Week 13 Assignment

Submit Assignment

Due Monday by 6:40pm **Points** 10 **Submitting** a text entry box or a file upload

This week's assignment is a group project. You have the ability to assign yourself to groups and make changes if necessary. All students in the group must participate in the work and understand the submission. Please choose a leader of the group who will facilitate the work and be prepared to explain any one of the solutions to the class in our session on Monday evening. If your group feels that any of the solutions is incorrect and wishes to not have to explain it to the class, please indicate this in your submission. I will pick which groups present each problem.

For each of the problems below, your work will be much less if you use the Python code that is provided with this week's module. For some of these problems, you won't have to write much new code at all.

Problem 0.

Design and implement an algorithm to find the shortest paths between every vertex and a single destination. What is the worst case running time of your algorithm. In a few sentences, describe your approach. (For this problem, if you use the code that has been provided, you are not writing much of your own code.)

Problem 1.

gn and implement an algorithm that determines the shortest path between any one of a set of source vertices and any one of a set of destination vertices. In a few sentences, describe your approach.

Problem 2.

Archeologists have completed their exploration of a dig site and have collected fossils from the Paleolithic and Mesozoic eras. A fossil is the remnant of a prehistoric animal. They wish to classify the fossils into the two groups but it is very hard for them to directly label any one fossil. So, they decide to adopt the following approach.

- For each pair of fossils i and j, they study them carefully side by side. If they're confident enough in their judgment, then they label the pair (i, j) either *same* (meaning they believe both are from the

same era) or *different* (meaning they believe they are from different eras). They also have the option of rendering no judgment on a given pair.
- So they have the collection of n fossils, as well as a collection of m judgments for the pairs. They'd like to know if this data is consistent with the idea that each fossil is from the Paleolithic era or the Mesozoic era.
- So more specifically, we'll declare the m judgments to be consistent if it is possible to label each specimen either Paleolithic or Mesozoic in such a way that for each pair (i, j) labeled same it is the case that i and j have the same label; and for each pair (i, j) labeled different, it is the case that i and j have different labels.

Design and code an efficient algorithm that determines whether the m judgments are consistent. Use the attached driver to implement your solution. Alternatively, if you wish to use Python, make sure that you run your algorithm using the test cases included in the attached file. Include in your submission:

1. A brief description of your approach and the worst case running time of your solution.
2. The output of running your program using the test cases provided
3. All of the code that you have written

judgements.py

Problem 3.

The archaeologists hired specialists to analyze the fossils, which we'll denote F1, F2, ..., Fn. A fossil is the remnant of a prehistoric animal. The specialists have made conclusions about when the animals that gave rise to the fossils lived relative to one another. Each conclusion has one of the following two forms:

- For some i and j, the animal that gave rise to Fi died **before** animal Fj was born; or
- for some i and j, the life spans of the animals that gave rise to Fi and Fj **overlap**ped at least partially.

The archaeologists want to check on these conclusions before they publish their results. So, they'd like you to determine whether the conclusions are at least internally consistent, in the sense that there could have existed a set of animals for which all these conclusions simultaneously hold. Give an efficient algorithm to do this task. If the conclusions are consistent, produce a list of potential dates of birth and death for each of the n animals so that all the facts hold true.

Your submission for this assignment must include:

1. The worst-case running time of your solution
2. A brief description of your approach and an indication of whether your code works

3. The output of running your algorithm using the *judgments* included in the file accompanying this assignment
4. All of the code that you wrote.

Sample Output

```
Fossil 3: birth -- Year: 0 Day of Year: 1

Fossil 3: death -- Year: 0 Day of Year: 2

Fossil 0: birth -- Year: 0 Day of Year: 3

Fossil 0: death -- Year: 0 Day of Year: 4

Fossil 1: birth -- Year: 0 Day of Year: 5

Fossil 2: birth -- Year: 0 Day of Year: 6

Fossil 2: death -- Year: 0 Day of Year: 7

Fossil 1: death -- Year: 0 Day of Year: 8
```

precedence.py

Problem 4.

Shortest path with one skippable edge. Given an edge-weighted digraph, design an *E* log *V* algorithm to find a shortest path from *s* to *t* where you can change the weight of any one edge to zero. Assume the edge weights are nonnegative.

The shortest path from s to t is s -> w -> t (weight 11) but the the shortest path with one skipped edge is > u -> v -> t (weight 3).

Your submission must include:

1. The worst-case running time of your solution
2. A brief description of your approach and an indication of whether your code works
3. The output of running your algorithm using the *graph skippable.txt* included in the file accompanying this assignment
4. All of the code that you wrote.

spt.png

skippable.txt

Problem 5.

The engineers of an internet service provider have designed a network of routers that serve their clients. They have modeled their network as an acyclic connected graph (a tree). They want to calculate the maximum number of hops between routers. Develop an efficient algorithm to calculate the maximum number of hops between endpoints in their network (this corresponds to the longest path). Implement your algorithm in Python and indicate the worst case running time of your algorithm.

Problem 6.

A consortium of international universities is doing a joint project on genetic sequencing. An ultra-high speed network is built to connect these universities using a tree-based structure. The schools decide to install a supercomputer at one of the schools to share and crunch massive datasets. They wish to choose a “central” location for the supercomputer. Given a tree T and a node v of T, let L(v) denote the length of a longest path from v to any other node of T. A node of T with minimum L(v) is called the optimal centrex of T.

1. Design and implement an efficient algorithm that, given an n-node tree T, computes the optimal centrex of T. Indicate the worst-case running time of your algorithm.
2. Is the optimal centrex unique? If not, how many distinct optimal centrexes can a tree have?