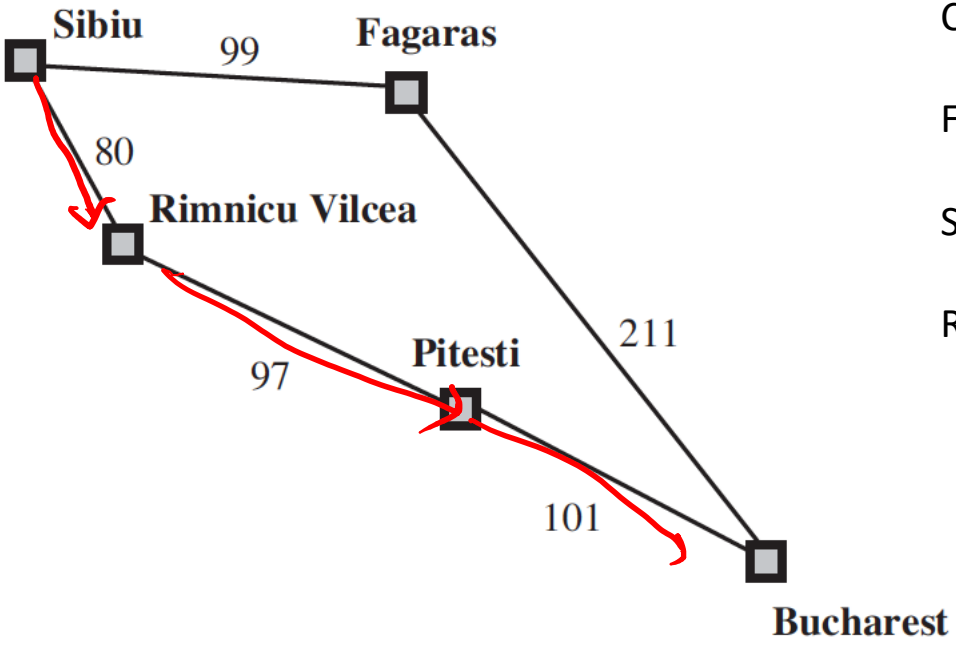
Uniform Cost Search



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

278

Initial State:	Sibiu
Goal State:	Bucharest

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

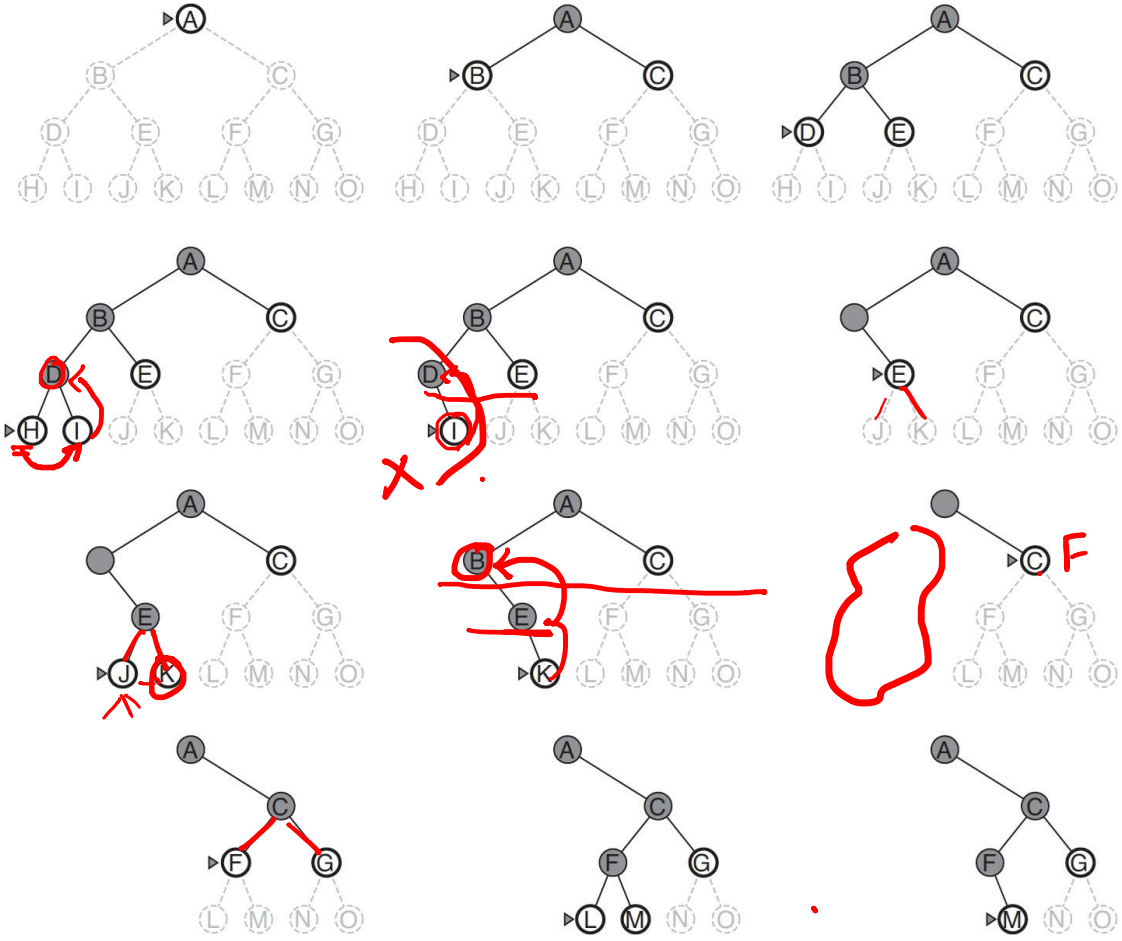
Optimal – It is Optimal. Whenever it selects a node, it is an optimal path to that node.

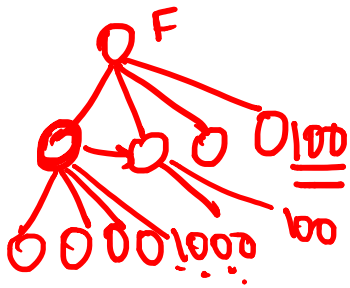
Time and Space complexity – Uniform cost search is guided by path costs not depth or branching factor.

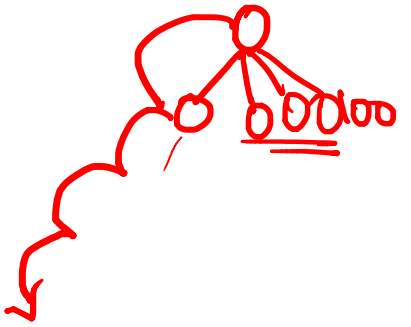
- If C* is the cost of optimal solution and ∈ is the min. action cost
- Worst case complexity = $\mathcal{O}(b^{1+\frac{C^*}{\epsilon}})$,
- When all action costs are equal → $\mathcal{O}(b^{d+1})$, the BFS would perform better
  - As Goal test is applied during expansion, Uniform Cost search would do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

Uninformed Search – DFS & its Variant

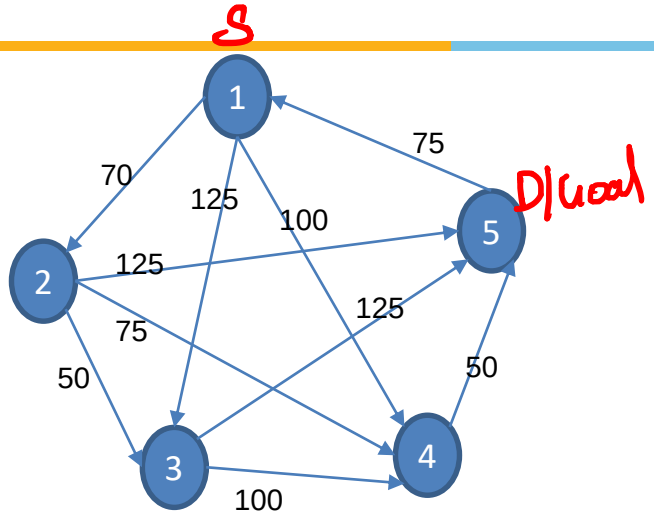
Depth First Search (DFS)

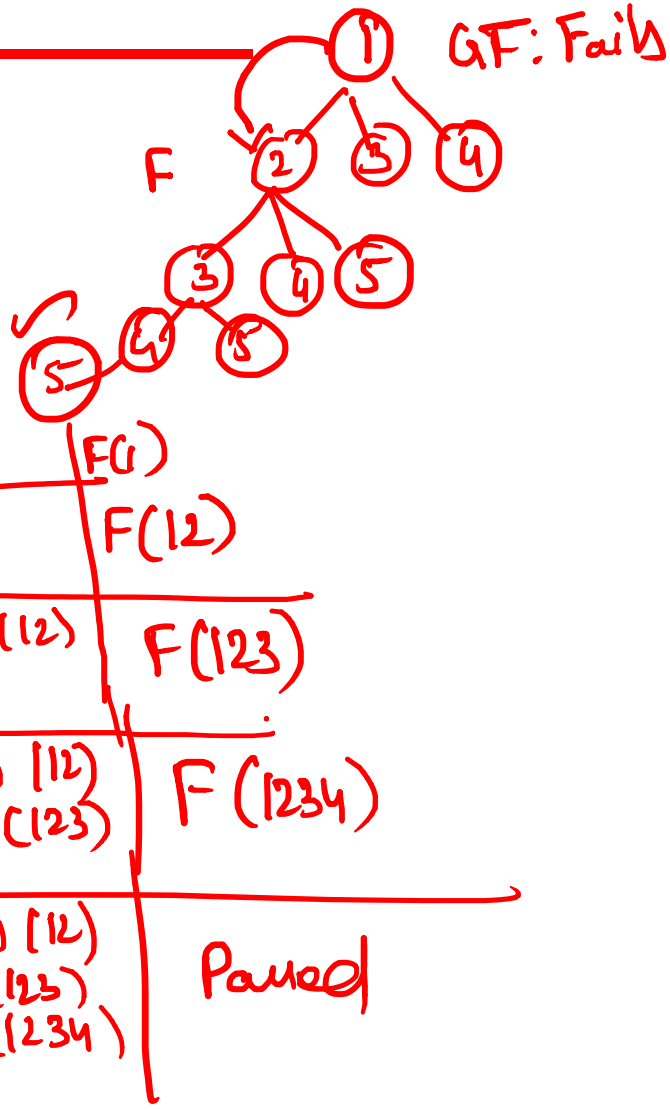




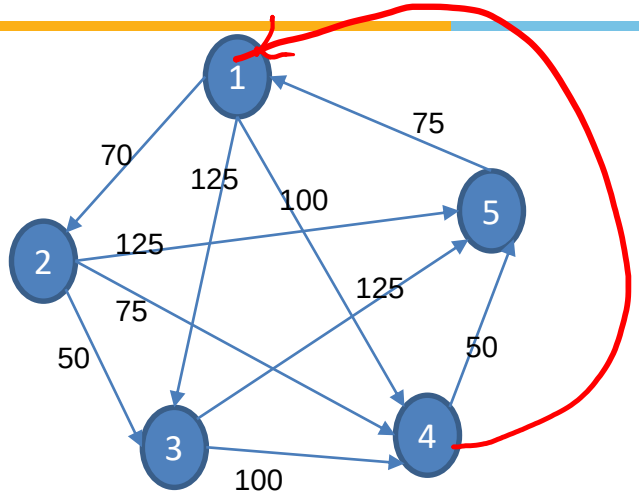


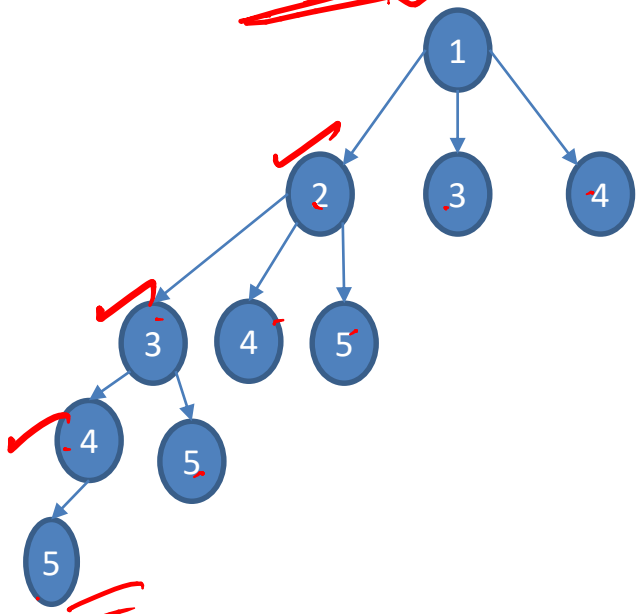
DFS – Uninformed → LIFO stack





DFS – Uninformed





(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

Depth First Search (DFS)

Completeness – Complete in finite state spaces because it will eventually expand every node

Optimal – Not Optimal as it would stop when the goal node is reached without evaluating if there is a better path

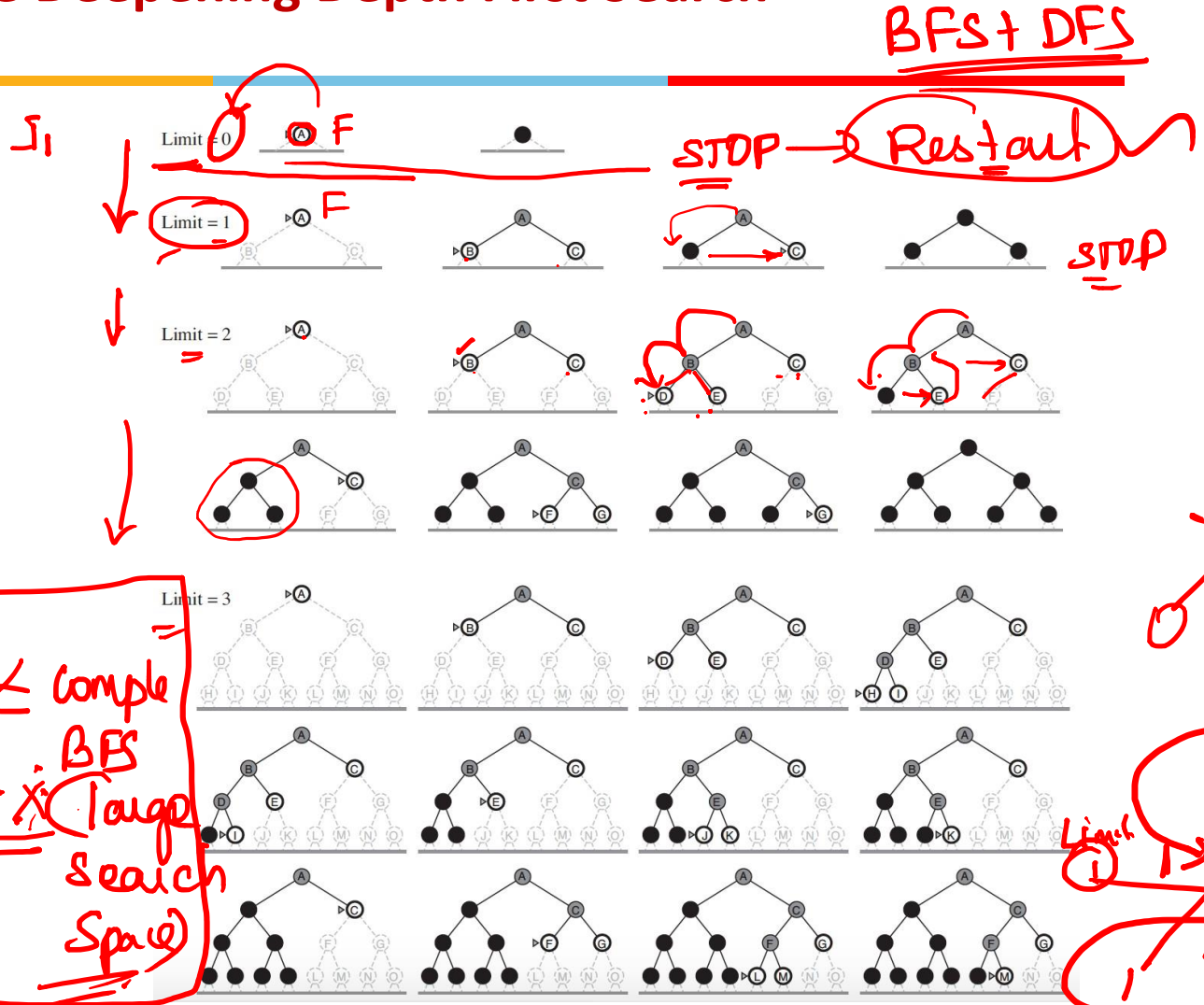
Time Complexity - $\mathcal{O}(b^m)$ where m = maximum depth of any node

- Can be much larger than the size of state space
- m can be much larger than d (shallowest goal)

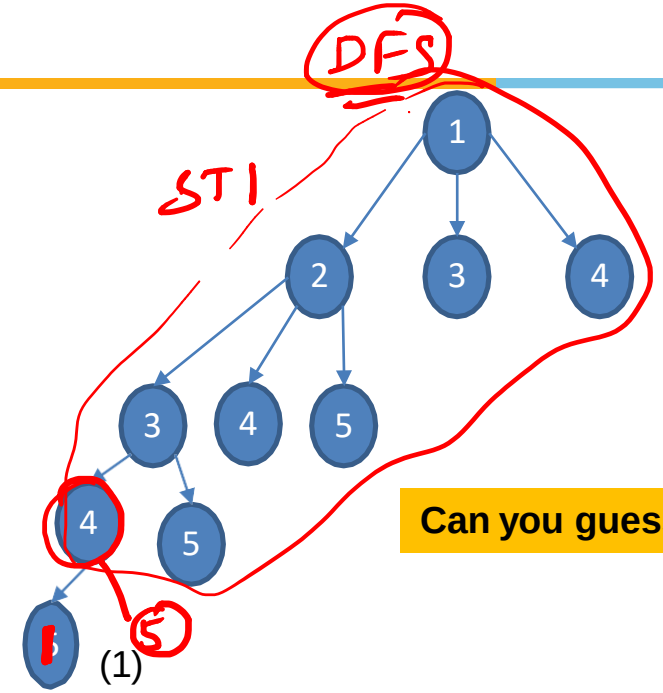
Space Complexity – Needs to store only one path and unexpanded siblings.

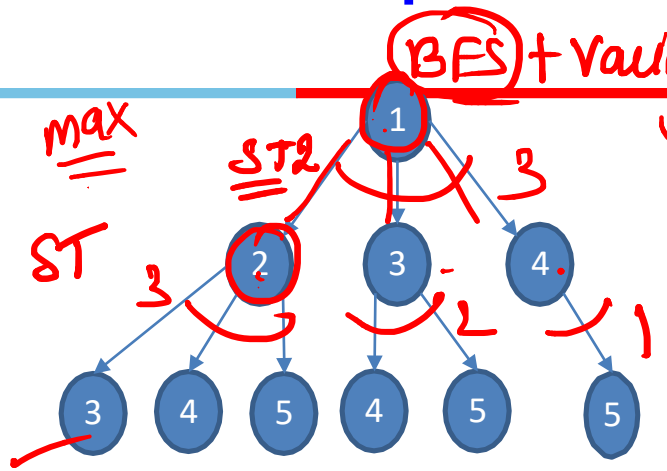
- Any node expanded with all its children can be removed from memory
- Requires storage of only $\mathcal{O}(bm)$, b – branching factor, m - max depth

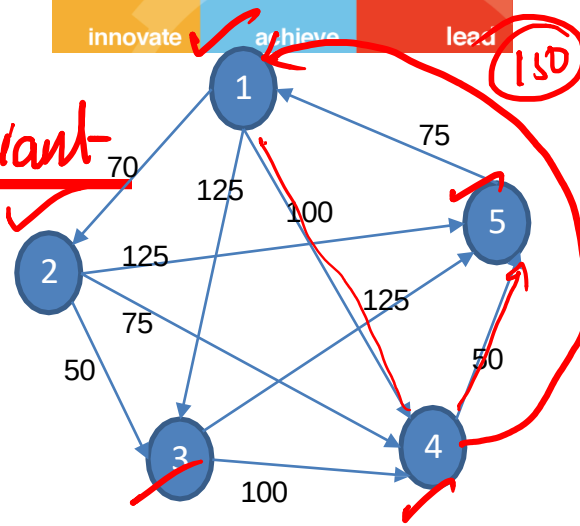
Iterative Deepening Depth First Search (IDS)



Search Algorithm – Uninformed Example







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

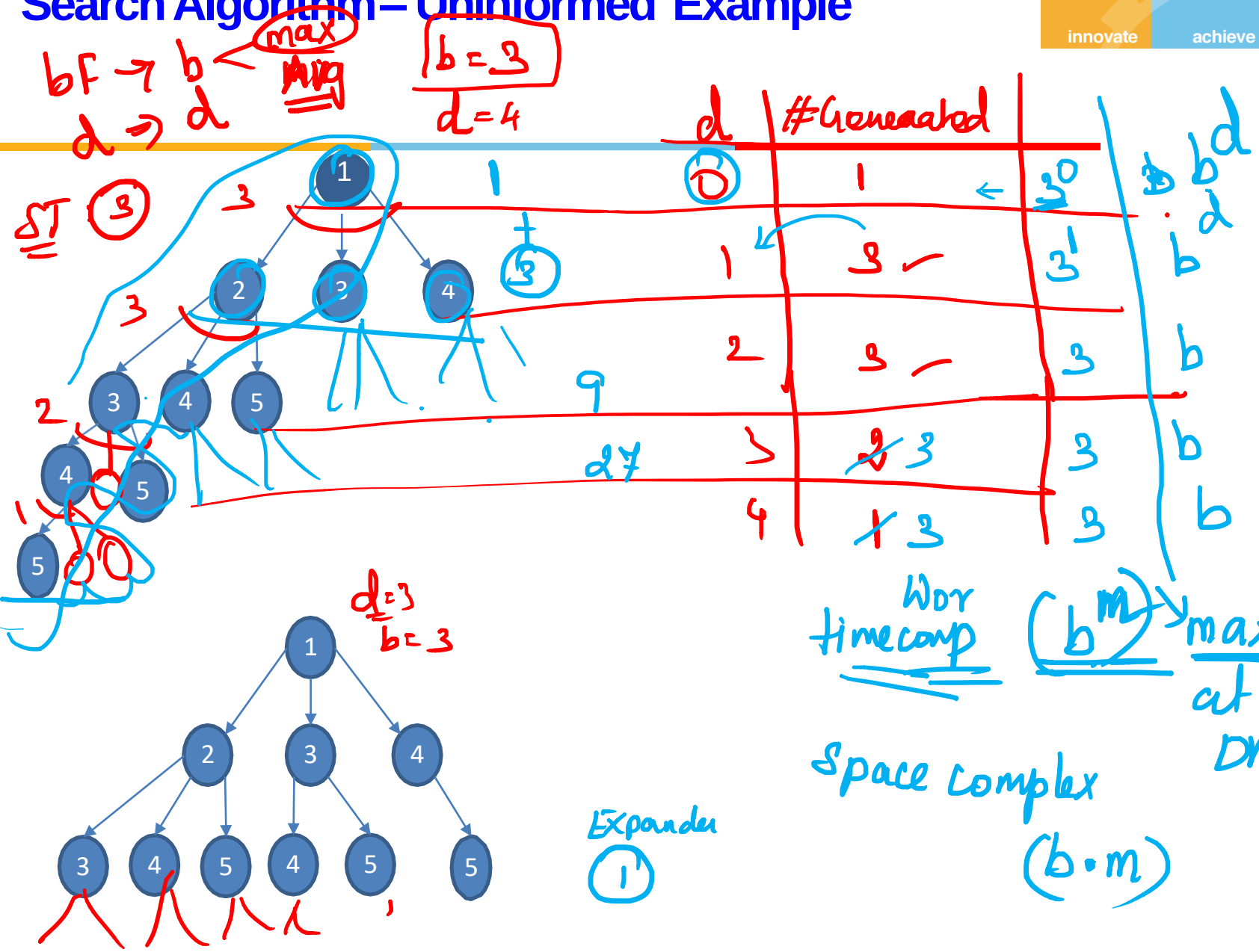
(1)
(1 2) (1 3) (1 4)
TEST FAILED

(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

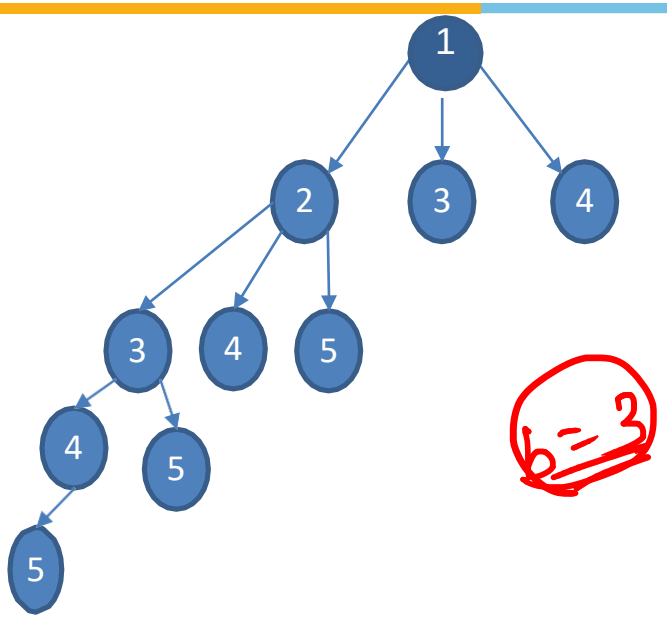
C(1-2-5) = 70 + 125 =195
Expanded : 4
Generated : 10
Max Queue Length : 6

1. optimality
2. completeness
3. Time Complexity
4. Space

Search Algorithm – Uninformed Example

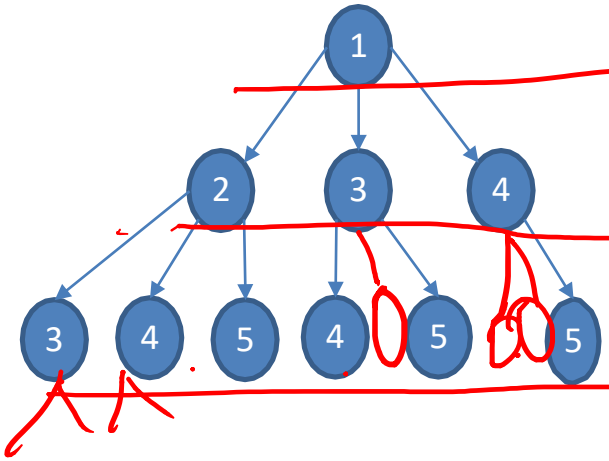


Search Algorithm – Uninformed Example



b = 3

(b^d) Goal test



d			b^d
0	1	3^0	
1	3	3^1	b^1
2	9	3^2	b^2
3	27	3^3	b^3

4

Iterative Deepening Depth First Search (IDS)

Run Depth Limited Search (DLS) by gradually increasing the limit l

- First with l=1, then l=2, l=3 and so on – until goal is found

Its is a combination of Depth First Search + Breadth First Search

Like DFS, memory requirement is a modest $\mathcal{O}(bd)$ where d is the depth of shallowest goal

Like BFS

- Complete when branching factor is finite
- Optimal when path cost is non decreasing function of depth

Breadth First Search – Evaluation

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

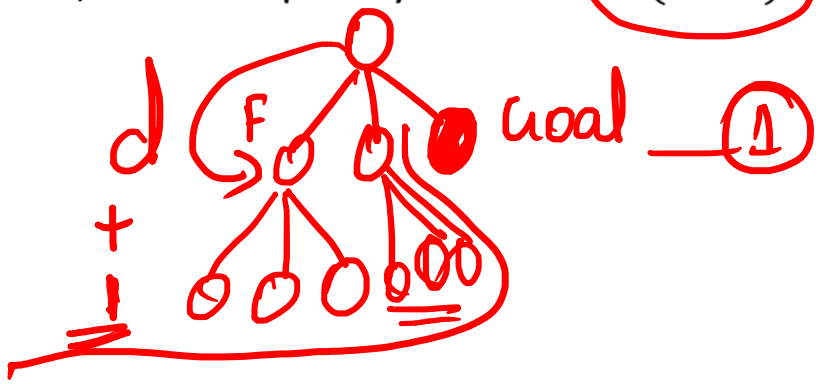
Optimal – Not necessarily. Optimal if path cost is non-decreasing function of depth of node. E.g., all actions have same cost

Time Complexity – $\mathcal{O}(b^d)$ 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 $\mathcal{O}(b^{d+1})$

Space Complexity – $\mathcal{O}(b^d)$

- $\mathcal{O}(b^{d-1})$ in explored set
- $\mathcal{O}(b^d)$ in frontier set



Depth First Search (DFS)

Completeness – Complete in finite state spaces because it will eventually expand every node

Optimal – Not Optimal as it would stop when the goal node is reached without evaluating if there is a better path

Time Complexity - $\mathcal{O}(b^m)$ where m = maximum depth of any node

- Can be much larger than the size of state space
- m can be much larger than d (shallowest goal)

Space Complexity – Needs to store only one path and unexpanded siblings.

- Any node expanded with all its children can be removed from memory
- Requires storage of only $\mathcal{O}(bm)$, b – branching factor, m - max depth

Uniform Cost Search – Evaluation

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

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

Iterative Deepening Depth First Search (IDS)

innovate achieve lead

Time Complexity:

- Appears that IDS is generating a lot of nodes multiple times
- However, most of nodes are present in the lower levels which are not repeated often
- Generation of nodes
  - At level 1 - b nodes generated d times – (d)b
  - At level 2 – b^2 nodes generated d-1 times – (d-1)b^2
  - At level d – b^d nodes generated once – (1) b^d
- Time Complexity N(IDS) = $\mathcal{O}(b^d)$ same as BFS

IDS is the preferred uninformed search method when search space is large and depth is unknown

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

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

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

Thank You for all your Attention

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