QUESTION 1

Problem	**Initial state**	**Goal test**	**Actions**	**Cost function**
Scenario 1 ***Map coloring***	No regions colored	All regions colored, adjacent ones = different colors	Assign color to region	Number of assignments
Scenario 2 ***Monkey & banana***	As described in question	Monkey gets bananas	Hop on/off crate, push crate, walk, grab bananas	Number of actions
Scenario 3 ***Water measuring***	Jugs have value [0, 0, 0]	A jug with exactly 1 gallon of water	Changing [x, y, z] by dumping or filling water and passing water (*)	Number of actions

(*): This can be described more mathematically vigorously. For example, the act of dumping water changes from jug #1 turns [x, y, z] to [0, y, z]. Do this for all actions possible in order to obtain a formal formulation of the problem to be implemented by code.

All scenarios have **deterministic, fully observable states**, hence are **single-state problems**. In general, within the scopt of this course, we will work with this kind of problem.

QUESTION 2

(The following answers are in the case where the child nodes (cities) on each depth level is considered in ***alphabetical order****. A different ordering can lead to a different result in the Breadth-first search and Depth-first search algorithms)*

(Note that these are the solution paths, not the entire traversal. In actuality, before finding these paths, the algorithm may already traverse other nodes)

1. Breadth-first search:

Lugoj → Mehadia → Drobeta → Craiova → Pitesi → Bucharest

2. Uniform-cost search:

Lugoj → Mehadia → Drobeta → Craiova → Pitesi → Bucharest – Cost: 503

3. Depth-first search

Lugoj → Mehadia → Drobeta → Craiova → Pitesi → Bucharest

4. Iterative-deepening search

d = 1: Reached Timisoara, Mehadia (FAIL)

d = 2: Reached the above cities and Drobeta, Arad (FAIL)

d = 3: Reached the above cities and Zerind, Sibiu, Craiova (FAIL)

d = 4: Reached the above cities and Oradea, Fagaras, Rimnicu Vilcea, Pitesti (FAIL)

d = 5: Reached Bucharest (SUCCESS)

QUESTION 3

(The following answers are in the case where the child nodes (cities) on each depth level is considered in ***alphabetical order****. A different ordering can lead to a different result in the Breadth-first search (BFS) and Depth-first search (DFS) algorithms)*

(Note that these are the solution paths, not the entire traversal. In actuality, before finding these paths, the algorithm may already traverse other nodes)

1. Breadth-first search:

HN → ST → LC → V

2. Uniform-cost search:

HN → ND → NB → TH → V – Cost: 65

3. Depth-first search

HN → HB → HP → TH → V

4. Iterative-deepening search

d = 1: Reached ND, TB, HB, ST (FAIL)

d = 2: Reached ND, TB, HB, ST and LS, QN, HP, LC, NB (FAIL)

d = 3: Reached ND, TB, HB, ST and LS, HP, QN and TH, V (SUCCESS)

HN → HB → QN → V

QUESTION 4

- Placing the scene in a Cartesian plane, if we consider all (x,y) points to be possible states, then there is an **infinite** number of states and paths.

- We note that the shortest distance between two points is a straight line, which is possible in the case of no obstacles. When this is not possible due to obstacles, the optimal path is a set of **shorter straight lines** that deviate from the straight line by as little as possible. So the first segment of this sequence must go from the start point to a **tangent point** on an obstacle – any path that gave the obstacle a wider girth would be longer. Because the obstacles are **polygonal**, the **tangent points** must be at **vertices** of the obstacles, and hence the entire path must go from vertex to vertex

- Now, the state space is the set of vertices, of which there are **35**.

One possible solution (not necessarily optimal) is below.

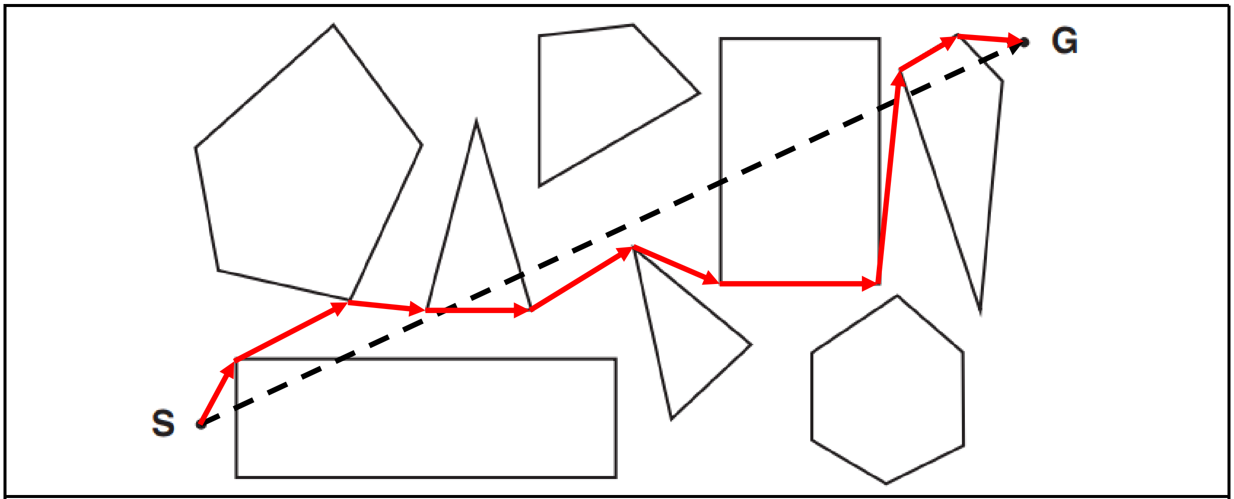

Figure 3.31 A scene with polygonal obstacles. S and G are the start and goal states.