CSCI 561: Foundations of Artificial Intelligence
Instructor: Prof. Laurent Itti
Homework 1: Uninformed Search
Due on September 23rd at 11:59pm, 2015

UNINFORMED SEARCH

There is an old city that has an antique plumbing system. There are a lot of pipes in the city that are useless or rarely work. We want to find a route for water to traverse from one side of the city to the other side. There is a source of water in which water will start flowing through the system. And there are some destinations which whenever water reaches one of them; we consider the task of routing the water through the city to be done. So, formally speaking, there is one start node (source) and one or more end nodes (destinations) in the city.

One strange fact about this city is that some of the pipes which are old do not function in specific times. So, the route for water might be different in different times. You will be given the time periods when each pipe doesn't work. But if water is already flowing in a pipe, it will continue flowing even if time reaches one of those periods. You just have to make sure that when you want to select the next pipe for the rest of your water route, it is in its working time! Keep in mind that water cannot stay in one point waiting for a pipe to start working. If you reach a point where there is no pipe open from there, simply this route isn't a valid route and you have to disregard it.

You have to keep track of the time too. This is simple. We assume that it takes n units of time for water to flow through a pipe if the length of the pipe is n. For example if you have two pipes with length 3 and 4 in your route till now and you have started from time 2, current time will be 2 + 3 + 4 = 9.

Your task is to find a route based on the availability of the pipes in a given time using search algorithms. You are given the start time that water will start flowing and then you have to report the time which water reaches the other side of the city.

You will be using uninformed search algorithms that you have learned in class, i.e. you will be implementing DFS – BFS – UCS for this assignment.

Input:

You will be given a text input file. First line of this file represents the number of test cases. The next line will be the beginning of the 1st test case. Each test case ends with an empty line. Each test case consists of the following lines:

- <task> algorithm that you are supposed to use for this case
- <source> name of the source node
- <destinations> names of the destination nodes
- <middle nodes> names of the middle nodes
- <#pipes> represents the number of pipes
- <graph> represents start-end nodes, lengths and off-times of pipes
- <start-time> the time when water will start flowing

Task:

This field indicates which algorithm you are going to use to solve the problem. The input is either "BFS", "DFS" or "UCS" (without the double quotes)

There are multiple definitions for these algorithms. In this homework we ask you to use the definition in subsection 3.4 of "Artificial Intelligence, A modern Approach, 3rd edition" (starting at page 81). The reference implementation used to create the problem solutions will use these algorithms. Hence beware that using different algorithms may result in incorrect solutions.

Source:

This is the name of the source of water.

Destinations:

This is a space separated line consisting of names of the destination (goal) nodes.

Middle-nodes:

This is a space separated line consisting of the middle nodes, i.e. nodes that are neither source nor destination. (It may be an empty line which means there are no middle nodes.)

#Pipes:

This number represents the number of pipes in the system.

Graph:

This section contains #pipes number of lines. Each line in this section represents one pipe of the system. Format of each line is as following: (There is one space between each field)

<start> <end> <length> <#off-periods> < $period_1$ > …. < $period_n$ >

Example: S E 10 3 10-12 15-16 25-29

It means that this pipe starts from point S , ends in point E , has the length 10, and it has 3 off-periods. It is not working from time 10 to 12, 15 to 16 and 25 to 29. Period 10-12 means that if we are at time 10, 11 or 12 we cannot use this pipe as the next pipe. Some pipes may always work. In that case, the 4th field for these pipes will be 0. Please note that the pipes are unidirectional, i.e. for a pipe that has starting point A and ending point B, the water can flow from A to B only and not in the reverse direction.

The pipe length will be 1 for both BFS and DFS (i.e. pipe length is ignored by these algorithms and is always assumed to be 1). Also, ignore the off-periods for these algorithms, i.e. when using BFS-DFS; assume that all pipes work all the time.

Start-time:

This is an integer denoting the time when water will start flowing from the source point.

Input example:

```
2
BFS
A
B C D
E F G H I
5
A B 12 0
A E 3 3 2-4 1-5 9-10
E H 2 1 1-2
H D 5 2 5-6 2-3
I C 6 1 10-14
3

UCS
AA
BA
CA DA
3
AA BA 10 1 1-2
AA CA 2 2 3-4 5-6
CA BA 4 0
0

```

NOTES:

- You can use C++, JAVA or PYTHON to implement your code.
- You need to create a file named "waterFlow.xxx" where "xxx" is the extension for the programming language you choose. ("py" for python, "cpp" for C++ and "java" for Java.) The command to run your program would as follows: (When you submit the homework on Vocareum.com, the following commands will be executed.)

  C++:

  ```
  g++ waterFlow.cpp –o waterFlow.o
  ./waterFlow.o –i inputFile
  ```

  Python:

  ```
  python waterFlow.py –i inputFile
  ```

  Java:

  ```
  javac waterFlow.java
  java waterFlow –i inputFile
  ```

- Input file is a text file ending with .txt extension.
- You will use Vocareum.com to submit your code. Please refer to http://help.vocareum.com/article/30-getting-started-students to get started with the system.
- For BFS-DFS, whenever you want to insert nodes to your frontier, insert them in alphabetical order and then remove them according to the algorithm. Also for UCS, upon choosing a node from the frontier, in case of ties, choose the one that comes first in alphabetical order.
- For DFS, you should not visit a node that has already been visited to avoid the infinite-loop issue.
- For UCS, you can pop a node from the frontier only if the pipe from current node to that particular node is active at that time.
- It's not possible for a node to be both source and destination.
- There will be one source node and at least one destination node. There is no bound on maximum number of destination/middle nodes and pipes.
- Names of the nodes (source, destinations and middle nodes) are unique, case-sensitive and are all alphabetical strings. (Just uppercase letters)
- Pipe lengths are positive integers.
- A pipe can have multiple off-times. The off-times will be specified in 0-23 hour format. So if the time goes to 24, revert it back to 0.
- First and second numbers in pipe off-periods are both positive integers and not equal. Also the second number is always greater than the first number.
- There may be some overlapping periods when the pipes don't work too, i.e. a pipe can have off-periods like 2-4 and 3-7.

- The input file given to your program will not contain any formatting errors, so it is not necessary to check for those.

Output:

Create a file named "output.txt". For each test-case, one per line, report the name of the destination node (case-sensitive, uppercase only) where water reached first and also the time that it reached that node. These two fields should be separated by a space. If no path was found to any of the destination node, i.e. none of the destination nodes can be reached, print "None".

Output example:

```
1  B 4
2  BA 6
3
```

NOTES:

- Keep in mind that even though you are ignoring the pipe lengths and off-periods for DFS and BFS, water will start flowing in a specific time which is not always 0, so you have to consider this time-offset.
- We will be using one input file and one output file for grading. So check for possible exceptions in your code and make sure you always produce an output file. No output file will result in a 0 score for the assignment.
- The final test cases would be different from the sample test cases provided. Your assignment will be graded based on the performance on the final test cases only.