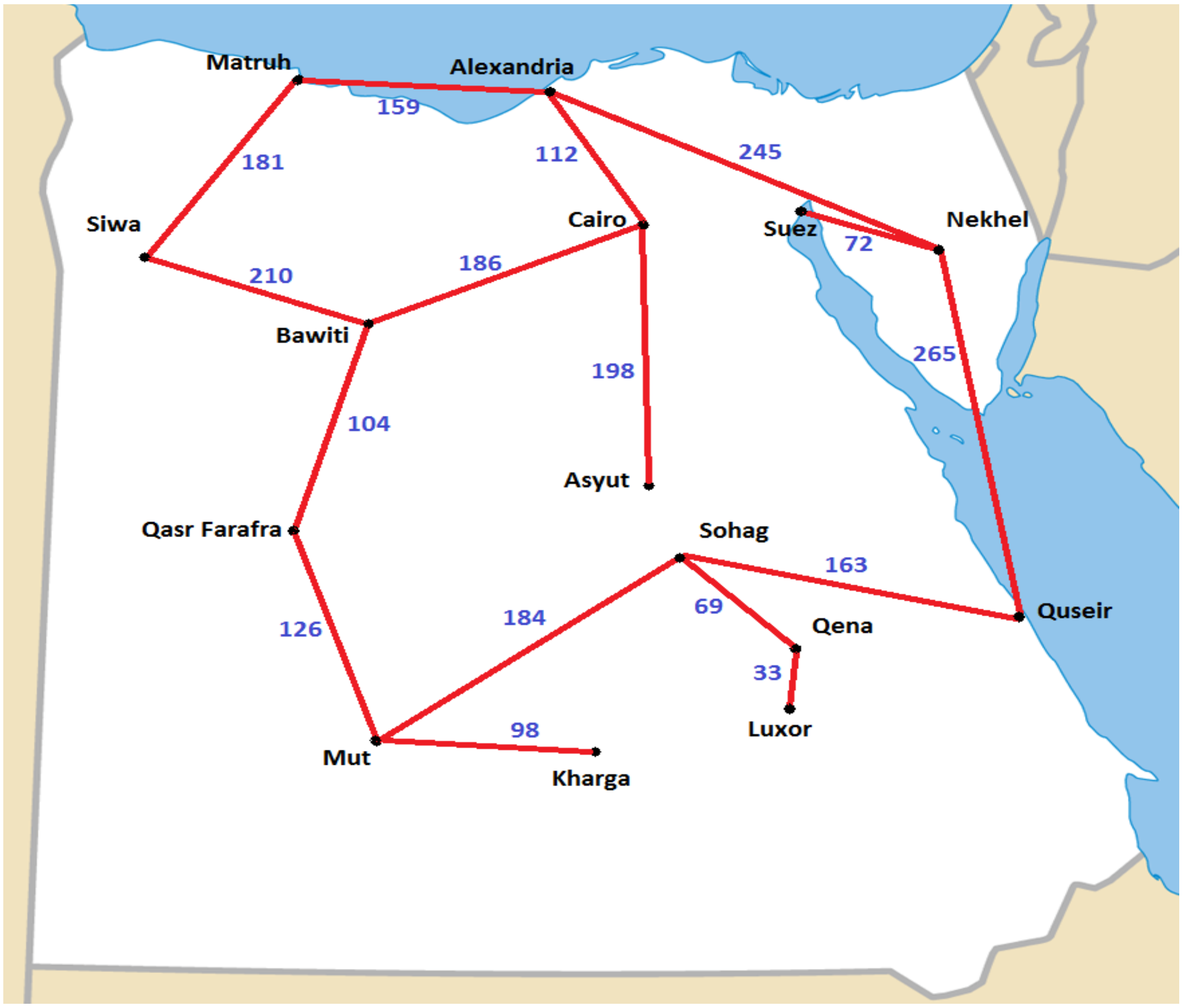
CSCI460 – Introduction to Artificial Intelligence
Instructor: Dr. K. Narayanaswamy

Assignment 1 – Search Algorithms
Due: 02/14/2014 11:59:59pm

The image below is a map of Egypt with a few major cities depicted. The map provides information on the straight line distances between cities. The red line indicates a path between cities, while the indigo number beside it is the distance in miles between them. This assignment involves the implementation of several search algorithms that will traverse the cities to understand how effective each is at searching a search-space.

Programming: Please write a program that will use Breadth First Search, Depth First Search and Uniform Cost Search to traverse the tree below starting at Alexandria and ending at Luxor.



Important Notes:

1. The focus of this assignment is on the implementation of algorithms. Therefore, your program should build a tree using the information from the map above. This map will serve as the search-space for the three different algorithms: BFS, DFS, and Uniform Cost Search.

2. Each of the 3 search algorithms differs only in its queuing function. You are to write a **generic search algorithm as well as three separate queuing functions**, one for BFS, DFS, and Uniform Cost. 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 🡐 breadthFirstQueuing(Nodes, Expand(Node)) // 1st test
        // Nodes 🡐 depthFirstQueuing(Nodes, Expand(Node)) // 2nd  test
        // Nodes 🡐 uniformCostQueuing(Nodes, Expand(Node)) // 3rd test
     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 simply be the names of the cities as they are traversed, beginning at Alexandria and ending at Luxor. A sample output is given below:

Output: Alexandria, Cairo, …, …., etc, Luxor

In case cities are equivalent, your algorithm should select the city that is lowest in alphabetic order: A comes before B, and B before C, and so forth. As an example, a partial solution for BFS is given below:

Output: Alexandria, Cairo, Matruh, Nekhel, Asyut, Bawiti, Siwa, Quseir, Suez,…, 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. 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.

Grading Policy: (Total Points: 100)

Programming (90 pts):
Your program must implement the generic search algorithm and the three search algorithms.

1. Correct traversal for Breadth First Search: 30 points
2. Correct traversal for Depth First Search: 30 points
3. Correct traversal for Uniform Cost Search: 30 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 (10 pts):

1. 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)
2. Instructions on how to compile and execute your code. (5 pts)
3. **Please include your name, student ID and email address on the top.**
4. 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.