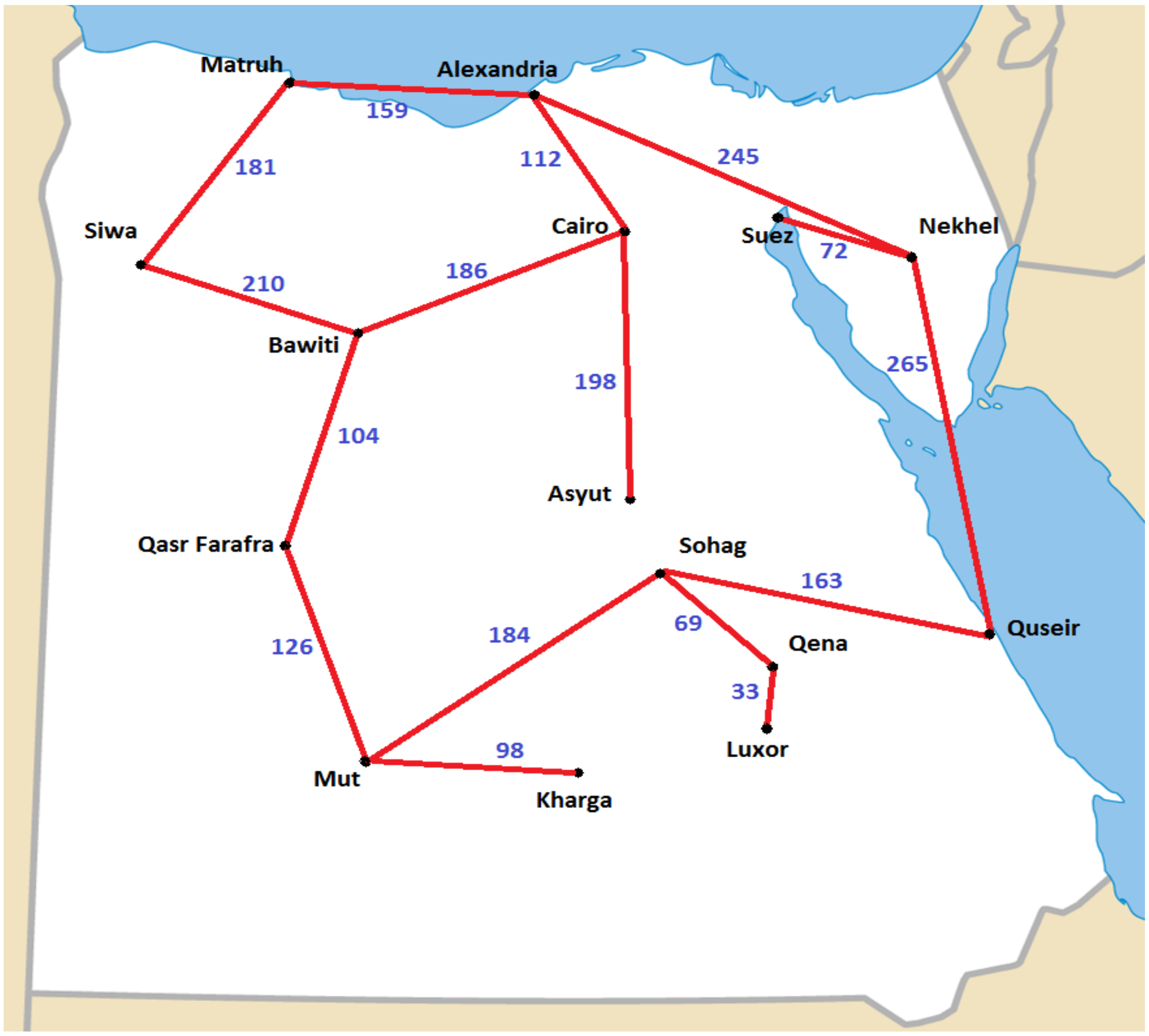
CSCI460 – Introduction to Artificial Intelligence
Instructor: Dr. K. Narayanaswamy

Assignment 2 – Greedy and A* Search Algorithms
Due: 02/28/2014 11:59:59pm

This assignment is a straightforward extension of the previous one, but now applying the Greedy and A* algorithms to perform the search. The map of Egypt is depicted again below with some new information.

1) The red line indicates an existing path or road between cities.
2) The indigo number is the actual path cost traveling by road between those two cities.



In addition to the map above, a table with information about the Straight Line Distances(SLD) for two admissible heuristics **h1** and **h2**, from each city to Luxor is also provided below.

Estimated SLD Distances to Luxor

From City	h1	h2
Matruh	174	189
Cairo	126	139
Nekhel	133	145
Siwa	132	148
Bawiti	105	118
Asyut	52	67
Suez	121	136
Qasr Farafra	68	77
Quseir	55	59
Mut	51	65
Kharga	24	38
Sohag	27	36
Qena	10	19

For example, using the information from both the map and the table above, we obtain the following for Alexandria and Cairo:

- The actual distance if you travel along roads from Alexandria to Cairo is 112 miles.
- The **h1** SLD heuristic returns a value of 126 miles for Cairo, i.e., **h1** believes that Cairo is exactly 126 miles from Luxor.
- The **h2** SLD heuristic returns a value of 139 miles for Cairo, i.e., **h2** believes that Cairo is exactly 139 miles from Luxor.

Programming: Please write a program that will use the Greedy search and A* search algorithms to traverse the tree above starting at Alexandria and ending at Luxor.

Important Notes:

1. Similar to the previous assignment, the focus is on the search algorithms, so you should construct a tree and use it as the search-space for the Greedy and A* algorithms. As before, the algorithms differ only in their queuing functions, so you are to write a **generic search algorithm as well as two separate queuing functions**, one for Greedy and one for A*.

2. Your Greedy and A* algorithms will both need to run twice, once using only the values for h1, and again using the h2 values. Your generic search algorithm should make a call to a queuing function and use it as the basis to conduct its search. The pseudo code below illustrates this setup:

```
search( )
{
        // generic search algorithm as below
    loop
        If nodes is empty then return failure
        Node ← Remove-Next (nodes)
        If Goal-Test(Node) then return Node as Result
        Nodes ← greedyQueuing(Nodes, Expand(Node), h1) // 1st test for h1
        // Nodes ← greedyQueuing(Nodes, Expand(Node), h2) // 2nd test for h2
        // Nodes ← aStarQueuing(Nodes, Expand(Node), h1) // 3rd test for h1
        // Nodes ← aStarQueuing(Nodes, Expand(Node), h2) // 4th test for h2

      end loop

    …..
}
```

3. We will be analyzing and reviewing your code and also testing the program by substituting in each of the different queuing functions. Your output should include the number of nodes or cities examined, as well as the names of the cities as they are traversed, beginning at Alexandria and ending at Luxor. A sample output is given below:

Output: Nodes examined = ?
Solution: Alexandria, Cairo, …, …., etc, Luxor

4. The grader will test each search algorithm by changing ONLY the line that calls the queuing function above – 1st test, 2nd test, 3rd test, 4th test. No other changes will be performed or accommodated. So, it is very important that you test your program by changing only the single line that calls different queuing functions as above.

Caveats: Please obey the following rules

1. Your program must be written in either Java or C++.
2. Your program **MUST** compile and run on aludra.usc.edu.
3. Do not expect any command line argument(s).
4. Your program will be deemed incorrect unless it follows the basic outline for the generic search algorithm as above, and works just by including the appropriate queuing function for each kind of search algorithm.
5. Write your own code. Files will be compared.

Question:

1. Comment on your observations regarding the performance of A* using both **h1** and **h2** as the estimated path cost to the goal. Which one (**h1** or **h2**) performs a more efficient search based on your output data (i.e., the examined nodes)? (5 points)
2. Provide explanations of how your data either matches the expected or predicted analytical result or why it does not match the expected or predicted result. (5 points)

Grading Policy: (Total Points: 100)

Programming (80 pts):
Your program must provide four solutions.

1. Correct traversal and nodes examined for Greedy Search using h1: 20 points
2. Correct traversal and nodes examined for Greedy Search using h2: 20 points
3. Correct traversal and nodes examined for A* Search using h1: 20 points
4. Correct traversal and nodes examined for A* Search using h2: 20 points

Note: <u>If your program runs correctly but you fail to implement the generic search algorithm, you will lose 25 points.</u>

Readme.txt (20 pts):

1. Your answer to the Question posed above. You must address each part in sufficient detail to get all the allocated credit (10 pts).
2. A brief description of the program structure and any other issues you think will be helpful for the grader to understand your program. (5 pts)
3. Instructions on how to compile and execute your code. (5 pts)
4. **Please include your name, student ID and email address on the top.**
5. You must submit a program in order to get any credit for the Readme.txt. In short, if you submit ONLY a Readme.txt file you will get 0.

Submission: Your program files will all be submitted via blackboard. You MUST follow the guidelines below. Failure to do so will incur a -25 point penalty.

1) Compress and zip ALL your homework files(this includes the Readme.txt and all source files you wish to include) into **one** .zip file. Note that only .zip file extensions are allowed. Other compression extensions such as .tar, rar, or 7z will **NOT** be accepted.

2) Name your zip file as follows: firstname_lastname.zip. For example, John_Smith.zip would be a correct file name.

3) To submit your assignment, simply select the appropriate assignment link from the Assignments subsection of the course blackboard website. Upload your zip file and click submit.

Please make sure ALL source files are included in your zip file when submitted. Errors in submission will be assessed –25 points. A program that does not compile as submitted will be given 0 points. Only your FINAL submission will be graded.

For policies on late submissions, please see the Syllabus from the course home page. These policies will be enforced with no exceptions.