Artificial Intelligence

Week 6

Dr Uzma Jamil

Department of Computer Science

Government College University, Faisalabad.

Algorithms for Search

Inputs:

- a specified initial state (a specific world state)
- a successor function $S(x)$ = {set of states that can be reached from state x via a single action}.
- a goal test a function that can be applied to a state and returns true if the state satisfies the goal condition.
- An action cost function $C(x,a,y)$ which determines the cost of moving from state x to state y using action a. ($C(x,a,y) = \infty$ if a does not yield y from x). Note that different actions might generate the same move of $x \rightarrow y$.

Algorithms for Search

Output:

- a sequence of states leading from the initial state to a state satisfying the goal test.
- The sequence might be, optimal in cost for some algorithms, optimal in length for some algorithms, come with no optimality guarantees from other algorithms.

Algorithms for Search

Obtaining the action sequence.

- The set of successors of a state x might arise from different actions, e.g.,
  - $x \rightarrow a \rightarrow y$
  - $x \rightarrow b \rightarrow z$
- Successor function S(x) yields a set of states that can be reached from x via **any** single action.
  - Rather than just return a set of states, we annotate these states by the action used to obtain them:
    - S(x) = {<y,a>, <z,b>}
      y via action a, z via action b.
    - S(x) = {<y,a>, <y,b>}
      y via action a, also y via alternative action b.

Template Search Algorithms

- The search space consists of **states** and actions that move between states.
- A **path** in the search space is a **sequence** of states connected by actions, $<s_0, s_1, s_2, \dots, s_k>$,
  for every s_i and its successor s_{i+1} there must exist an action a_i that transitions s_i to s_{i+1}.
  - Alternately a path can be specified by
    (a) an initial state s_0, and
    (b) a sequence of actions that are applied in turn starting from s_0.

- The search algorithms perform search by examining alternate paths of the search space. The objects used in the algorithm are called **nodes**—each node contains a path.

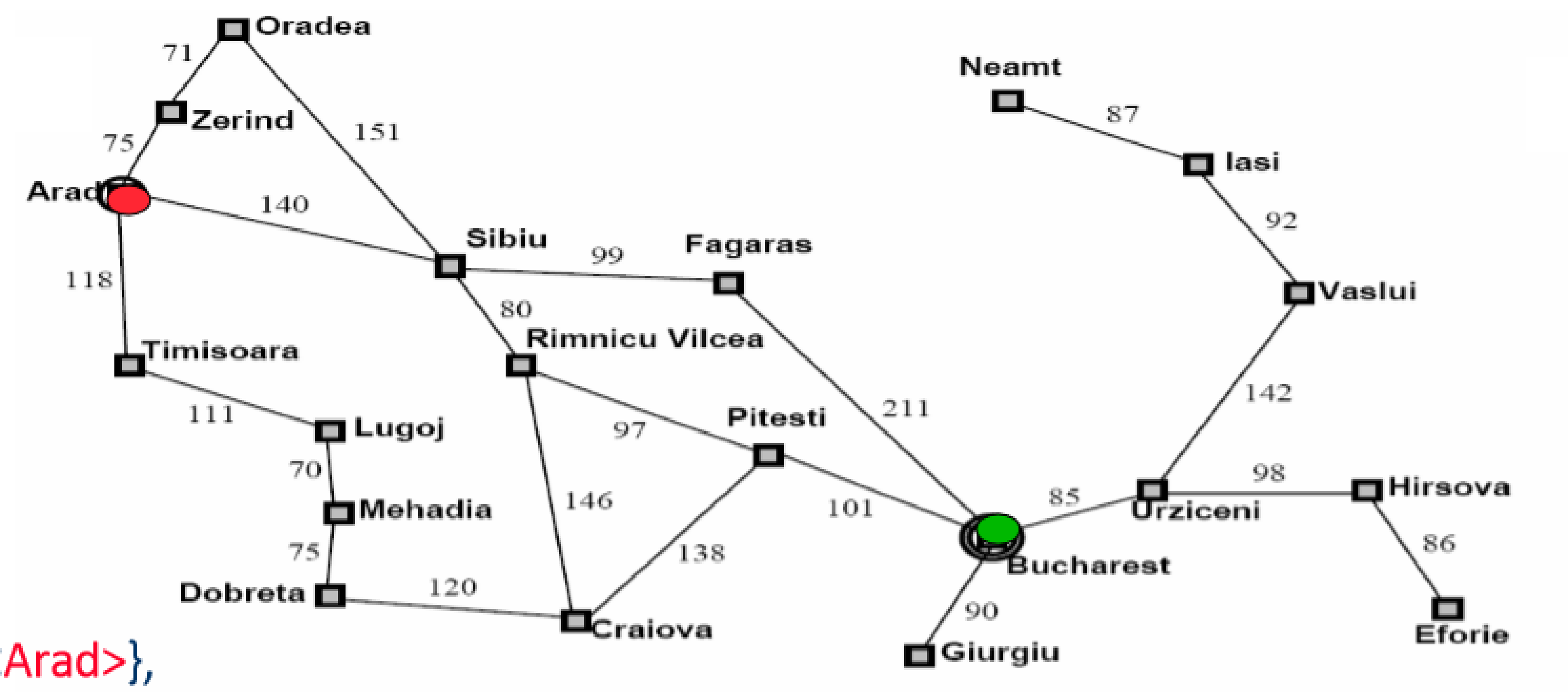
Template Algorithm for Search

- We maintain a set of Frontier nodes also called the OPEN set.
  - These nodes are paths in the search space that all start at the initial state.
- Initially we set OPEN = {<Start State>}
  - The path (node) that starts and terminates at the start state.
- At each step we select a node n from OPEN. Let x be the state n terminates at. We check if x satisfies the goal, if not we add all extensions of n to OPEN (by finding all states in S(x)).

Template Algorithm for Search

```
Search(OPEN, Successors, Goal? )
    While(Open not EMPTY) {
        n = select node from OPEN
        Curr = terminal state of n
        If (Goal?(Curr)) return n.
        OPEN = (OPEN– {n}) U_{s ∈ Successors(Curr)} <n,s>
          /* Note OPEN could grow or shrink */
    }
    return FAIL
```

When does OPEN get smaller in size?



{<Arad>},

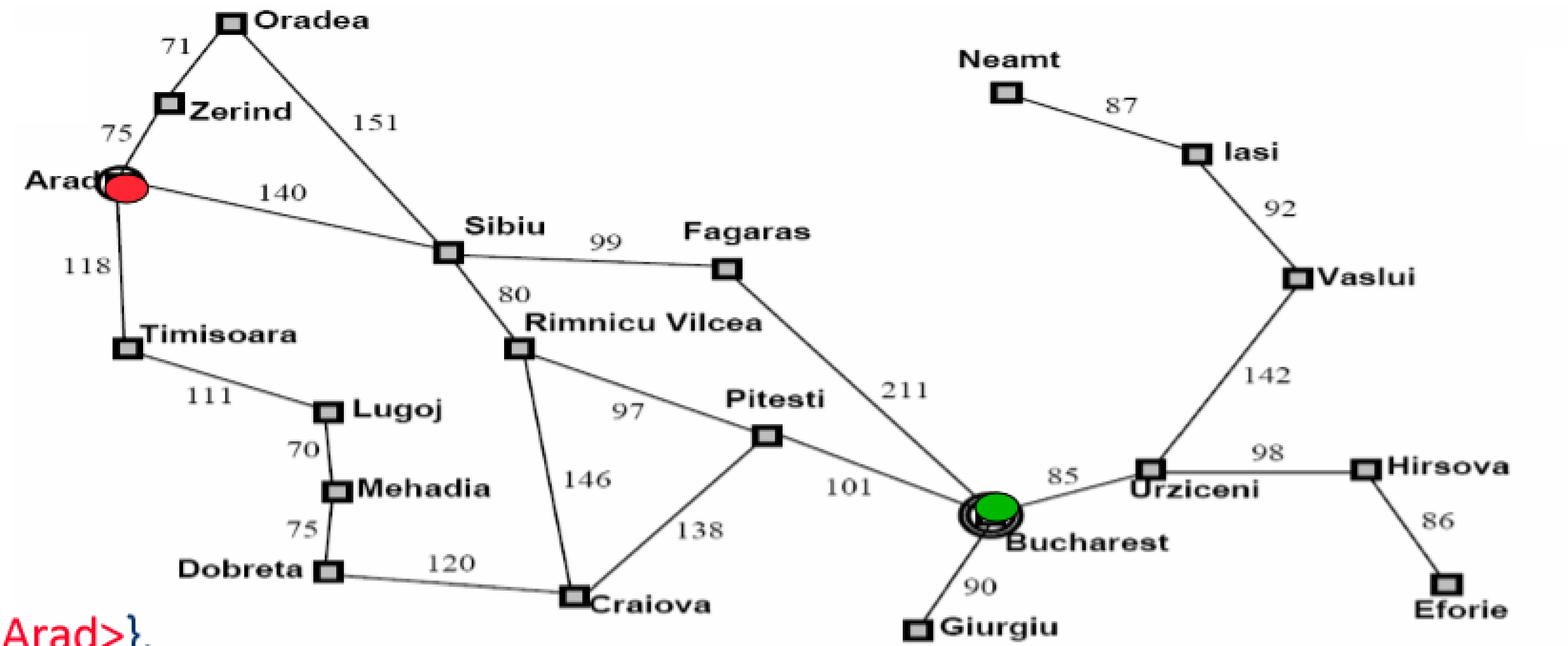
{**<A,Z>, <A,T>, <A, S>**},

{<A,Z>, <A,T>, **<A,S,A>, <A,S,O>, <A,S,F>, <A,S,R>**}

{<A,Z>, <A,T>, <A,S,A>, <A,S,O>, <A,S,R>, **<A,S,F,S>, <A,S,F,B>**}

Solution: Arad -> Sibiu -> Fagaras -> Bucharest

Cost: 140 + 99 + 211 = 450



{<Arad>},

{**<A,Z>, <A,T>, <A, S>**},

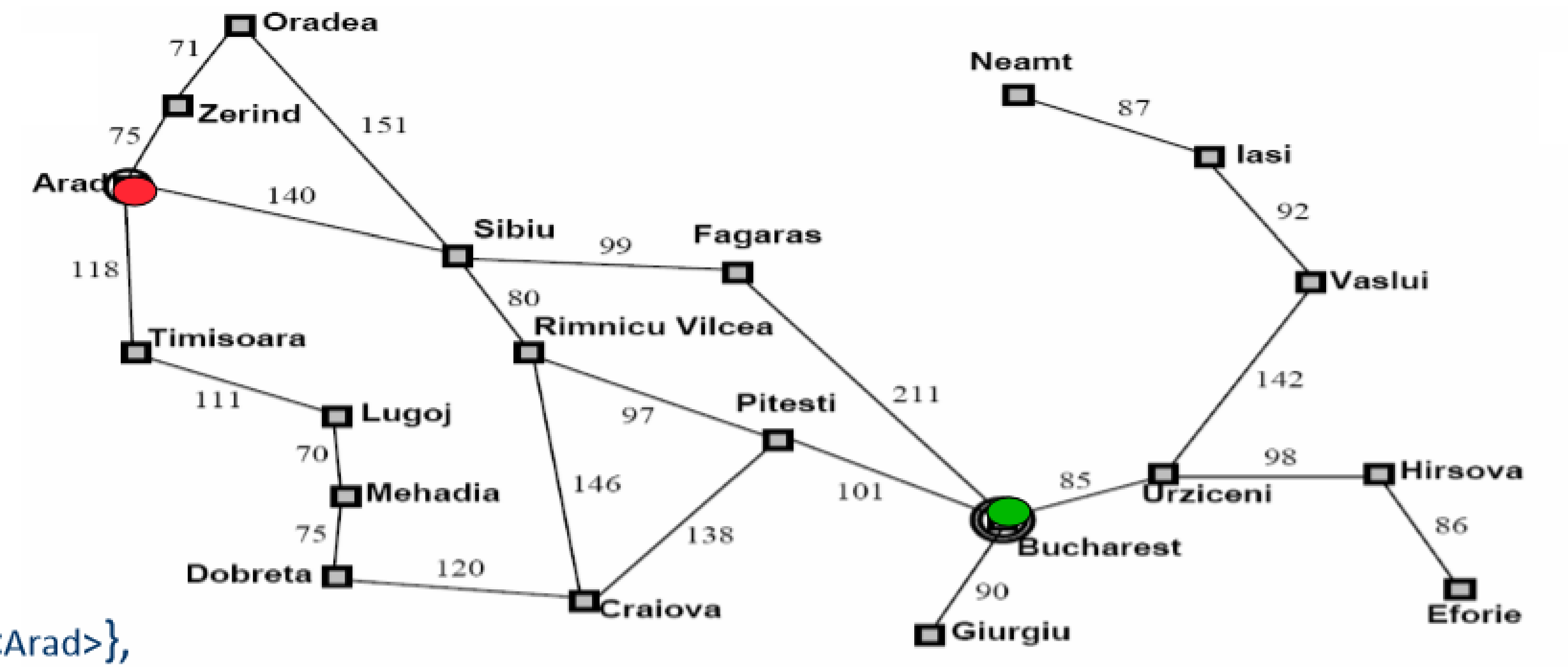
{<A,Z>, <A,T>, **<A,S,A>, <A,S,O>, <A,S,F>, <A,S,R>**}

{<A,Z>, <A,T>, <A,S,A>, <A,S,O>, <A,S,F>, **<A,S,R,S>, <A,S,R,C>, <A,S,R,P>**}

{<A,Z>, <A,T>, <A,S,A>, <A,S,O>, <A,S,F>, <A,S,R,S>, <A,S,R,C>, **<A,S,R,P,R>**, **<A,S,R,P,C>**, <A,S,R,P,B>}

Solution: Arad -> Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucharest

Cost: 140 + 80 + 97 + 101 = 418



{<Arad>},

{<A,Z>, <A,T>, <A, S>},

{<A,Z>, <A,T>, **<A,S,A>,** <A,S,O>, <A,S,F>, <A,S,R>}

{<A,Z>, <A,T>, **<A,S,A>**, <A,S,O>, <A,S,R>, **<A,S,F,S>**, <A,S,F,B>}

.....

cycles can cause non-termination!

Selection Rule

The order paths are selected from OPEN has a critical effect on the operation of the search:

- Whether or not a solution is found
- The cost of the solution found.
- The time and space required by the search.

How to select the next path from OPEN?

All search techniques keep OPEN as an ordered set (e.g., a priority queue) and repeatedly execute:

- If OPEN is empty, terminate with failure.
- Get the **next** path from OPEN.
- If the path leads to a goal state, terminate with success.
- Extend the path (i.e. generate the successor states of the terminal state of the path) and put the new paths in OPEN.

Critical Properties of Search

- Completeness: will the search always find a solution if a solution exists?
- Optimality: will the search always find the least cost solution? (when actions have costs)
- Time complexity: what is the maximum number of nodes (paths) than can be expanded or generated?
- Space complexity: what is the maximum number of nodes (paths) that have to be stored in memory?