CS 461
ARTIFICIAL INTELLIGENCE

Lecture # 04
March 11, 2021
SPRING 2021
FAST – NUCES, CFD Campus

Dr. Rabia Maqsood
rabia.maqsood@nu.edu.pk

Today’s Topics

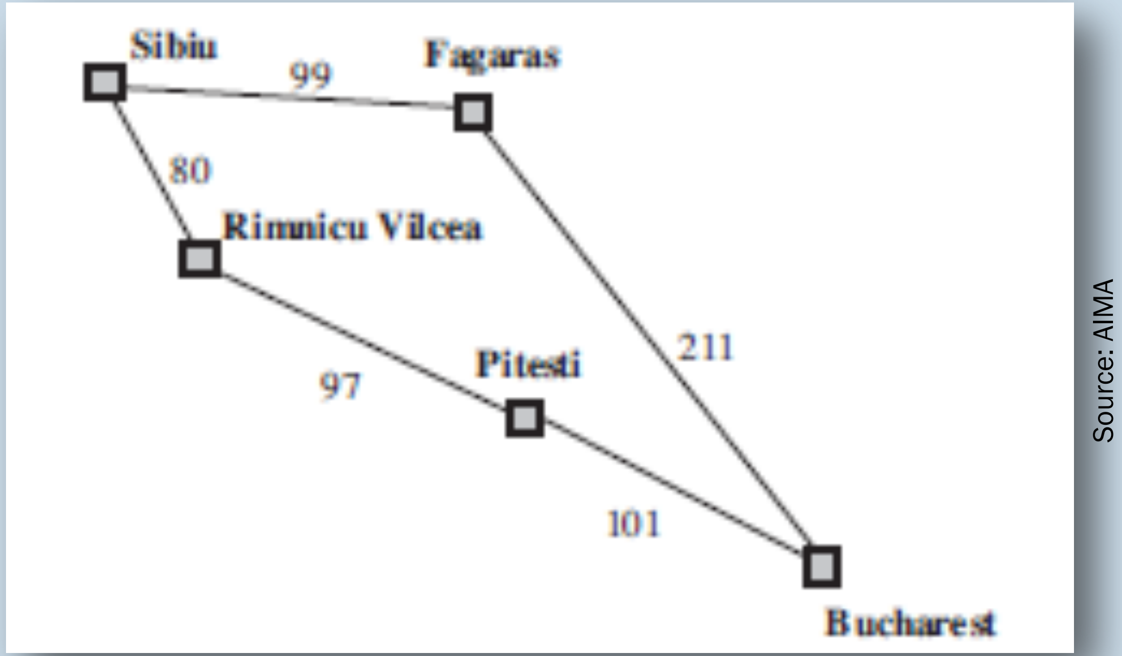
- Search strategies
  - *Uninformed search algorithms*
    - Uniform Cost Search
    - Depth Limited Search
    - Iterative Deepening Search
    - Bidirectional Search

Uniform-Cost Search (UCS)

Recall that, BFS is optimal if all step costs are equal

- Basic idea
  - *Expand the least-cost unexpanded node*
  - *Works well for any cost function*

- More insights on UCS vs. BFS
  - ***Goal test** is applied to a node when it is selected for expansion*
  - *A test is added in case a better path is found to a node currently on the frontier*



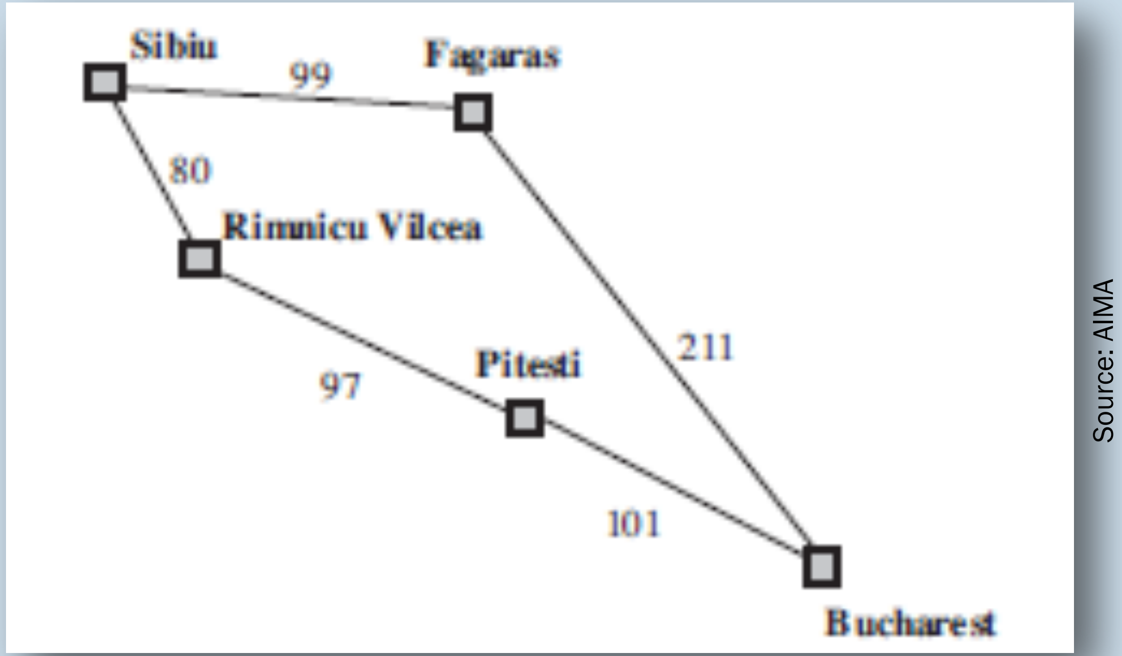
Source: AIMA

Uniform-Cost Search (UCS)

Recall that, BFS is optimal if all step costs are equal

In UCS, the frontier is a priority queue

- Basic idea
  - *Expand the least-cost unexpanded node*
  - *Works well for any cost function*
- Implementation
- Properties
  - *Complete?*
  - *Optimal?*
  - *Time?*
  - *Space?*



Source: AIMA

Uniform-Cost Search (UCS)

- Complete: Yes (if step cost ≥ ε)
- Optimal: Yes (node with least-cost is always expanded)
- Time: *No. of nodes with g <= cost of optimal solution*
- Space: *No. of nodes with g <= cost of optimal solution*

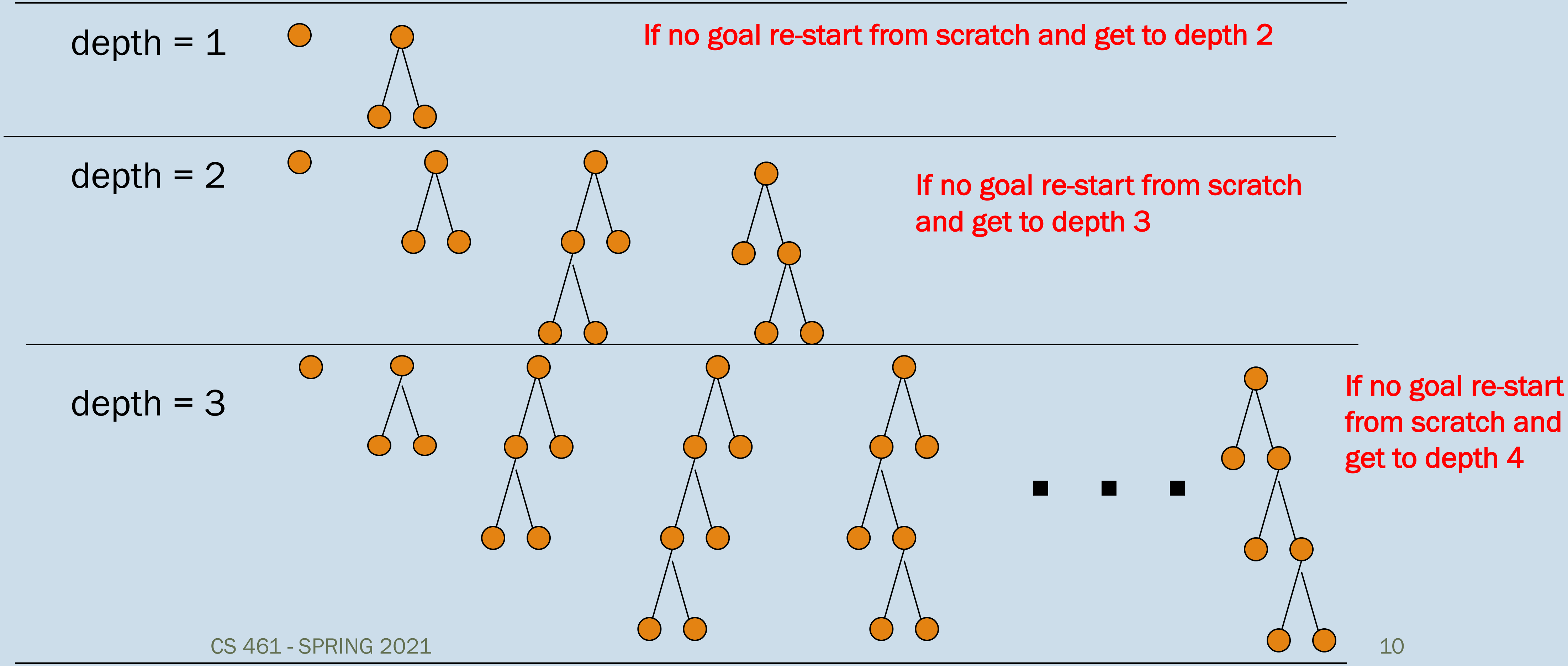
Depth-limited Search

- Basic idea
  - *DFS with depth limit k, i.e., nodes at depth k have no successors*
- Implementation
- Properties
  - *Complete: NO (if k < d)*
  - *Optimality: NO (if k > d)*
  - *Time: O(b^k)*
  - *Space: O(bk)*

Iterative Deepening Search (IDS)

- Basic idea
  - Use DFS to look for solutions at depth 1, then 2, then 3, etc.
  - For depth D, ignore any paths with longer length
  - Depth-bounded depth-first search

Iterative Deepening Search (IDS)



depth = 1

If no goal re-start from scratch and get to depth 2

depth = 2

If no goal re-start from scratch and get to depth 3

depth = 3

. . .

If no goal re-start from scratch and get to depth 4

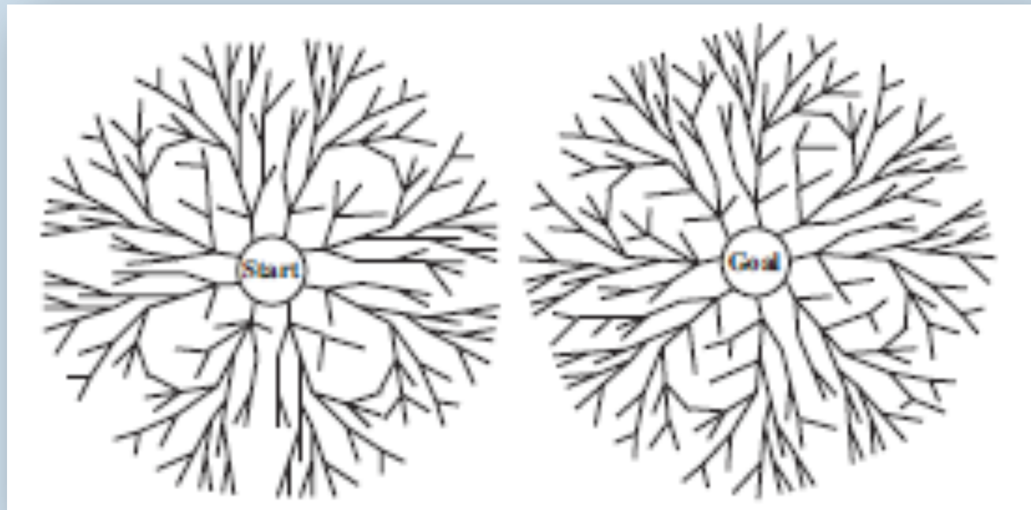
Iterative Deepening Search (IDS)

- Properties
  - *Complete: Yes*
  - *Optimal: Yes, if step cost = 1*
  - *Time: (d)b + (d-1)b^2 + … + (1)b^d = O(b^d)*
  - *Space: O(bd)*

- Repetition is wasteful – isn't it?

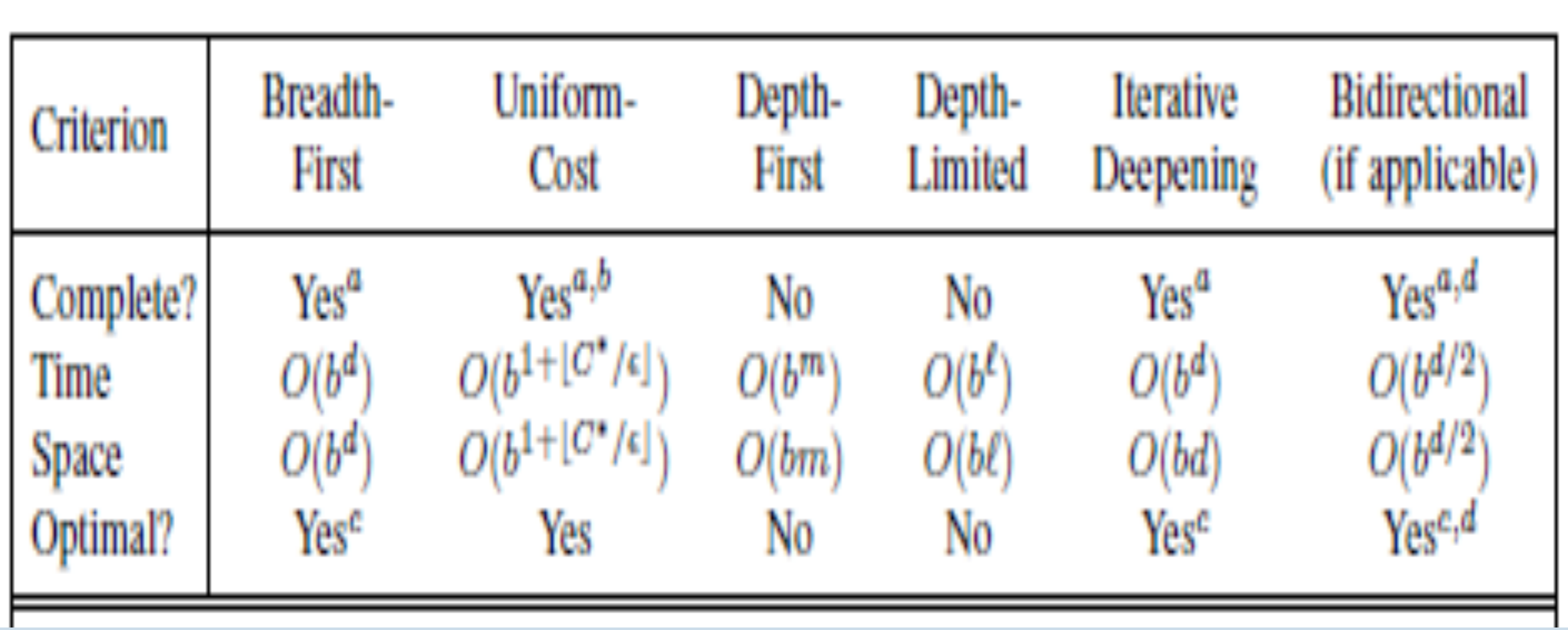
Bidirectional Search

- Basic idea
  - *Run two simultaneous searches, **one forward** from the initial state and **other backward** from the goal state, hoping that the two searches meet in the middle*
- Implementation
- Properties
  - *Complete:* Yes
  - *Optimal:* No
  - *Time:* $O(b^{d/2})$
  - *Space:* $O(b^{d/2})$



Source: AIMA

Comparison: Uninformed Search Strategies



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Source: AIMA

Summary

- Problem formulation usually requires abstracting away real-world details to define a state space that can feasibly be explored
- Variety of uninformed search strategies
- Iterative deepening search uses only linear space and not much more time than other uninformed algorithms

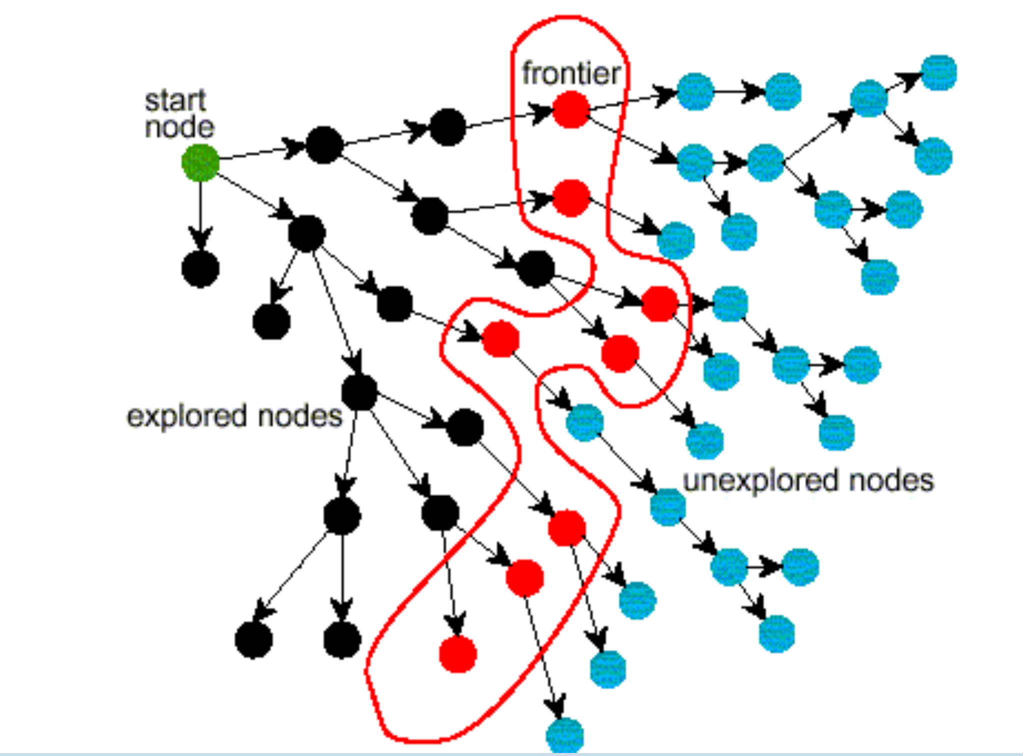
Graph search: basic idea

Input:

- a graph
- a set of start nodes
- Boolean procedure goal(n) testing if n is a goal node

```
frontier:= [<s>: s is a start node];
While frontier is not empty:
    select and remove path $<n_o,....,n_k>$ from
        frontier;
    If goal($n_k$)
        return $<n_o,....,n_k>$;
    For every neighbor n of $n_{k,}$
        add $<n_o,....,n_k, n>$ to frontier;
end
```

The way in which the frontier is expanded defines the **search strategy**



Reading Material

- Russell & Norvig: Chapter # 3
- David Poole: Chapter # 3