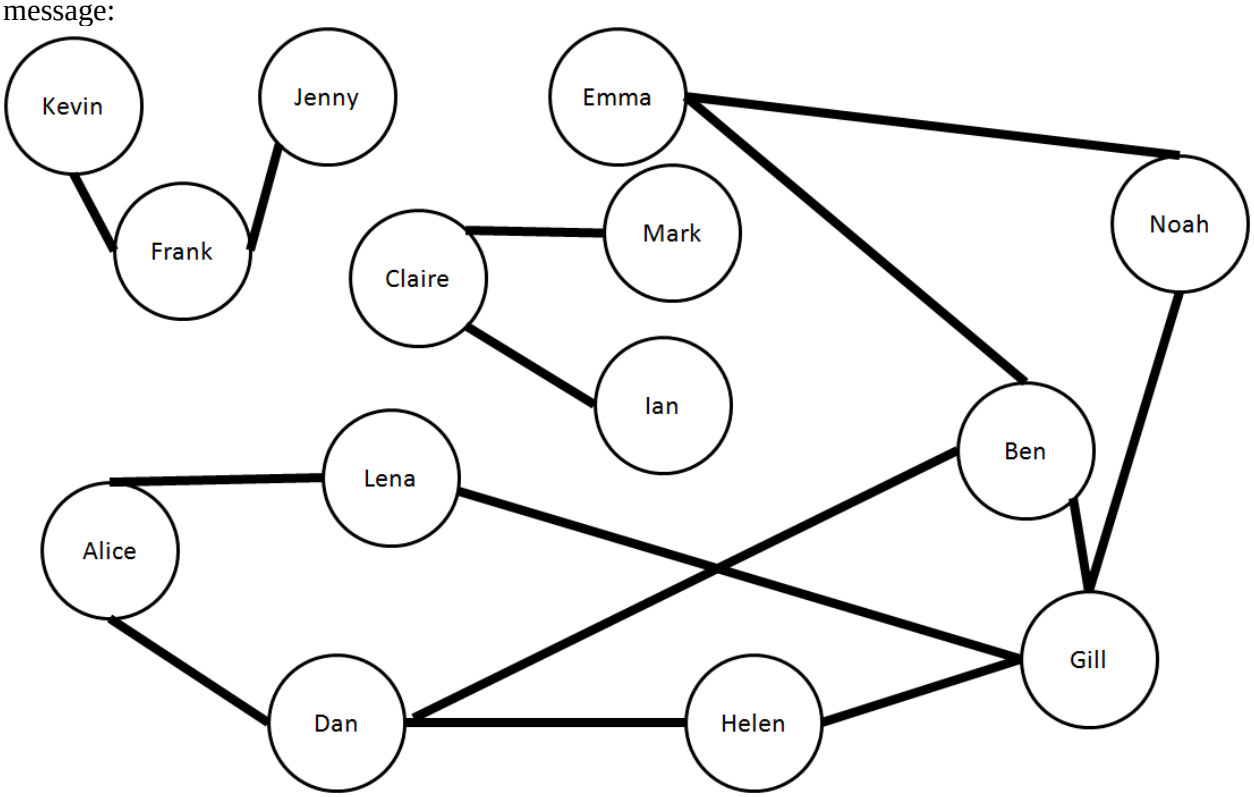
CSCI 561 – Foundations of Artificial Intelligence
Instructor: Dr. K. Narayanaswamy

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

1. Introduction

In this assignment, we will re-examine the small-world experiment in social network using uninformed (blind) search algorithms. For those who are not familiar with the experiment, please refer to [1][2]. To give you a realistic and modern context, we twist the experiment and put it into a limited scope of a simplified social network. Suppose ***Alice*** is a graduate student in computer science at USC, and she wants to start her own IT company by applying a great idea. She has to look for investment from industry. ***Noah*** catches her eye as the head of a notable venture capital company. ***Alice*** plans to send ***Noah*** a message and give him an outline of the startup plan. To make herself more influential, ***Alice*** thinks it is better to send the message via Facebook to someone ***Noah*** trusts and who is also her friend in an effort to convey the message. Thanks to Facebook, ***Alice*** is able to see the friends ***Noah*** has. Unfortunately, ***Alice*** is not able to find any mutual friends she has with ***Noah***. However, with the help of one of her friends, who is an expert in social network mining, Alice manages to find the relevant people who might potentially be able to forward her message. The following figure depicts the social network, where links between two nodes represent "friend" relationship between the two people. There are three factors Alice should consider while sending her message:

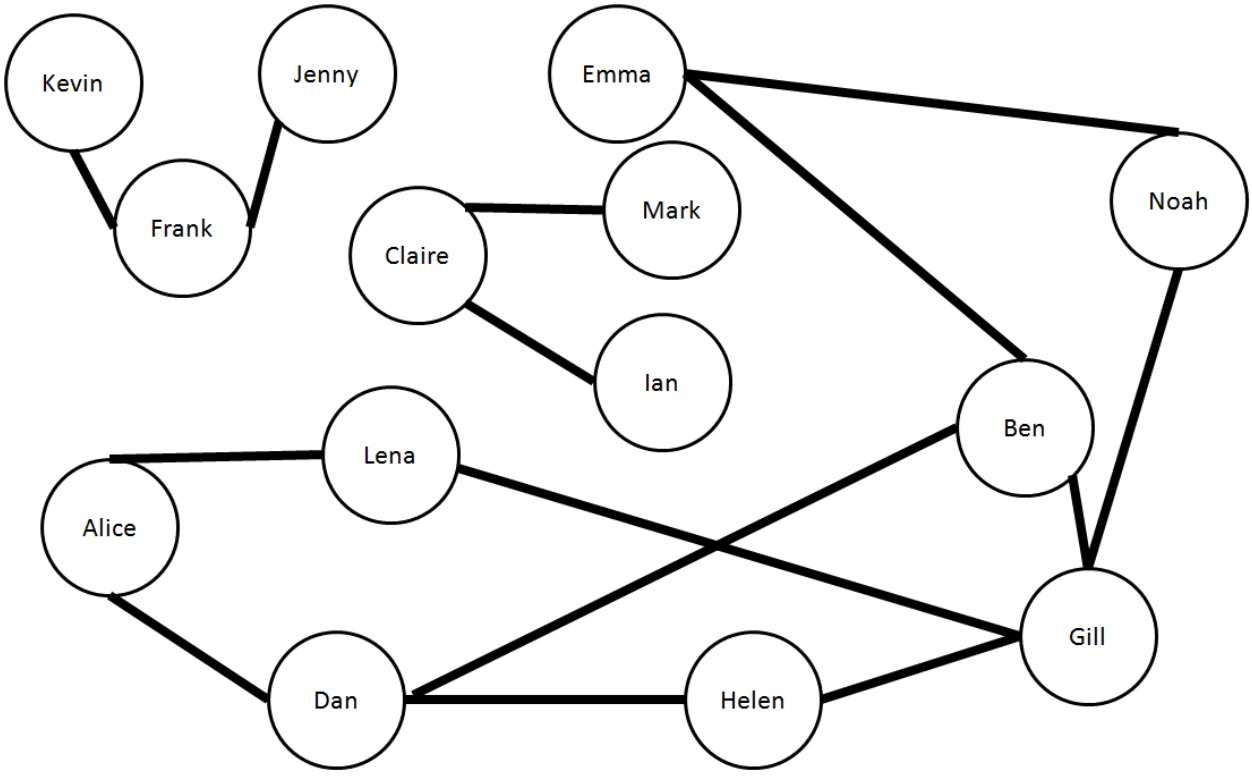


1. To transmit the message as efficiently as possible, ***Alice*** will use a postscript in her message to tell anyone who receives the message that the recipient should only forward it to her/his friend(s). This means the message is forwarded exclusively along the "friendship" links in the social network. Otherwise, Alice could be at risk of failure because someone might not forward the message if she/he does not know the sender of the message.
2. The delivery time is one of ***Alice***'s concerns. Her expert friend helped estimate the reaction time it takes for one to receive a message from another person and forward the message. It was done by mining the social interactions in Facebook. To deliver the message as soon as possible, ***Alice*** should choose a traversal path that is optimal.

2. Communities Detection

This section is for the task 4. In the real community detection problem, a community refers to groups of nodes with ***dense connections*** internally and ***sparser connections*** between groups.

In this homework, we simplify the community detection problem to the connected component problem (Thus, "*six degrees of separation*" theory in the small world experiment does not hold in this artificial dataset.). Our goal is to find sub graphs of the given social network graph. More details of connected component problem can be found from [3]. You will need to understand the algorithms described in [3] and select either breadth-first search or depth-first search to implement in your program for this task.



In this example, there are 3 communities. Alice, Ben, Dan, Emma, Gill, Helen, Lena and Noah are in the first community. Claire, Ian, and Mark are in the second community. Frank, Jenny and Kevin are in the third community.

3. Tasks

In this assignment, you need to write a program to implement the following search algorithms Breadth-first search, Depth-first search and Uniform-cost search. There are a total of four tasks in this assignment.

1) Find a path between two specific nodes with Breadth-first search in Graph-Search version

2) Find a path between two specific nodes with Depth-first search in Graph-Search version

3) Find the optimal path between two specific nodes with Uniform-cost search in Graph-Search version. The optimal path is the path that has minimal delivery time end to end.

4) Find communities (connected components) using breadth-first search or depth-first search. Specify in the readme.txt which algorithm that you use.

Tie breaking: For all algorithms and all tasks, if there are multiple equivalent choices about which node goes next, your program should pick the node that comes first in **alphabetic** order. A list of node strings in alphabetic order is given in a separate file so that you do not need to sort the names in the program. Please refer to the input section for further details.

4. Input

There are four inputs for your programs

4.1 **The task to be performed:** there are four possible values

41.1 **1**: Task number 1

4.1.2 **2**: Task number 2

4.1.3 **3**: Task number 3

4.1.3 **4**: Task number 4

4.2 **The start node** (Only for tasks 1-3)

4.3 **The goal node** (Only for tasks 1-3)

4.4 **The social network graph**: is provided in the form of an adjacency list. The graph is assumed to beweighted and undirected. Each line represents an edge of a graph. There are three columns per line

separated by comma. The first two columns represent the nodes of an edge. The third column represents the cost of an edge connecting the nodes – representing an estimate of the time required for the recipient to respond to the message.

The number of nodes or their names can be different from the given examples.

You may assume that the input files you will be provided are valid – no need to do any error processing on the input files.

Example:

Lena,Claire,17

This represents that it takes 17 hours for ***Claire*** to react and forward the message if ***Lena*** send the message to ***Claire***. Please note that you may assume the friendship is mutual, so it also takes equal amount of time to send the message from ***Claire*** to ***Lena***. It also applies to the risk value.

4.5 The tie breaking file:

This file is used for the tie breaking. The input nodes are given in sorted order. The order in which the nodes appear should be used for prioritization in selection of the nodes that are equivalent in expansion. This will be alphabetic usually, but your code should simply use the order in which the nodes appear in this file as the priority order of the nodes when their scores are otherwise equivalent.

Example:

Alice
Dan
Kevin
Lena

In the example above, if there is a tie between Dan and Lena, you should expand Dan before Lena.

5. Output

In this assignment, you need to output how your search algorithms traverses the graph and the path as well , in two output files.

5.1 A path between two nodes (task1-3)

List the names of the nodes in the path between two nodes (separated by new lines) in the order.

Example:

Alice

Dan

Ben

Emma

Noah

Please see the complete examples in output1_path_t1.txt, output1_path_t2.txt, output1_path_t3.txt, output2_path_t1.txt, output2_path_t2.txt, output2_path_t3.txt.

5.2 Communities (task4)

List the names of the nodes of each community (separated by new lines). Each line contains nodes of that community in alphabetic order. For example,

Alice, Ben, Dan, Emma, Gill, Helen, Lena, Noah

Claire, Ian, Mark

Frank, Jenny, Kevin

5.3 A traverse log (task 1-4)

List the names of the nodes (separated by new lines) in the order traversed by your program. Please see the complete examples in output1_tlog_t1.txt, output1_tlog _t2.txt, output1_ tlog_t3.txt, output1_tlog_t4.txt, output2_tlog_t1.txt, output2_tlog_t2.txt, output2_tlog_t3.txt, output2_tlog_t4.txt

6. Program Specifications

6.1 Your program must be written in either Java or C++.
6.2 Your program MUST compile and run on aludra.usc.edu
6.3 Write your own code. Files will be compared and any cheating will be detected and reported to University administration authorities.
6.4 Your program name must be "search" (without quotation mark).

7. Execution Details

Your program will be tested on aludra.usc.edu on unseen input files that meet the input file specifications. Your program will receive 7 arguments for task 1-3 and 5 arguments for task 4 (No start and goal node for task 4).

7.1 C/C++

The grader will execute this command:

```
search -t <task> -s <start_node> -g <goal_node> -i <input_file> -t <tie_breaking_file> -op <output_path> -ol <output_log>
```

Example:

```
search -t 1 -s Alice -g Noah -i input1.txt -t tiebreak.txt -op output1_path_bfs.txt -
ol output1_tlog_bfs.txt

search -t 4 -i input1.txt -t tiebreak.txt -op output1_path_cc.txt -ol
output1_tlog_cc.txt
```

7.2 JAVA

The grader will execute this command:

```
java search -t <task> -s <start_node> -g <goal_node> -i <input_file> -t
<tie_breaking_file> -op <output_path> -ol <output_log>
```

Example:

```
java search -t 1 -s Alice -g Noah -i input1.txt -t tiebreak.txt -op
output1_path_bfs.txt -ol output1_tlog_bfs.txt

java search -t 4 -i input1.txt -t tiebreak.txt -op output1_path_cc.txt -ol
output1_tlog_cc.txt
```

7.3 Arguments:

7.3.1 <task> : there are 4 possible values **1**, **2, 3** and **4**.

7.3.2 <start_node>: the start node

7.3.3 <goal_node> : the goal node

7.3.4 <input_file>: location of an input file.

7.3.5 <tie_breaking_file>: location of a tie breaking file.

7.3.6 <output_path>: location of an path output.

7.3.7 <output_log>: location of an traverse log output.

Thus, you should interpret the example:

```
search -t 1 -s Alice -g Noah -i input1.txt -t tiebreak.txt -op output1_path_bfs.txt -
ol output1_tlog_bfs.txt
```

The first task is chosen. The start node is Alice and the goal node is Noah. The location of the input file is same as your program and its name is input1.txt. The location of the tie breaking file is same as your program and its name is tiebreak.txt. The location of path output file is same as your program and its name is output1_path_bfs.txt. Finally, the location of traverse log output file is same as your program and its name is output1_tlog_bfs.txt.

```
search -t 4 -i input1.txt -t tiebreak.txt -op output1_path_cc.txt -ol
output1_tlog_cc.txt
```

The forth task is chosen. There is no start and goal node. The location of the input file is same as your program and its name is input1.txt. The location of the tie breaking file is same as your program and its name is tiebreak.txt. The location of path output file is same as your program and its name is output1_path_cc.txt. Finally, the location of traverse log output file is same as your program and its name is output1_tlog_cc.txt.

8. Grading Policy: (Total Points: 100)

8.1 Programming (90 pts):

8.1.1 Correct outputs for task 1: 20 points
8.1.2 Correct outputs for task 2: 20 points
8.1.3 Correct outputs for task 3: 20 points
81.4 Correct outputs for task4: 20 points
8.1.5 Your explanation must be part of readme.txt.: 10 points
8.1.5.1 Your choice of the search algorithm for task4.
8.1.5.2 Your analysis of similarities/differences between these three algorithms on the path-finding problems.
8.1.5.3 What will happen if we use Uniform cost search in the connected component problem? Will the result, time/space complexity be different?

8.2 Readme.txt (10 pts):

8.2.1 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)
8.2.2 Instructions on how to compile and execute your code. (5 pts)
8.2.3 Please include your name, student ID and email address on the top.
8.2.4 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.
8.2.5 Remember to also include the explanation of outputs (Part 8.1.5)

9. Submission Guidelines

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.

9.1 Compress and zip ALL your homework files (this includes the Readme.txt and all source files) 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.
9.2 Name your zip file as follows: firstname_lastname.zip. For example, John_Smith.zip would be a correct file name.
9.3 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 (clicking send is not enough).

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

References

[1]. http://en.wikipedia.org/wiki/Small-world_experiment
[2]. http://www.cs.cornell.edu/home/kleinber/swn.d/swn.html
[3]. http://en.wikipedia.org/wiki/Connected_component_%28graph_theory%29