CSCI 5430 Artificial Intelligence Assignment 4

Due 11:59pm, Nov 6th, Sunday, 2016

1. Explain how the graph search for sensorless problems can be revised to make use of the fact that any action sequence which is a solution for a belief state b is also a solution for a subset of b. (10 points)
   1. **Graph search can be used for the sensorless problems because just like in the graph search where it looks to see if a vertex is connected to another vertex and if it is connected it removes the vertex from the set which means that all the vertices that come before the current vertex(b) in the independent set are a subset of b which means they are also a solution. Just like that any actions or steps taken before belief state b(which is a subset of b) are also a solution.**
2. In the slippery vacuum world, suppose the current belief state of the agent is {1,3} (see slides or textbook for definitions of the physical states 1~8). After the agent performs a Right action, an observation of [B, dirty] is sensed. What is the agent’s updated belief state? (10 points)
   1. **The updated belief state is 2 after performing a right action**
3. Implement a local search algorithm to solve the maximum independent set problem. See <https://en.wikipedia.org/wiki/Independent_set_(graph_theory)> for problem description. Choose from Hill-climbing search, Simulated annealing search, Local beam search, Genetic algorithms or one of their variants described in slides or textbook or other reference you found. You may choose to use Python, Java, or C/C++ to implement your algorithm. Input graph format is as follows: The first line specifies the number of vertices and edges; the following lines specifies all the edges, one line for each edge. Each edge is specified by its two end point labels. For a graph with n vertices and m edges, the valid vertex labels are from 0 to n-1 and the graph file will consist of m+1 lines. See example graph file gsample.txt. Output (to stdout) the independent set your algorithm found in the first line and its size in the second line. See the following sample output. Try to make your codes compilable and runnable on each of the major platforms Windows, Linux, Mac. (I will test your codes on two graph instances on Linux.) Your documentation need to specify how to compile and run your codes; specify what algorithm you have chosen to implement; what problem formulation and data structures you have chosen and how they fit into your implementation. About one page of explanation is expected unless you feel lesser amount is adequate. You will get 0 points if your code does not implement one of the requested algorithms. At least 40 points will be deducted if your code 1) does not compile, or 2) does not finish within 5 seconds, or 3) does not give correct results. (40 points (code)+ 40 points (documentation)=80 points)

Sample output (not for gsample.txt):

1 2 6 9 15 20

Size: 6

Answer to Question 3 is below

For the third question, I decided to solved the maximum independent set problem with the hill-climbing search algorithm. First, I created a class that would construct a vertex object and made it so the other vertices could be added to a vertex to add an edge, which is stored in ArrayList<Vertex> edges for each vertex object. The next method I created was getTopNumbers(), this method is used to retrieved the top two numbers from the inputted text file, I used ran25.txt and ran50.txt, and saved these two numbers as v\_vertices and v\_edges respectively so the program nows how many vertices and edges there are in the graph. I then created a method that would convert the rest of the information from the text file into an int[][] array to be useable with the Vertex object class, this method is called readFileInformation(), which returns the int[][] array once this method has completed running. The next method, createGraph(), which as the name implies is the method that creates the graph itself. It goes through a for-loop creating vertices 0-n, n being v\_vertices, and adds the vertices to the ArrayList<Vertex> vertices. After it creates the ArrayList of all the vertices it uses another method called, addEdges(), which takes two vertices using their id and connects them to each other adding them to their ArrayList<Vertex> edges. The createGraph() calls this method inside a for-loop from 0-n, n being v\_edges. Now the program calls on the HillClimb() method, this method does the following in this order:

1. Defines an empty set to hold the independent set
2. Chooses a random vertex to start at
3. Add it to the independent set
4. Then it goes into a for loop
   1. It gets another random vertex
   2. IF Statement to catch if the random vertex selected has already been added then gets another random vertex and if it has it starts the loop over by “--“ the value the for loop is adding to and using the keyword “continue”
   3. Once it is past the IF statement it adds the random vertex to the independent set
   4. Another IF statement uses method isIndependent(), to check to see if the newly added vertex is connected to any of the other vertices in the independent set and if it is the newly added vertex is removed from the independent set
5. Once the method has gone through all of the vertices it returns the independent set.

The last thing the program does is prints out the independent set and the size of the set. Below is a sample output:

Maximum Independent Set using Hill-Climbing Search

--------------------------------------------------

[10, 20, 13, 12, 2, 8, 21]

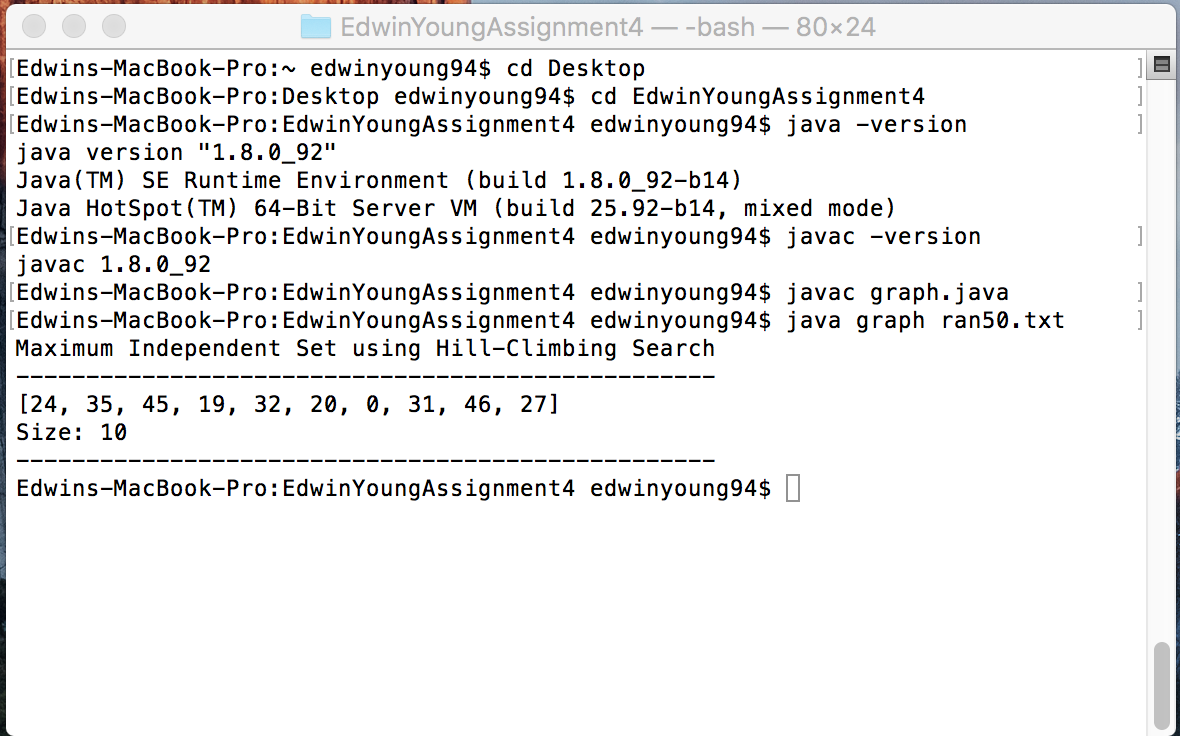
Size: 7

--------------------------------------------------

**The next 4 pages are documentation on how to compile and run the code on the following operating systems: Mac OSX, Windows, Ubuntu, CentOS**

**Mac OSX**

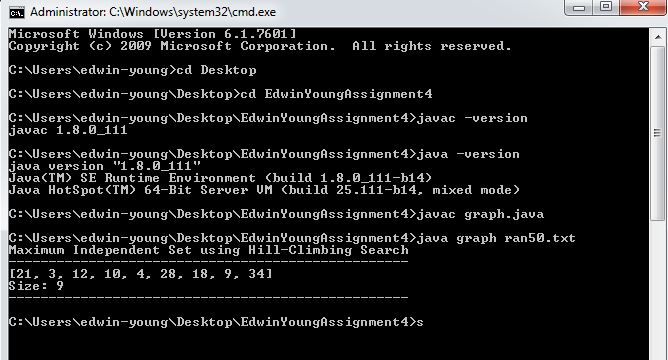
1. Download the .zip folder from Folio to Desktop
2. Extract to a folder to Desktop
3. Open Terminal and navigate to output folder on desktop
   1. For me it was
      1. cd Desktop
      2. cd EdwinYoungAssignment4
4. Make sure you have java and java complier installed by running the following commands(both should be installed before proceeding:
   1. java –version
   2. javac –version
5. Compile graph.java by typing following command into the terminal window
   1. javac graph.java
6. Run new java class that was just created from previous command by running the following command in the terminal window and add the .txt file you want to use for the vertices and edges(I used ran50.txt which is included in the .zip)
   1. java graph ran50.txt
7. Here is a screenshot of the terminal when I follow these steps:



**Windows**

**Completed this using Windows 7 Service Pack 1**

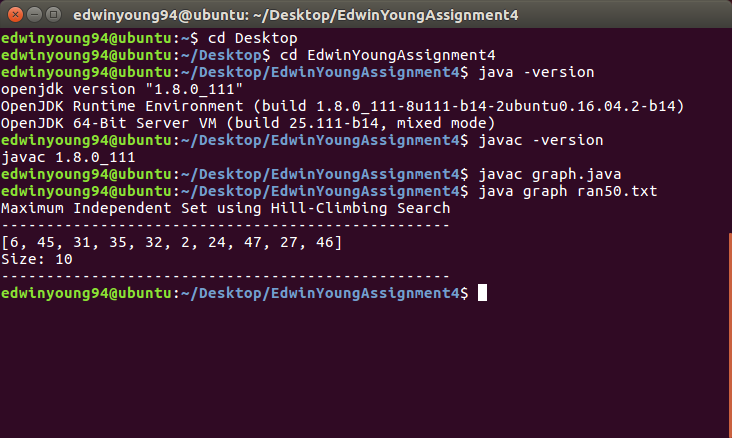
1. Download the .zip folder from Folio to Desktop
2. Extract to a folder to Desktop in a folder called “EdwinYoungAssignment4” or what ever is easiest for you to remember.
3. Open command prompt but hitting the windows key and typing “cmd” in start menu search bar then it “Enter” key and navigate to output folder on desktop
   1. For me it was
      1. cd Desktop
      2. cd EdwinYoungAssignment4
4. Make sure you have java and java complier installed by running the following commands(both should be installed before proceeding:
   1. java –version
   2. javac –version
5. Compile graph.java by typing following command into the command prompt window
   1. javac graph.java
6. Run new java class that was just created from previous command by running the following command in the command prompt window and add the .txt file you want to use for the vertices and edges(I used ran50.txt which is included in the .zip)
   1. java graph ran50.txt
7. Here is a screenshot of the command prompt when I follow these steps:



**Unbuntu**

**Completed using Ubuntu Desktop 16.04**

1. Download the .zip folder from Folio to Desktop
2. Extract to a folder on Desktop by clicking on it and naming folder for output “EdwinYoungAssignment4” or something that is easy to remember.
3. Open Terminal and navigate to output folder on desktop
   1. For me it was
      1. cd Desktop
      2. cd EdwinYoungAssignment4
4. Make sure you have java and java complier installed by running the following commands(both should be installed before proceeding:
   1. java –version
   2. javac –version
5. Compile graph.java by typing following command into the terminal window
   1. javac graph.java
6. Run new java class that was just created from previous command by running the following command in the terminal window and add the .txt file you want to use for the vertices and edges(I used ran50.txt which is included in the .zip)
   1. java graph ran50.txt
7. Here is a screenshot of the terminal when I follow these steps



**CentOS**

**Using version CentOS 7 DVD ISO**

1. Download the .zip folder from Folio to Desktop
2. Extract to a folder on Desktop by clicking on it and naming folder for output “EdwinYoungAssignment4” or something that is easy to remember.
3. Double click the output folder created on the desktop and right click in the window not on a file and select the option “Open in terminal
4. Make sure you have java and java complier installed by running the following commands(both should be installed before proceeding:
   1. java –version
   2. javac -version
5. Compile graph.java by typing following command into the terminal window
   1. Javac graph.java
6. Run new java class that was just created from previous command by running the following command in the terminal window and add the .txt file you want to use for the vertices and edges(I used ran50.txt which is included in the .zip)
   1. java graph ran50.txt
7. Here is a screenshot of the terminal when I follow these steps:

