IEOR 4008: Computational Discrete Optimization 02/21/2020

Assignment 2

Deadline: 03/04/2020, 1:10pm

Submit electronically via canvas. Depending on the exercise, submit a pdf document and / or the code.

Exercise 1

Implement the 2-OPT algorithm for TSP. For each $n \in \{30, 40, 50\}$, create 20 random instances of Euclidean TSP. For each of those instances, run the greedy algorithm and, starting from the optimal solution of the greedy algorithm, run 2-OPT. For each value of n, record the percentage improvement that 2-OPT gives over greedy.

Exercise 2

Consider a set $s_1, \ldots, s_n$ of current species. A *phylogeny* is a tree representation of the evolution that led, from a common unique ancestor, to $s_1, \ldots, s_n$.

The exact phylogeny is usually unknown, and it is an important problem in computational biology to find the one that is most likely. One of the prominent methods to reconstruct the phylogeny is the following: The input is given by a matrix $M \in \{0,1\}^{n \times m}$, where each row represents a species, and each column a character (e.g. the presence of a certain gene in the DNA, or a physical characteristic), and entry $M[i,j]$ is 1 if species i has character j, and 0 otherwise (see Figure 1, left). Figure 1, right gives a possible phylogeny of four species (lamprey, shark, salmon, lizard). It assumes that:

(*) all the species evolved from a common ancestor species a_1, where none of the characters are present, through a sequence of successive ancestors.
(**) Each species evolved from a previous ancestor species by developing one or more characters, or after one or more characters have disappeared.
(***) no two species with the exactly the same characters appear at any time during the evolution.

In the tree T in Figure 1, a_2 evolved from a_1 by developing characters a, b, and shark, salmon, lizard evolved from the common ancestor a_2. Salmon and lizard evolved from the common ancestor a_3, that has characters a, b, c. Points at which characters emerge are indicated by labeled bars. A *State change* in T is the emergence or the disappearance of a certain character between two different species. For instance, between ancestors a_1 and a_2, there are two state changes. T has in total 7 state changes.

a) Find a phylogeny tree for the instance in Figure 1 that respects hypothesis (*), (**), (***) and has at least 8 state changes.

b) A *most parsimonious tree* is a phylogeny tree that respects conditions (*),(**), (***) and has a minimum number of state changes. Those trees are often considered very good candidates for representing the real evolution of the species. Give a graph G and a distance function c such that the problem of finding the most parsimonious tree for the input in Figure 1, left can be formulated as a Steiner Minimum Tree problem with respect to a distance c.

c) Show that the problem of finding a most parsimonious tree for a generic input $M \in \{0,1\}^{m \times n}$ can be formulated as a Steiner Minimum Tree problem.

	a	b	c	d	e	f
lamprey	0	0	0	0	0	1
shark	1	1	0	1	0	0
salmon	1	1	1	1	0	0
lizard	1	1	1	0	1	0

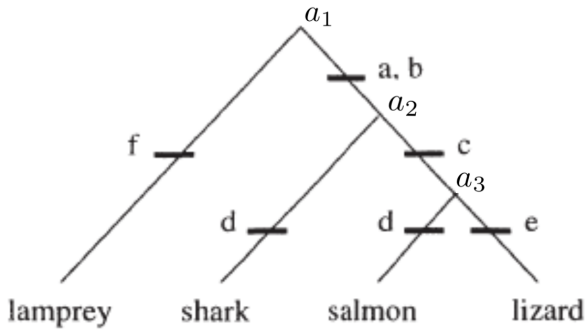

Figure 1: The data from Exercise 2

Exercise 3
The file *graph-to-color.txt* contains the edges of a graph G with 189 nodes. Find a feasible coloring for G with as few colors as you can. Submit all the codes and algorithms you implemented for solving this problem. *NB: you are allowed to reuse code seen in class and to write your own code, but not to use pre-built coloring algorithms available in python libraries.*

Exercise 4
In class, we saw how to use Dynamic Programming to find the length of the *longest common subsequence* of two given strings S and T. The common subsequence does not necessarily need to be contiguous: for example, if $S = abcxe$ and $T = abce$, then the longest common subsequence is $abce$ of length 4. In this exercise, we consider a related but different problem, *longest common substring*. Again we are given two strings S and T, but this time we want to find the length of the longest common *substring* that has to be contiguous: when $S = abcxe$ and $T = abce$, the longest common substring is abc of length 3. Give a DP algorithm to find the length of the longest common substring given two strings S and T. It is enough to provide the recursive formula and explanations.