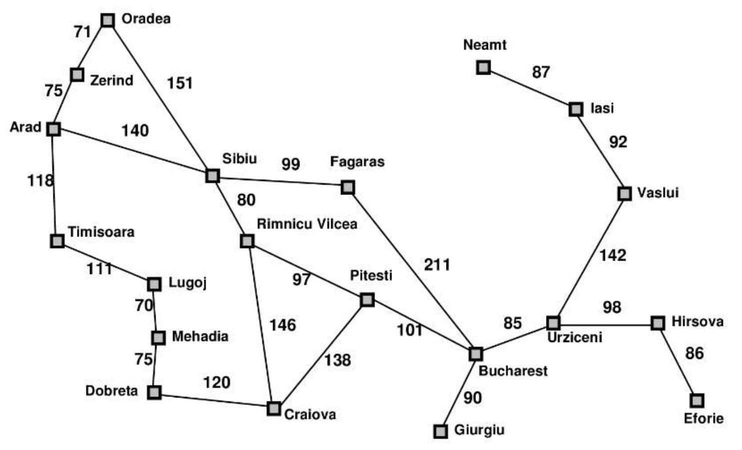
Lab 1: Searching Algorithms

Part 1: Breadth-first Search

In this part, we will explore breadth-first search algorithm.

In [1]:
```
# Don't mind this code. It is for showing images.

from IPython.display import display, Image, SVG, Math, YouTubeVideo
i = Image(filename='map.png')
display(i)
```



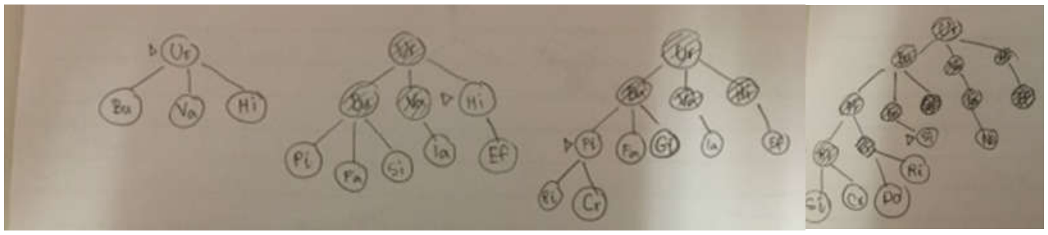
Straight–line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

The map above is a map of Romania. A square represents a city. The number between two squares is a distance between two cities. The numbers on the right are the distances from cities to Bucharest. We will use the numbers on the right in the next lab.

Perform breadth-first search from Urzecini to Sibiu and answer the following questions.

1. Draw out the search process.



2. What is the path from Urzecini to Sibiu?

   Urzecini -> Bucharest -> Fagaras -> Sibiu

3. Is there any shorter path in term of the distance?

   Yes

4. Is there any shorter path in term of steps?

   No

Part 2: Uniform-cost Search

Perform uniform-cost search from Urzecini to Sibiu and answer the following questions.

1. Draw out the search process.

Urziceni -> Bucharest	cost = 85 *	
Urziceni -> Vaslui	cost = 142 *	
Urziceni -> Hirsova	cost = 98 *	
Urziceni -> Bucharest -> Pitesti	cost = 85+101 = 186 *	
Urziceni -> Bucharest -> Fagaras	cost = 85+211 = 296 *	
Urziceni -> Bucharest -> Glurgiu	cost = 85+90 = 175	[NONE]
Urziceni -> Hirsova -> Eforie	cost = 98+86 = 184	[NONE]
Urziceni -> Vaslui -> lasi	cost = 142+92 = 234 *	
Urziceni -> Bucharest -> Pitesti -> Riminucu Vilcea	cost = 186+97 = 283 *	
Urziceni -> Bucharest -> Pitesti -> Craiova	cost = 186+138 = 324 *	
Urziceni -> Vaslui -> lasi -> Neamt	cost = 234 + 87 = 314	[NONE]
Urziceni -> Bucharest -> Pitesti -> Riminucu Vilcea -> Sibiu	cost = 283+80 = 363	[CHOOSE]
Urziceni -> Bucharest -> Pitesti -> Riminucu Vilcea -> Craiova	cost = 283+146 = 429	
Urziceni -> Bucharest -> Fagaras -> Sibiu	cost = 296+99 = 395	[END]
Urziceni -> Bucharest -> Pitesti -> Craiova -> Dobreta	cost = 324+120 = 444	
Urziceni -> Bucharest -> Pitesti -> Craiova -> Riminucu Vilcea	cost = 324+97 = 421	

2. What is the cost function in this problem?

   Distances

3. What is the path from Urzecini to Sibiu?

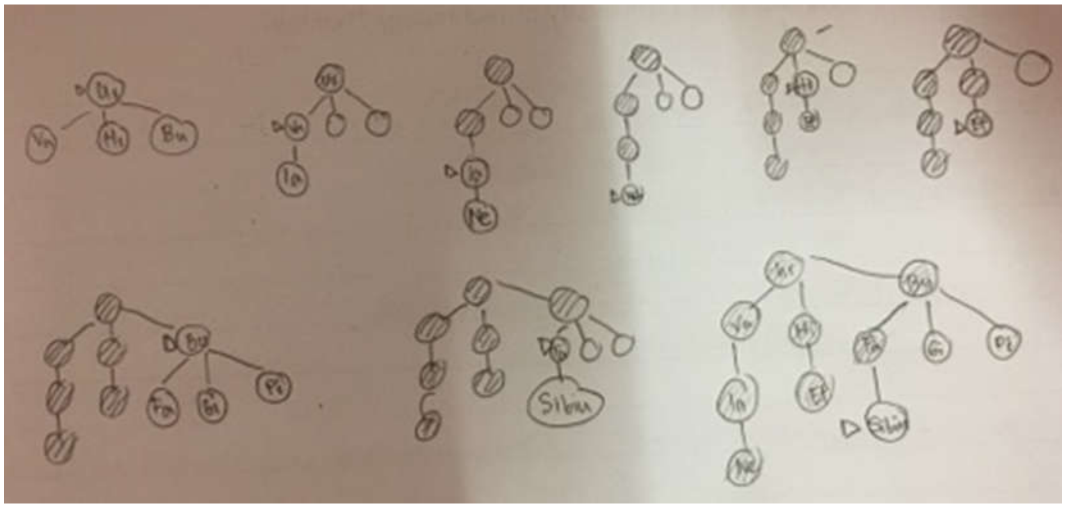
   Urzecini -> Bucharest -> Pitesti -> Rimnicu Vilcea -> Sibiu

Part 3: Depth-first Search

In this part, we will explore depth-first search algorithm.

Perform depth-first search from Urzecini to Sibiu. Branching order is North-East-West-South. Answer the following questions.

1. Draw out the search process.



2. What is the path from Urzecini to Sibiu?

   Urzecini -> Bucharest -> Fagaras -> Sibiu

3. Is there any shorter path in term of the distance?

   Yes

4. Is there any shorter path in term of steps?

   No

Part 4: Summary

Answer these questions.

1. Comparing 3 algorithms, analyze the size of the search

   UCS > BFS > DFS

2. How big are the trees for each algorithm?

   BFS -> 8 subtree

   DFS -> 5 subtree

   UCS -> 8 subtree

3. Which algorithm outputs the shortest path?

   DFS, BFS

4. Compare and contrast the algorithms.

   BFS: จะเป็นการ search และ expand ไปเรื่อยๆ ทุกๆ child

   DFS: expand lowest node First จนกระทั่ง expand ไม่ได้จึงกลับมา expand ที่ child ถัดไป

   UCS: สนใจเฉพาะ total cost ไม่สนใจจำนวน path

Part 5: Path Planning Simulation

1. Download 3 files (Agent.py, Sim.py, and World.py) from FRA311/Uninformed_Search Github.
2. Run Sim.py. Observe the behavior.
3. Analyze the behavior and answer what is this algorithm? Why?

   BFS because it searches and expand every child
4. Look at the code and understand it. Ignore the pygame part.

   Visit คือ ช่องที่ search ไปแล้ว

   Current คือ ตำแหน่งที่กำลังจะ search

   Neighbors คือ child ที่เป็นไปได้ของ parent นั้นๆ โดยจะเป็น queue แบบ FIFO เพื่อให้เป็นการ search แบบ BFS

   Code จะมีการทำงานเป็น loop โดยมีเงื่อนไขคือ สมาชิกใน queue จะต้องมากกว่า 0 ถ้า เท่ากับ 0 หมายความว่า search ไม่เจอ แต่ถ้า search เจอจะทำการ break ออกจาก loop

Part 6: Coding Part

Modify the code so it performs Depth-first Search, and print out the final path.

Hint

1. Try using a Priority Queue instead of a Queue.
2. What to do with the infinite loop? (Like we discuss in class)
3. If we want to print out a path, how do you keep track of the path? (Hint: Dictionary)
4. Try expanding the algorithm to Iterative Deepening Depth-first Search.