Akshay Nalwaya
200159155

ASSIGNMENT #01

1. **[20 points] Suppose we have an agent that works to make an office building 'intelligent'. This means that the agent is responsible for automatically heating, cooling, and lighting areas of the building where there are people present. Sensors can tell the agent where people are currently. The agent must turn lights on when people enter a room and turn them off again within 2 minutes of the people leaving the room. The agent must keep the areas with people currently present at an appropriate temperature, say within a set four-degree range. If the room is too cold or too hot for more than 20 minutes, people will leave and the company will lose money. Heating and cooling rooms takes some time, depending on the size of the room. Therefore, the room may not be within the four degree range when a person enters, but should get within the range before the 20-minute time limit. Answer the following questions about the agent.**

   a. **Define a PEAS (R&N Ch. 2) specification for the agent.**

   Ans. PEAS specification for the agent:
   Performance Measure – Room temperature, rate of heating/cooling, safety provisions, minimize fluctuations, maximize satisfaction among people
   Environment – Rooms, people, electrical equipment (lights, etc.), outside temperature, operators
   Actuators – Heating/Cooling system, light controls, display unit, speaker
   Sensors – motion sensor, temperature control, light sensor

   b. **Is it sufficient for the agent to be simple reflex? Why or why not?**

   Ans. No, it is not sufficient for agent to be simple reflex. It would have following drawbacks:
   - The decision to heat or cool the room will not be based on any parameter but only the current room temperature. This would have been a good choice if the only purpose was to control the lighting in rooms as there are only on/off options there which can done with condition-action rules.
   - System should be capable of estimating the time required to bring the room to desired temperature and hence aim to maintain a balance between resources and cost incurred to achieve the goal, which is definitely not possible in simple-reflex agents.

   c. **Would it be beneficial for the agent's performance if it randomly heated or cooled rooms where there are no people currently? Identify possible disadvantages to this sort of random action.**

   Ans. Randomly heating or cooling the rooms would not increase the performance of system, rather randomness might reduce it due to improper use of energy and resources.

- If large room with no people gets randomly heated, and then some people plan to meet there, the system will require to do twice the efforts if there were no such random policy in place: first to bring it back to normal temperature and then to cool if further.
- If the room that is randomly cooled/heated remains unoccupied for the whole day, all the resources and energy invested by system will get wasted.

d. **Suggest one improvement to the agent design. Since every improvement carries drawbacks, what are the drawbacks to yours?**

Ans.
Suggested Modification: System should keep a record of size of rooms and also track how frequently is room is occupied for meetings/gatherings. Having this data will facilitate predicting the chances of a room being occupied and system can then prioritize its actions.
Possible shortcoming: If there is an event which is not very frequent (for e.g., quarterly meet) but large people gather, or there is a long break, during which office will have very few people working, so, in such cases, the system would not be able to pre-determine the actions and might require human intervention to take suitable actions.

2. **[10 points] Answer the following questions based on the reading materials provided about IBM's Watson.**

a. **Describe a PEAS specification for Watson.**

Ans. PEAS Specification for Watson:
Performance Measure – Precision, Confidence, Answering Speed, Clue Selection, and Deciding optimum betting amount
Environment – Competitors, Question, Question Category, Value of question
Actuators – Buzzer, Voice Generator, keyboard
Sensors – Voice/Audio sensor, light sensor

b. **Describe Watson's environment. (Fully/partially observable, Deterministic/stochastic, etc.)**

Ans. Watson's environment can be described as:

- *Fully Observable* - Watson has complete information about environment state at any point of time. All the actions by others are also visible.
- *Multiagent* - It's a competitive multiagent environment as there are other people playing jeopardy and it has to maximize its performance in order to move ahead of other players.
- *Stochastic* - The outcomes are not solely determined by actions of Watson, it is affected by responses from other players also.
- *Sequential* - The response in any step might affect the overall earnings and future earnings. Also, while choosing the daily double wager in case of jeopardy, watson uses algorithms to determine the amount.

- *Dynamic* - Watson will have to keep a look at the environment while it's thinking as it is possible that while finding a response, someone else might have already pressed the buzzer. Hence, environment is dynamic.
- *Discrete* - The question is fed as an input and the response is output. There are only two possibilities for a response to the question, i.e, it can be correct or incorrect.
- *Known* - Watson is fully aware of the outcomes of all the actions and it knows how the system/game works and proceeds.

c. **Discuss at least three separate aspects of the Jeopardy problem domain together with the hardware and/or software design choices in Watson that are rational given those problem aspects.**

Ans.

- Watson can processes data at the rate of 500 gbps facilitating parallel processing of information at high speeds and has very large memory to store all the information in RAM. All this ensures that Watson operates at a very high speed.
- Watson uses Natural Language Processing to understand the questions and then comes up with candidate answers and doesn't rely on unstructured data, so it can answer a wide range of question categories.
- Watson keeps learning the ways to interpret clues which in turn changes the confidence value for answers and hence increases accuracy. Also, it only hits the buzzer if the confidence level for at least one answer is greater than a threshold value.

d. **Describe the DeepQA approach developed for Jeopardy and name the six architectural roles that are designed in this model.**

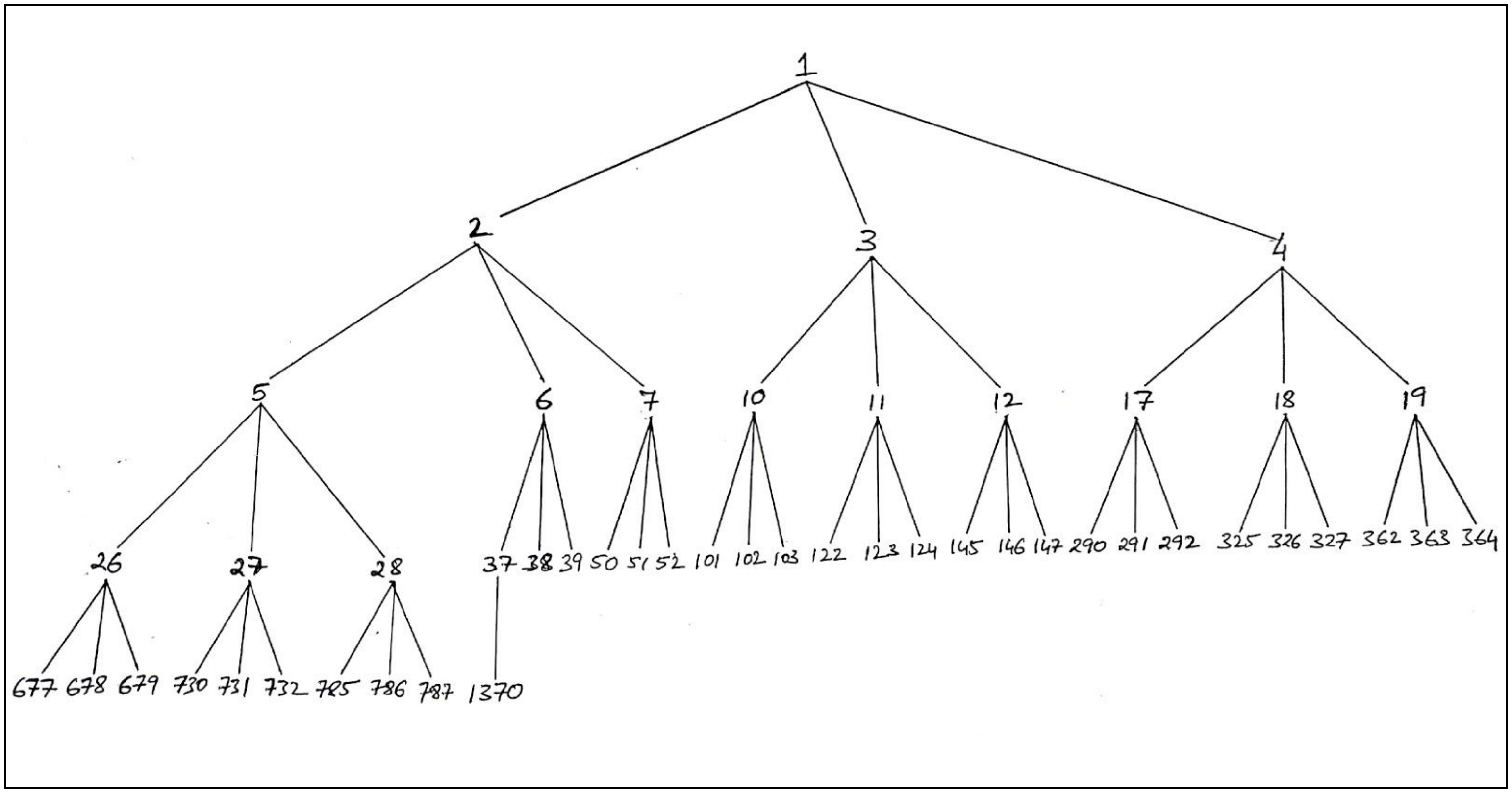
Ans. DeepQA Approach:

DeepQA is a parallel, probabilistic, evidence-based software architecture built for QA tasks. It uses several techniques and mechanisms to find answers for the questions/clues given during the game. Results of all these approaches are merged to combine them and give a result with high accuracy, at a fast speed and acceptable confidence. Watson, through DeepQA, finds a large number of evidences, for both supporting and refuting an answer and then combines the result to reach a conclusion. As a result of massive parallelism, many experts, very precise confidence estimation, and integration of shallow and deep knowledge, DeepQA ensure precise and accurate answer at high speed.

Architectural Roles in DeepQA -

1. Content Acquisition
2. Question Analysis
3. Hypothesis Generation
4. Soft Filtering
5. Hypothesis and Evidence Scoring
6. Merging, Ranking and Confidence Estimation

3. **[20 points] Consider a state space where the start state is labeled 1 to each state k has three successors: labeled (k^2) + 1, (k^2) + 2, and (k^2) + 3 respectively. Draw the portion of the state space for states 1 to 1370.**



a. **Suppose the goal state is 101. List the order in which states will be visited for the breadth-first search.**

Ans. Breadth-First Search traversal:
1 → 2 → 3 → 4 → 5 → 6 → 7 → 10 → 11 → 12 → 17 → 18 → 19 → 26 → 27 → 28 → 37 → 38 → 39 → 50 → 51 → 52 → 101

b. **Suppose the goal state is 101. List the order in which states will be visited for the depth-limited search with limit 3.**

Ans. Depth-Limited Search traversal: (Limit = 3) *[Assumed root node at level = 1]*
1 → 2 → 5 → 6 → 7 → 3 → 10 → 11 → 12 → 4 → 17 → 18 → 19
Goal node not found as we have reached the depth limit (= 3) but goal node is not present within this limit.

c. **Suppose the goal state is 101. List the order in which states will be visited for the iterative deepening search with initial cutoff 1 and cutoff increment 1.**

Ans. Iterative Deepening Search traversal:
1 → 1 → 2 → 3 → 4 → 1 → 2 → 5 → 6 → 7 → 3 → 10 → 11 → 12 → 4 → 17 → 18 → 19 → 1 → 2 → 5 → 26 → 27 → 28 → 6 → 37 → 38 → 39 → 7 → 50 → 51 → 52 → 3 → 10 → 101

4. **[10 points] Which agent structure is appropriate for each of the following agents, considering their task environment? Give a brief explanation for your answer.**

a. **Room temperature controller**

Ans. *Utility-based agent*
Because in addition to having the goal of bringing the room temperature to the desired level, it is also the responsibility of this controller to manage resources efficiently and perform the operation using resources judiciously.

b. **Taxi driver**

Ans. *Learning Agent*
If the taxi driver goes to a route which is in some manner different from the routine routes, it will have to learn the environment and then use those learnings to take decisions. Thus, in order to give best performance, it should have a goal, achieve that in optimized manner and also learn from the environment.

c. **Tic-tac-toe player**

Ans. *Simple-Reflex Agent*
It is a fully observable environment so all moves are visible to everyone. Also, there are no infinite loops as there are a limited number of turns that each player can take owing to the 9 squares on board. Therefore, we can say that decision for each turn is taken based on current scenario and there is no use of past moves by a player or opponent.

d. **Face recognition system**

Ans. *Goal-based*
Face recognition system might have a number of small goals in order to achieve the final goal of identifying the face, hence, for each step it must prefer actions that get the system to its final goal.

5. **[40 points] Here is a road map of Romania similar to R&N Fig. 3.2 [map]. (The numbers on the edges indicate the distance between the cities connected, but you don't need these distances until the next assignment.)**
In a language of your choice (Java, Python, or C++), implement the Depth-First Search and Breadth-First Search algorithms. Your code should keep track of nodes expanded and should be able to compute the length of this list. Then run

your algorithms on the Romanian road map. To save a bit of typing, you may use this file for the cities and roads: [roads.pl]. This is a Prolog source file, and this assignment does not use Prolog, so you will have to modify it for use with your code. Notice also that Assignment 1 does not use the road distances, or the longitude/latitude of the cities.

a. **[10 points] Consider the path from Fagaras to Dobreta and the path from Dobreta to Fagaras. Run your algorithms and show the paths returned by DFS and BFS results for each case. How do the solution paths compare for the two algorithms? Give an explanation for what you observe.**

Ans.

Outputs:

DFS - Fagaras to Dobreta

```
Nodes traversed: fagaras bucharest giurgiu pitesti craiova
dobreta
Nodes Expanded: 6
DFS Path: fagaras bucharest pitesti craiova dobreta
```

BFS - Fagaras to Dobreta

```
Nodes traversed: fagaras bucharest sibiu giurgiu pitesti
urziceni arad oradea rimnicu_vilcea craiova hirsova vaslui
timisoara zerind dobreta
Nodes Expanded: 17
BFS Path: fagaras bucharest pitesti craiova dobreta
```

DFS - Dobreta to Fagaras

```
Nodes traversed: dobreta craiova pitesti bucharest fagaras
Nodes Expanded: 5
DFS Path: dobreta craiova pitesti bucharest fagaras
```

BFS - Dobreta to Fagaras

```
Nodes traversed: dobreta craiova mehadia pitesti
rimnicu_vilcea lugoj bucharest sibiu timisoara fagaras
Nodes Expanded: 13
BFS Path: dobreta craiova pitesti bucharest fagaras
```

Output Analysis - Since Fagarus is not very near to Dobreta, BFS has to expand and traverse a greater number of nodes to reach Dobreta, but DFS, traversing till the leaf node for each child node, gets to Dobreta in lesser number of node expansions.

b. **[10 points] Is there a case where Depth-First performs worse than Breadth-First (in terms of number of cities visited in the path, not the distance)? If yes, what is the case? If not, explain why.**

Ans.

When goal node is not too far from the source node, then it is better to use BFS, as DFS will go on expanding nodes till the leaf node which is not required.
For e.g., Dobreta to Mehadia, both are neighbouring cities but if we use DFS, then we have to expand the whole node from Craiova but in case of BFS, we can reach goal in the first iteration of search where we expand connecting nodes of Dobreta.

DFS - Dobreta to Mehadia

```
Nodes traversed: dobreta craiova pitesti bucharest fagaras
sibiu arad timisoara lugoj mehadia
Nodes Expanded: 10
DFS Path: dobreta craiova pitesti bucharest fagaras sibiu
arad timisoara lugoj mehadia
```

BFS - Dobreta to Mehadia

```
Nodes traversed: dobreta craiova mehadia
Nodes Expanded: 3
BFS Path: dobreta mehadia
```

c. **[10 points] Is there a case where Breadth-First performs worse than Depth-First (in terms of number of cities visited in the path, not the distance)? If yes, what is the case? If not, explain why.**

Ans.

When the goal city is deep in the graph from the source city, in such cases, BFS will unnecessarily expand each node at each level and consume a lot of memory, while in DFS it will simply go to the leaf node in each case to search for the goal city which will occupy lesser memory.
For e.g., Oradea to Timisoara
Output of DFS - Oradea to Timisoara

```
Nodes traversed: oradea sibiu arad timisoara
Nodes Expanded: 4
DFS Path: oradea sibiu arad timisoara
```

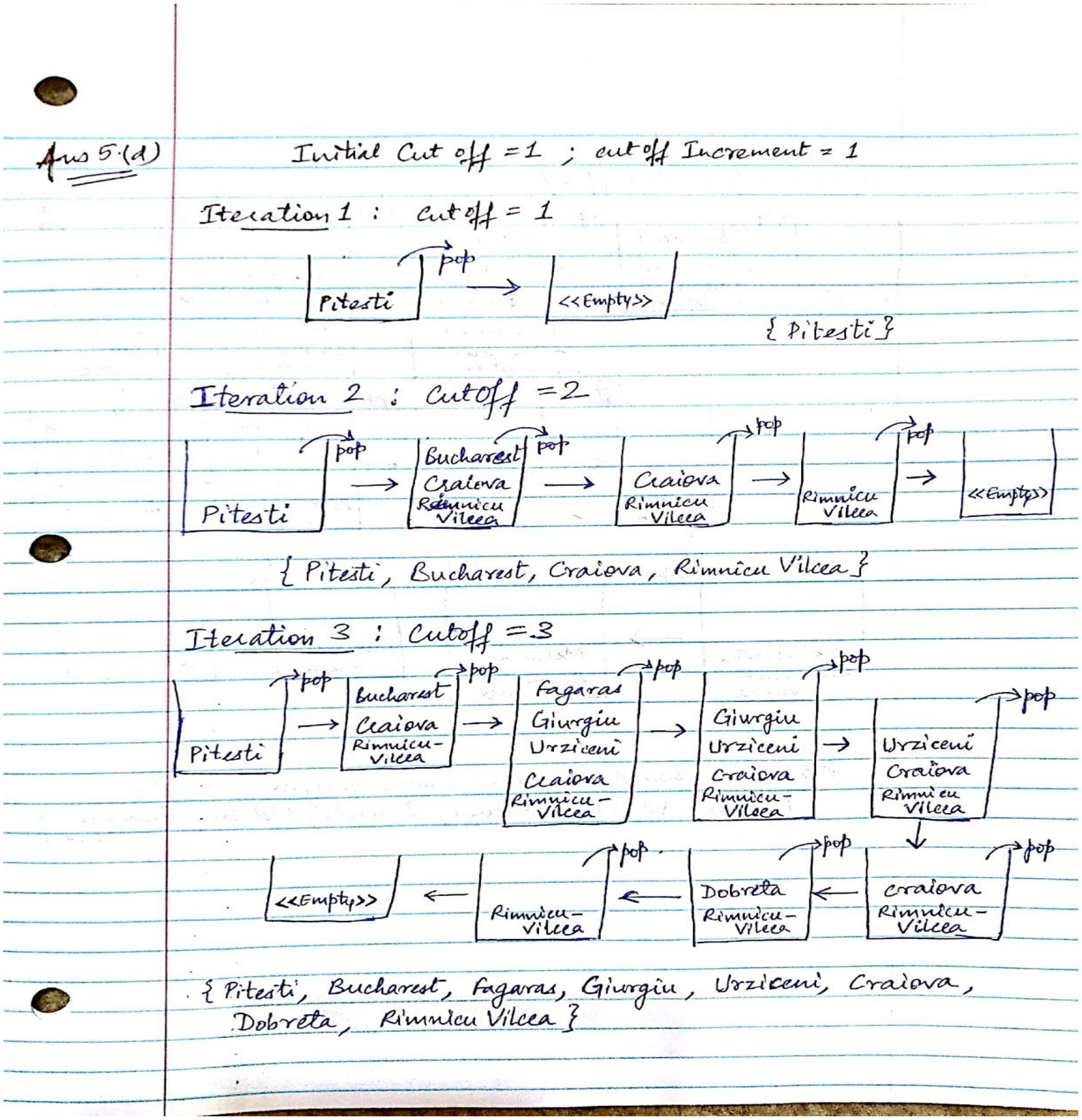
Output of BFS - Oradea to Timisoara

```
Nodes traversed: oradea sibiu zerind arad fagaras
rimnicu_vilcea timisoara
Nodes Expanded: 8
BFS Path: oradea sibiu arad timisoara
```

d. **[10 points] For the same graph, perform a hand-execution of Depth-First Iterative Deepening (DFID) with increment and cutoff initialized to 1, starting at Pitesti. List the nodes in the order expanded for the first five**

iterations of DFID, and the state of the datastructure (stack) after each iteration. Expand the nodes alphabetically and insert them in nondecreasing alphabetical order. How does this list compare with the list of expansions in Breadth-First Search?

Ans.



Ans 5.(d)

Initial Cut off = 1 ; cut off Increment = 1

Iteration 1 : Cut off = 1

Pitesti
pop
<<Empty>>

{ Pitesti }

Iteration 2 : Cutoff = 2

Pitesti
pop
Bucharest
Craiova
Rimnicu Vilcea
pop
Craiova
Rimnicu Vilcea
pop
Rimnicu Vilcea
pop
<<Empty>>

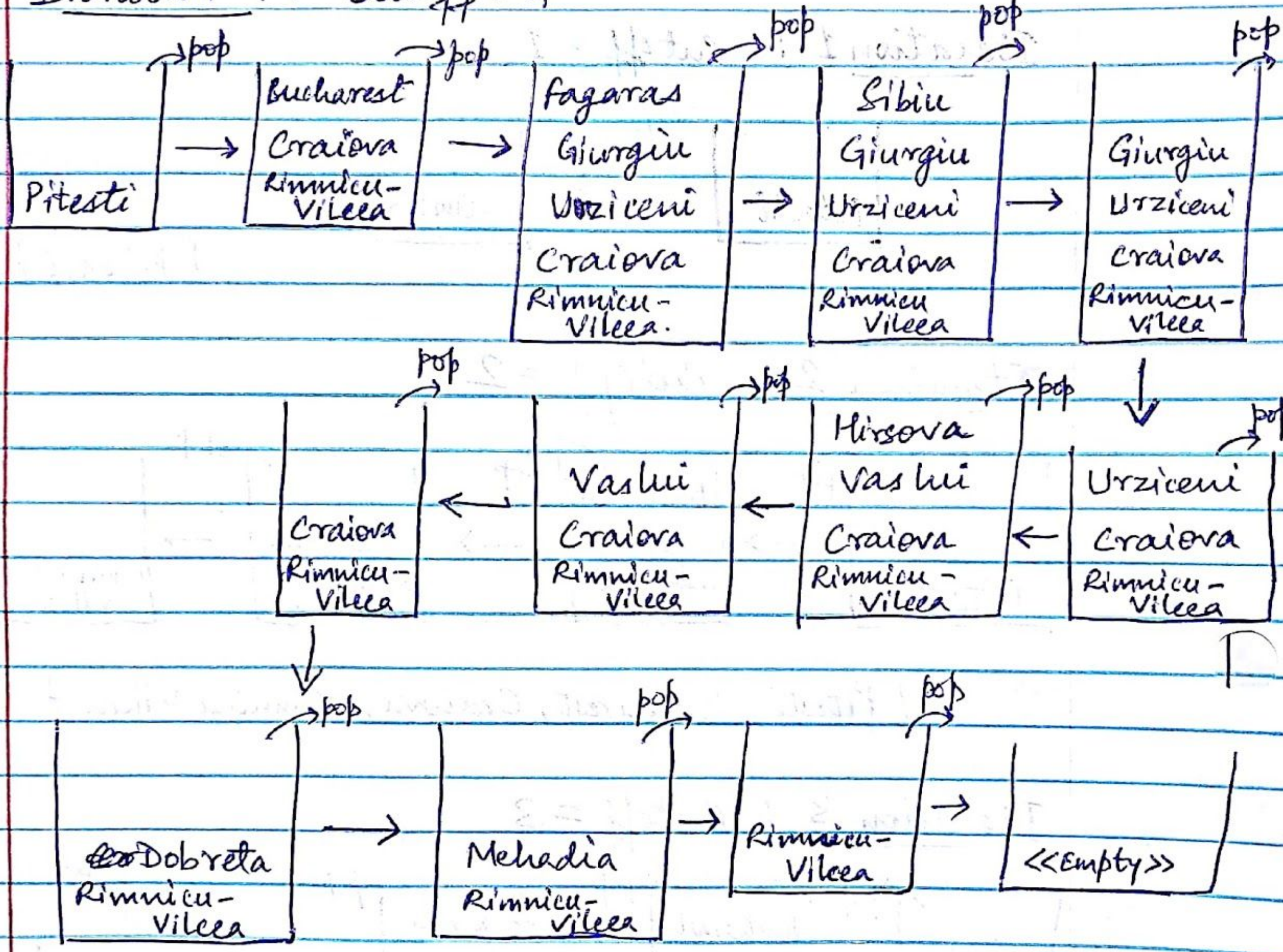
{ Pitesti, Bucharest, Craiova, Rimnicu Vilcea }

Iteration 3 : Cutoff = 3

Pitesti
pop
Bucharest
Craiova
Rimnicu-Vilcea
pop
Fagaras
Giurgiu
Urziceni
Craiova
Rimnicu-Vilcea
pop
Giurgiu
Urziceni
Craiova
Rimnicu-Vilcea
pop
Urziceni
Craiova
Rimnicu Vilcea
pop
Craiova
Rimnicu-Vilcea
pop
Dobreta
Rimnicu-Vilcea
pop
Rimnicu-Vilcea
pop
<<Empty>>

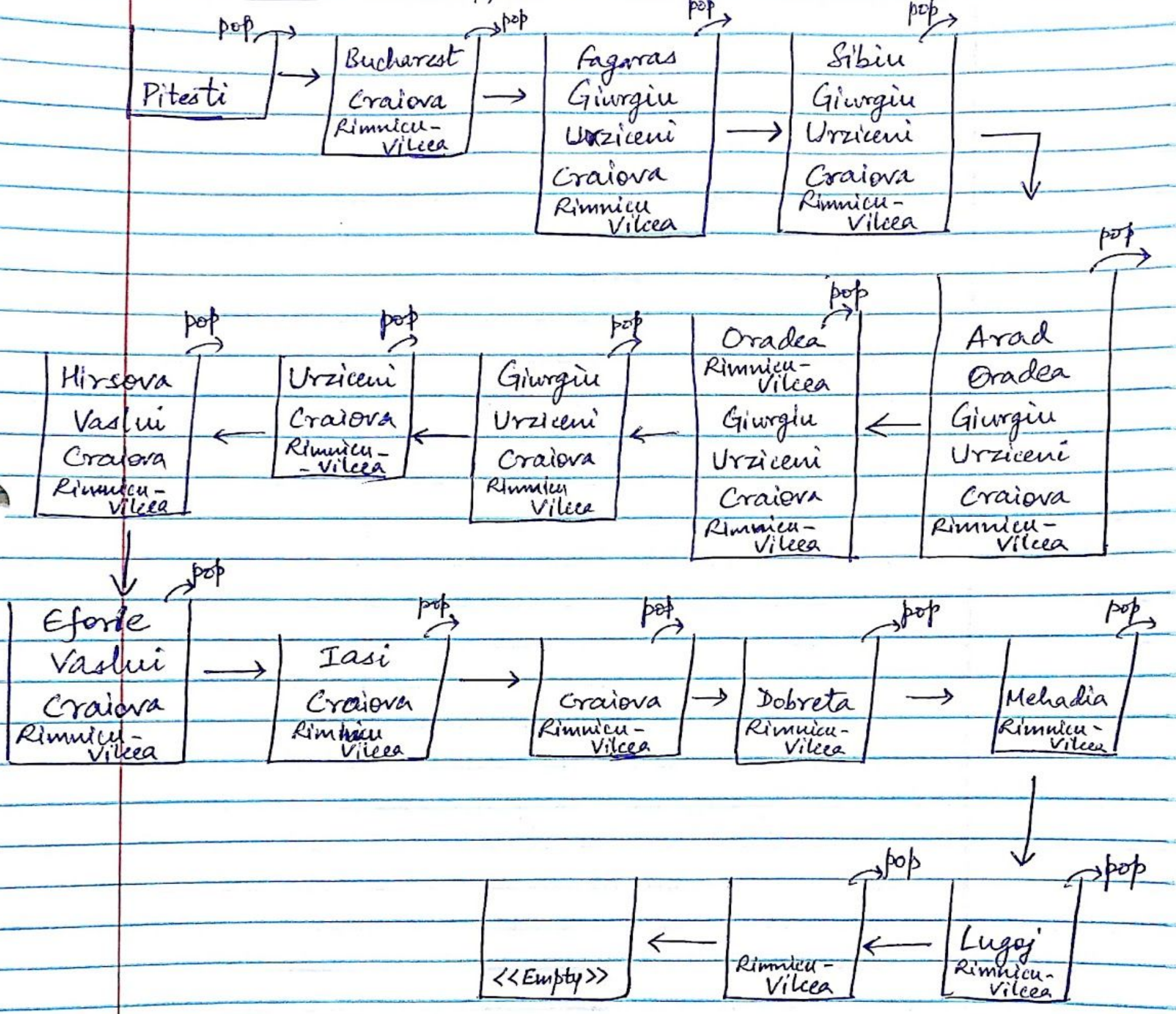
{ Pitesti, Bucharest, Fagaras, Giurgiu, Urziceni, Craiova, Dobreta, Rimnicu Vilcea }

Iteration 4 : Cutoff = 4



{ Pitesti, Bucharest, Fagaras, Sibiu, Giurgiu, Urziceni, Hirsova, Vaslui, Craiova, Dobreta, Mehadia, Rimnicu Vilcea }

Iteration 5 : Cutoff = 5



{Pitesti, Bucharest, Fagaras, Sibiu, Arad, Oradea, Giurgiu, Urziceni, Hirsova, Eforie, Vaslui, Iasi, Craiova, Dobreta, Mehadia, Lugoj, Rimnicu Vilcea }

BFS traversal path (starting from Pitesti):
`Pitesti → Bucharest → Craiova → Rimnicu Vilcea → Fagaras → Giurgiu → Urziceni → Dobreta → Sibiu → Hirsova → Vaslui → Mehadia → Arad → Oradea → Eforie → Iasi → Lugoj → Timisoara → Zerind → Neamt`

	Depth-First Iterative Deepening	**Breadth-First Search**
Number of nodes expanded	**42** (approx.)	**20**

In this case, BFS is better than DFID as it expands more number of nodes than BFS. Therefore, BFS will be preferred here.
But in a more general sense, BFS and DFID can be compared as follows:

- DFID uses less memory as compared to BFS approach
- In DFID approach, same nodes are re-visited many times but in BFS each node is visited only once
- On the other hand, BFS uses a large amount of memory and its memory usage is directly proportional to the width of tree/graph
- BFS consumes a lot of time in case of larger trees/graphs
- In BFS, if there are more than one solution, then we can find the minimal solution