Course: CSC520, Introduction to Artificial Intelligence
Homework 2
Student: Xusheng Xiao
Unity ID: xxiao2
Email: xxiao2@ncsu.edu

---

1. **(34 points) Consider the following English sentences.:**

   (a) John is a lawyer.

   (b) Lawyers are rich.

   (c) Rich people have big houses.

   (d) Big houses are a lot of work to maintain.

   (e) John has a house.

   Answer the following questions:

   (a) (10 points) Convert the sentences into first order predicate logic. Use the following lexicon:
   Functions: $houseof(X)$ – the house of X
   Predicates: $lawyer(X)$ – X is a lawyer
   $rich(X)$ – X is rich
   $big(X)$ – X is big
   $lotofwork(X)$ – X is a lot of work
   $house(X, Y)$ – the owner of house X is Y

      i. $lawyer(john)$
      ii. $\forall X(lawyer(X) \Rightarrow rich(X))$
      iii. $\forall X(rich(X) \Rightarrow house(houseof(X), X) \wedge big(houseof(X)))$
      iv. $\forall X(big(houseof(X)) \Rightarrow lotofwork(houseof(X)))$
      v. $house(houseof(john), john)$

   (b) (12 points) Convert the logic statements into CNF.
      i. $lawyer(john)$
      ii. $\neg lawyer(X1) \vee rich(X1)$
      iii. $\neg rich(X2) \vee house(houseof(X2), X2)$
      iv. $\neg rich(X3) \vee big(houseof(X3))$
      v. $\neg big(houseof(X4)) \vee lotofwork(houseof(X4))$
      vi. $house(houseof(john), john)$

   (c) (12 points) Using resolution, prove that "John's house is a lot of work to maintain."
   Conclusion: $house(houseof(john), john) \wedge lotofwork(houseof(john))$

Negated conclusion:
1. $\neg house(houseof(john), john) \vee \neg lotofwork(houseof(john))$

2.	$rich(john)$	$i + ii\{john/X\}$
3.	$house(houseof(john), john)$	$2 + iii\{john/X2\}$
4.	$big(houseof(john))$	$2 + iv\{john/X2\}$
5.	$lotofwork(houseof(john))$	$4 + v\{john/4\}$
6.	$\neg house(houseof(john), john)$	$5 + 1$
7.	$\emptyset$	$3 + 6$

2. **(24 points) Consider the following English sentences.**

- The College of Engineering is in Daniels Hall and the College of Business is in Nelson Hall.
- Computer Science and Civil Engineering are departments in the College of Engineering.
- Finance is a department in the College of Business.
- Smith is a professor in Computer Science, Jones in Civil Engineering, and Edison in Finance.

Now answer the following questions. Submit the final prolog code.

(a) (8 points) Convert the English sentences into a Prolog database.

```
in(ce,daniel).
in(cm,nelson).
department(cs,ce).
department(civil,ce).
department(finance,cm).
prof(smith,cs).
prof(jones,civil).
prof(edison,finance).
```

(b) (4 points) Write Prolog rules for each of the following statements and add the rules into the Prolog database.

- A faculty member of a department is also a faculty member in the corresponding College.

  ```
  prof(F,S) :- department(D,S),prof(F,D).
  ```

- The location of the department in the same as the location of corresponding College.

  ```
  in(D,L) :- department(D,C),in(C,L).
  ```

(c) (4 points) Now, write a query to find the location of Prof. Smith. and show the result. (Hint: the professor will be in same building as his/her department)

The query is : prof(smith,Dept),in(Dept,Location).

Dept = cs, Location = daniel ;

(d) (4 points) Add to the Prolog database a rule with head location(X, Y) that will display the location of a faculty given a query such as location(smith,Where).

location(F,L) :- prof(F,D),in(D,L).

(e) (4 points) Now write a single query for each of the following questions and show the results.

- Name the faculty in the College of Engineering.

  prof(Faculty, ce).

  Faculty = smith ;

  Faculty = jones ;
- List ALL the occupants of Daniels Hall.

  in(Occupant, daniel).

  Occupant = ce ;

  Occupant = cs ;

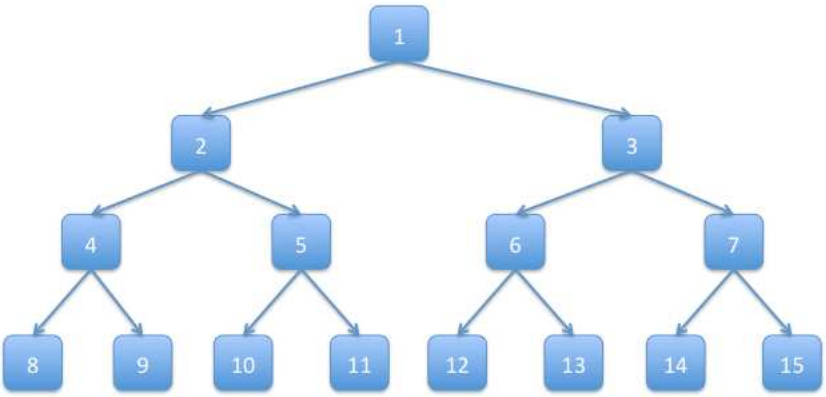
  Occupant = civil ;

3. **(12 points) Question 3.15 a, b, c, and d from Russell & Norvig, p. 116.**
   Consider a state space where the start state is number 1 and the successor function for state n returns two states, numbers 2n and 2n + 1.

   (a) Draw the portion of the state space for states 1 to 15.

   (b) Suppose the goal state is 11. List the order in which nodes will be visited for breadthfirst search, depth-limited search with limit 3, and iterative deepening search.

   - **Breadth First Search:**
     1,2,3,4,5,6,7,8,9,10,11



- **Depth-limited Search with Limit 3:**
  1,2,4,8,9, 5, 10, 11
- **Iterative Deepening Search:**
  - 1
  - 1,2,3
  - 1,2,4,5,3,6,7
  - 1,2,4,8,9,5,10,11

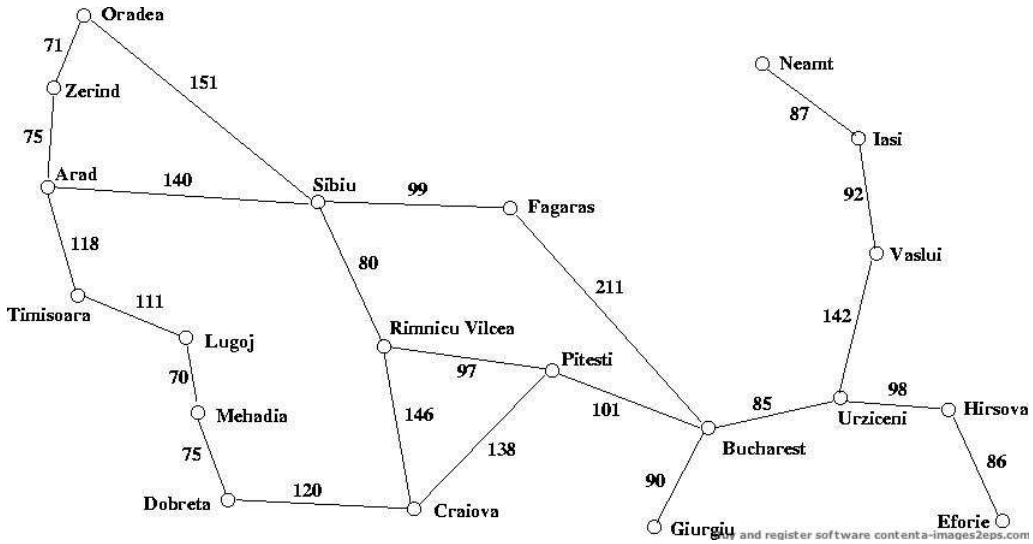
(c) Would bidirectional search be appropriate for this problem? If so, describe in detail how it would work.

I think bidirectional search is appropriate for this problem. We can apply breadth first search on both start state and goal state. Each direction searches one node at a time. Then the order of nodes visited would be: [1, 2, 3] , [11, 5, 2]. Since it meets at the node 2, the search finds a path.

(d) Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

Yes. If we reverse the start state and the goal state, then the start state would be 11 and goal state would be 1. From state 11, there is only one out going edge, which lead to 5. Similarly for 5 and 2, they only have one out going edge. Thus, there is only one path that lead to goal state 1 and we do not need to search.

4. **(30 points) Here is a road map of Romania map. (The numbers on the edges indicate the distance between the cities connected, but you don't distances them until Homework 3.).**



For this assignment, you may use the Prolog source in the webnotes for dfs.pl, bfs2.pl and roads.pl.

(a) (8 points) Consider the path from Arad to Bucharest and the path from Bucharest to Arad. Show the paths returned by DFS and BFS results for each case. Does either DFS or BFS return a path through the same set of cities for either? Give a reason for this behavior.

i. **Arad to Bucharest**

- **DFS**:
  - arad timisoara lugoj mehadia dobreta craiova rimnicu_vilcea pitesti bucharest
  - arad timisoara lugoj mehadia dobreta craiova rimnicu_vilcea sibiu fagaras bucharest
  - arad timisoara lugoj mehadia dobreta craiova pitesti rimnicu_vilcea sibiu fagaras bucharest
  - arad timisoara lugoj mehadia dobreta craiova pitesti bucharest
  - arad sibiu rimnicu_vilcea craiova pitesti bucharest
  - arad sibiu rimnicu_vilcea pitesti bucharest
  - arad sibiu fagaras bucharest
  - arad zerind oradea sibiu rimnicu_vilcea craiova pitesti bucharest
  - arad zerind oradea sibiu rimnicu_vilcea pitesti bucharest
  - arad zerind oradea sibiu fagaras bucharest
- **BFS**:
  - arad sibiu fagaras bucharest
  - arad sibiu rimnicu_vilcea pitesti bucharest

ii. **Bucharest to Arad**

- **DFS**:
  - bucharest pitesti craiova dobreta mehadia lugoj timisoara arad
  - bucharest pitesti craiova rimnicu_vilcea sibiu oradea zerind arad
  - bucharest pitesti craiova rimnicu_vilcea sibiu arad
  - bucharest pitesti rimnicu_vilcea craiova dobreta mehadia lugoj timisoara arad
  - bucharest pitesti rimnicu_vilcea sibiu oradea zerind arad
  - bucharest pitesti rimnicu_vilcea sibiu arad
  - bucharest fagaras sibiu rimnicu_vilcea craiova dobreta mehadia lugoj timisoara arad
  - bucharest fagaras sibiu rimnicu_vilcea pitesti craiova dobreta mehadia lugoj timisoara arad
  - bucharest fagaras sibiu oradea zerind arad
  - bucharest fagaras sibiu arad
- **BFS**:
  - bucharest fagaras sibiu aradtrue

DFS and BFS do return some paths through same set of cities. In such a undirectional graph, the set of paths found by DFS is the super set of the set of paths found by BFS, since BFS only finds shortest paths in a graph where the weight for each link in the graph is uniform.

(b) (6 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 it? If not, explain why.

Yes. In a path "arad timisoara lugoj mehadia dobreta craiova rimnicu_vilcea pitesti bucharest" found by DFS, BFS found a shorter path "arad sibiu fagaras bucharesttrue ".

(c) (6 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 it? If not, explain why.

No. Since in such a uniform weight graph, BFS always finds shortest paths.

(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 Oradea. List the nodes in the order expanded and the state of the data structure for the first five iterations of DFID. Expand the nodes alphabetically and insert them in nondecreasing

alphabetical order. Compare this list with Breadth-First Search. (No Prolog is required for the DFID portion of this question.).

- **Limit = 1.**
  - **expanded nodes:** Oradea
  - **data structure:**
    $[Oradea, Sibiu]$
    $[Oradea, Zerind]$
- **Limit = 2.**
  - **expanded nodes:** Oradea, Sibiu, Zerind
  - **data structure:**
    $[Oradea, Sibiu, Arad]$
    $[Oradea, Sibiu, Fagaras]$,
    $[Oradea, Sibiu, RimniceVilcea]$
    $[Oradea, Zerind, Arad]$
- **Limit = 3.**
  - **expanded nodes:** Oradea, Sibiu, Arad, Fagaras, Rimnice ViLCea, Zerind, Arad,
  - **data structure:**
    $[Oradea, Sibiu, Arad, Timisoara]$
    $[Oradea, Sibiu, Arad, Zerind]$
    $[Oradea, Sibiu, Fagaras, Bucharest]$,
    $[Oradea, Sibiu, RimniceVilcea, Craiova]$
    $[Oradea, Sibiu, RimniceVilcea, Pitesti]$
    $[Oradea, Zerind, Arad, Sibiu]$
    $[Oradea, Zerind, Arad, Timisoara]$
- **Limit = 4.**
  - **expanded nodes:** Oradea, Sibiu, Arad, Timisoara,Zerind, Fagaras, Bucharest, Rimnice ViLCea, Craiova, Pitesti, Zerind,Arad, Sibiu, Timisoara
  - **data structure:**
    $[Oradea, Sibiu, Arad, Timisoara, Lugoj]$
    $[Oradea, Sibiu, Arad, Zerind]$
    $[Oradea, Sibiu, Fagaras, Bucharest, Giurgiu]$
    $[Oradea, Sibiu, Fagaras, Bucharest, Pitesti]$
    $[Oradea, Sibiu, Fagaras, Bucharest, Urziceni]$
    $[Oradea, Sibiu, RimniceVilcea, Craiova, Dobreta]$
    $[Oradea, Sibiu, RimniceVilcea, Craiova, Pitesti]$
    $[Oradea, Sibiu, RimniceVilcea, Pitesti, Bucharest]$
    $[Oradea, Sibiu, RimniceVilcea, Pitesti, Craiova]$
    $[Oradea, Zerind, Arad, Sibiu, Fagaras]$
    $[Oradea, Zerind, Arad, Sibiu, RimniceVilcea]$
    $[Oradea, Zerind, Arad, Timisoara, Lugoj]$
- **Limit = 5.**

- **expanded nodes:** expanded nodes: Oradea, Sibiu, Arad, Timisoara,Luogj, Zerind, Fagaras, Bucharest, Giurgiu, Pitesti, Urziceni, Rimnice Vilcea, Craiova, Dobreta, Pitesti, Pitesti, Bucharest, Zerind,Arad, Sibiu, Fagaras, Rimnice Vilcea, Timisoara, Luogj
- **data structure:**
  $[Oradea, Sibiu, Arad, Timisoara, Lugoj, Mehadia]$
  $[Oradea, Sibiu, Arad, Zerind]$
  $[Oradea, Sibiu, Fagaras, Bucharest, Giurgiu]$
  $[Oradea, Sibiu, Fagaras, Bucharest, Pitesti, Craiova]$
  $[Oradea, Sibiu, Fagaras, Bucharest, Pitesti, RimniceVilcea]$
  $[Oradea, Sibiu, Fagaras, Bucharest, Urziceni, Hirsova]$
  $[Oradea, Sibiu, Fagaras, Bucharest, Urziceni, Vaslui]$
  $[Oradea, Sibiu, RimniceVilcea, Craiova, Dobreta, Mehadia]$
  $[Oradea, Sibiu, RimniceVilcea, Craiova, Pitesti, Bucharest]$
  $[Oradea, Sibiu, RimniceVilcea, Pitesti, Bucharest, Giurgiu]$
  $[Oradea, Sibiu, RimniceVilcea, Pitesti, Bucharest, Urziceni]$
  $[Oradea, Sibiu, RimniceVilcea, Pitesti, Craiova, Dobreta]$
  $[Oradea, Zerind, Arad, Sibiu, Fagaras, Bucharest]$
  $[Oradea, Zerind, Arad, Sibiu, RimniceVilcea, Craiova]$
  $[Oradea, Zerind, Arad, Sibiu, RimniceVilcea, Pitesti]$
  $[Oradea, Zerind, Arad, Timisoara, Lugoj, Mehadia]$

The total number of nodes generated by DFID is

$$N(DFID) = (d)b + (d-1)b^2 + ... + (1)b^d$$

, which gives a time complexity of $O(b^d)$. The total number of nodes generated by BFS is:

$$N(BFS) = b + b^2 + ... + b^d + (b^{d+1} - b)$$

. Thus, the time complexity of DFID is better than BFS. In general, iterative deepening is the preferred uninformed search method when there is a large search space and the depth of the solution is not known.