DSC 40B Homework 05

Due: Wednesday, February 18

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justi cation for your answer. Homeworks are due via Gradescope at 11:59 p.m.

Problem 1.

Suppose we are given two (unsorted) arrays of real valued numbers A and B, where $|A| = n$ and $|B| = m$. Suppose we wish to merge these two arranges into a single array C, but remove duplicates. We call this the MergeArray problem. For example, $A = \ 3, 12, 5\}$ and $B = \ 7, 12, 9, 22, 5\}$. Then the merged array C should contain the list of numbers $\ 3, 12, 5, 7, 9, 22\}$ (note that numbers do not need to be in this order).

Describe an algorithm to solve this MergeArray problem (using 6 sentences or less): you can use any data structure or procedures we have descried in class. Provide the time complexity analysis of your answer. Slower algorithms receive fewer points.

Problem 2.

In the following, let

$$V = \ 0, 1, 2, 3, 4, 5, 6, 7\},$$
$$E = \ (0, 1), (1, 5), (5, 2), (0, 5), (7, 3), (6, 7)\}$$

For this problem, you do not need to show your work.

a Draw the *undirected* graph $G = (V, E)$. Remember that when writing the edges of an undirected graph, we often abuse notation and write (u, v) when we really mean $\ u, v\}$; we have done so here.

b Draw the graph $G = (V, E)$, assuming that G is directed.

c Write down the connected components of G, assuming that G is undirected.

d Write the adjacency matrix representation of G, assuming that G is undirected.

e Write the adjacency matrix representation of G, assuming that G is directed.

Problem 3.

Suppose A is the adjacency matrix of an undirected graph. Let A be the *squared* matrix, obtained by matrix multiplying A by itself. Show that the (i, j) entry of A is the number of ways to get from i to j in exactly two hops (i.e., the number of paths of length two between node i and node j). *Hint* Consult your linear algebra notes/textbook to remember a formula for the (i, j) entry of the product of two matrices.

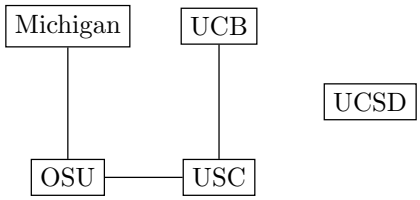
Problem 4.

Suppose we wish to develop an algorithm HasCompOfSize G, k), which, given a undirected graph $G = (V, E)$ and an integer k, we wish to return whether there exists a connected component of G whose size is **exactly** k. Here, the size of a graph is simply the number of nodes in this graph (and note that a connected component is itself a graph, which is subgraph of G).

Describe an efficient algorithm for this test HasCompOfSize G, k) (in 6 sentences or less): you can refer to / use any procedures described in class. Give the time complexity of your algorithm; slower algorithms receive fewer points.

Programming Problem 1.

We can use a graph to represent rivalries between universities. Each node in the graph is a university, and an edge exists between two nodes if those two schools are rivals. For instance, the graph below represents the fact that OSU and Michigan are rivals, OSU and USC are rivals, UCB and USC are rivals, but UCSD does not have a rival.



In a le called `assign_good_and_evil.py`, write a function `assign_good_and_evil(graph)` which determines if it is possible to label each university as either "good" or "evil" such that every rivalry is between a "good" school and an "evil" school. The input to the function will be a graph of type `UndirectedGraph` from the `dsc40graph` package. If there is a way to label each node as "good" and "evil" so that every rivaly is between a "good" school and an "evil" school, your function should return it as a dictionary mapping each node to a string, `good` or `evil` ; if such a labeling is not possible, your function should return `None`.

For example:

```
>>> example_graph = dsc40graph.UndirectedGraph()
>>> example_graph.add_edge( Michigan ,  OSU )
>>> example_graph.add_edge( USC ,  OSU )
>>> example_graph.add_edge( USC ,  UCB )
>>> example_graph.add_node( UCSD )
>>> assign_good_and_evil(example_graph)
{
     OSU :  good ,
     Michigan :  evil ,
     USC :  evil ,
     UCB :  good ,
     UCSD :  good
}
```

If in the above graph, there is also an edge between `Michigan` and `USC` , then there does not exist such a label and your funtcion should return `None`.

Of course, there might be several ways to label the graph – your code need only return one labeling.

You can install `dsc40graph` by following one of the methods mentioned on lecture 10, slide 44.