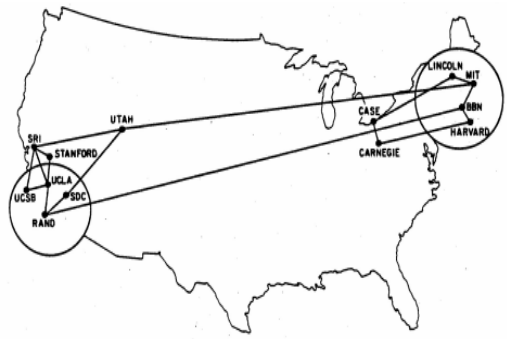
CSCI 561 – Foundations of Artificial Intelligence
Instructor: Dr. K. Narayanaswamy

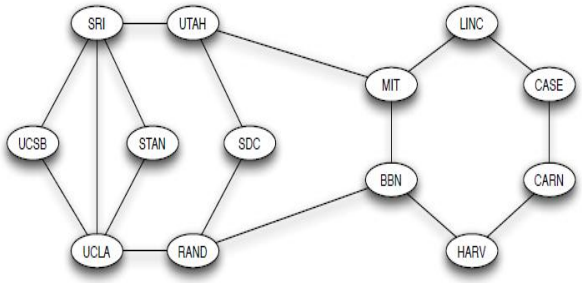
Assignment 1 – Search Algorithms
Due: 09/27/2013 11:59:59pm

1. Introduction

In this assignment, you will implement a path finding algorithm for the Internet. Specifically, you will write a program that will apply Breadth-first search, Depth-first search and Uniform-cost search to find a path between two specific nodes.

The left image below depict the sites on Internet known as the ARPANET. The right image is an alternate drawing of the thirteen node internet graph from December 1970.





2. Tasks

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

For **tasks 1-3**, we assume that the **network is completely reliable**.

1) Find the optimal path between two specific nodes with Breadth-first search in Tree-Search version

2) Find the optimal path between two specific nodes with Depth-first search in Tree-Search version

3) Find the optimal path between two specific nodes with Uniform-cost search in Graph-Search version

For task 4, we assume that the network is unreliable

4) Find the optimal path between two specific nodes with Uniform-cost search in Graph-Search version using **expected time** for traversal as the cost function.

In this task, we assume that some edges in the graph are unreliable. The cost for sending a message is uncertain. For simplicity, probability distribution of a cost of an unreliable edge is assumed to be a uniform distribution [k,k+1] or k+U where U is a uniform distribution from 0 to 1. This kind of distribution is called Irwin-Hall Distribution. More details of a Irwin-Hall distribution can be found from [1][2].

Assume the cost of a path p is a random variable, c_p

Our goal is to find

$$p^* = \text{argmin}_p E(c_p)$$

where p* is an optimal path.

E(c_p) is the expected value of a random variable c_p. Under the Irwin Hall distribution, c_p can be computed by the following equation.

$$c_p = k_1 + k_2 + k_3 + ... + k_n + \sum_{i=1}^{m} Ui$$

Where n is a number of reliable edges and m is number of unreliable edges.

k_i is the cost of edge i

The following are illustrative examples on how to compute the expected time for Irwin Hall Distribution on unreliable paths using sample data:

Node A	Node B	Cost(time)	reliable
UCSB	SRI	1.2	1
SRI	UTAH	1.5	0
UTAH	SDC	0.7	0
SDC	RAND	0.6	0
RAND	BBN	1.2	0
UTAH	MIT	1.9	1
MIT	BBN	0.8	0

Ex1: Compute expected time from UCSB->BBN on the path

p_1 = UCSB->SRI->UTAH->SDC->RAND->BBN

$$C_{p1} = 1.2+1.5+0.7+0.6+1.2+ \sum_{i=1}^{4} Ui$$

$\sum_{i=1}^{m} Ui$ is a Irwin–Hall distribution or (uniform sum distribution)

$$\mathbf{E}(C_{p1}) = 1.2+1.5+0.7+0.6+1.2 + E(\sum_{i=1}^{4} Ui)$$

= 5.2 + 2.0

= 7.2

Ex2: Compute expected time from UCSB->BBN on the path

p_2= UCSB->SRI->UTAH->MIT->BBN

E(C_{p2}) = 1.2+1.5+1.9+0.8 + $E(\sum_{i=1}^{2} Ui)$

= 5.4+1.0

= 6.4

For Task 4, you will implement the cost function as above, and rerun the UCS algorithm. You will need to compare and understand similarities and differences (if any) between the optimal path derived in Task 3 and Task 4, and **comment on it in the readme.txt file.**

3. Input

There are four inputs for your programs

3.1 **The question number:** there are four possible values

3.1.1 **1**: Task number 1

3.1.2 **2**: Task number 2

3.1.3 **3**: Task number 3

3.1.3 **4**: Task number 4

3.2 **The start node**

3.3 **The goal node**

3.4 **The internet graph**: is provided in the form of an adjacency list. The graph is assumed to be a weighted undirected graph. Each line represents an edge of a graph. There are four columns per line of adjacency data. The first two columns represent the nodes of an edge. The third column represents the cost of an edge connecting the nodes. The fourth column represents the reliability of the edge. This column will be 1 (meaning the edge is reliable) or 0 (meaning the edge is unreliable).

Each column is separated from the next by a comma. The number of nodes or their names can be different for a given example.

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

Example:

UCLA ,UCSB,1.5,0

SRI,UCLA,0.7,1

This input file can be read as there are three nodes and two edges in this graph. The first edge is between UCLA and UCSB. The second edge is between SRI and UCLA. Two input sample files are provided in the assignment packgage: input1.txt and input2.txt. Please see them for more information.

4. Output

In this assignment, we are interested in both how your search algorithms traverse the graph and the path. There are two output files.

4.1 **A path between two nodes**

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

Example:

SRI

UCLA

UCSB

Please see the complete examples in output1_path_t1.txt, output1_path_t2.txt, output1_path_t3.txt, output1_path_t4.txt, output2_path_t1.txt, output2_path_t2.txt, output2_path_t3.txt, output2_path_t4.txt.

4.2 A traverse log

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

5. Program Specifications

5.1 Your program must be written in either Java or C++.
5.2 Your program MUST compile and run on aludra.usc.edu
5.3 Write your own code. Files will be compared and any cheating will be reported.
5.4 Your program name must be "search" (without quotation mark).

6. Execution Details

Your program will be tested on aludra.usc.edu on unseen input files that meet the input file specifications. Your program will recieve 6 arguments, 4 for inputs and 2 for outputs.

6.1 C/C++

The grader will execute this command:

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

Example:

search -t 1 -s UCSB -g BBN -i input1.txt -op output1_path_bfs.txt -ol output1_tlog_bfs.txt

6.2 JAVA

The grader will execute this command:

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

Example:

java search -t 1 -s UCSB -g BBN -i input1.txt -op output1_path_bfs.txt -ol output1_tlog_bfs.txt

6.3 Arguments:

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

5.3.2 <start_node>: the start node

5.3.3 <goal_node> : the goal node

5.3.4 <input_file>: location of an input file.

5.3.5 <output_path>: location of an path output.

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

Thus, you should interpret the example:

search -t 1 -s UCSB -g BBN -i input1.txt -op output1_path_bfs.txt -ol output1_tlog_bfs.txt

The first task is chosen. The start node is UCSB and the goal node is BBN. The location of the input file is same as your program and its name is input1.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.

7. Grading Policy: (Total Points: 100)

7.1 Programming (90 pts):
Your program must implement the three search algorithms.

7.1.1 Correct outputs for task 1: 20 points

7.1.2 Correct outputs for task 2: 20 points

7.1.3 Correct outputs for task 3: 20 points

7.1.4 Correct outputs for task4: 20 points

7.1.5 Your analysis of similarities/differences between optimal paths in UCS with reliable edges and UCS using unreliable edges. Your explanation must be part of readme.txt.: 10 points

7.2 Readme.txt (10 pts):

7.2.1 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)

7.2.2 Instructions on how to compile and execute your code. (5 pts)

7.2.3 Please include your name, student ID and email address on the top.

7.2.4 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.

7.2.5 Remember to also include the explanation of outputs (Part 6.1.5)

8. 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.

8.1 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.

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

8.3 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

9. References

1) http://en.wikipedia.org/wiki/Irwin%E2%80%93Hall_distribution

2) http://mathworld.wolfram.com/UniformSumDistribution.html