HW 3 (due Monday, 6pm, September 21, 2015)

NEW CS 473: Theory II, Fall 2015 Version: **1.04**

Collaboration Policy: For this homework, Problems 1–2 can be worked in groups of up to three students. Submission is online on moodle.

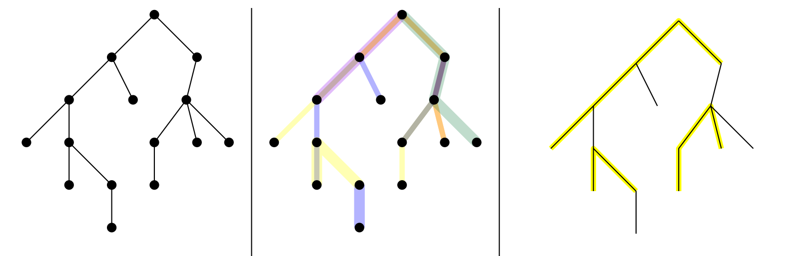
---

1. (50 PTS.) Packing heavy snakes on a tree.

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a given rooted tree with n vertices. You are given also t snakes $s_1, \ldots, s_t$, where a ***snake*** is just a simple path in the tree (with fixed vertices – a snake has only a single location where it can be placed). Every snake s_i has associated weight w_i, and your purpose is to pick the maximum weight subset S of snakes (of the given snakes) such that (i) the weight of the set is maximized, and (ii) no two snakes of S share a vertex. We refer to such a set S as a ***packing*** of snakes.

Describe an efficient algorithm (i.e., provide pseudo-code, etc), as fast as possible, for computing the maximum weight snake packing. (You can not assume G is a binary tree - a node might have arbitrary number of children.) What is the running time of your algorithm as function of n?

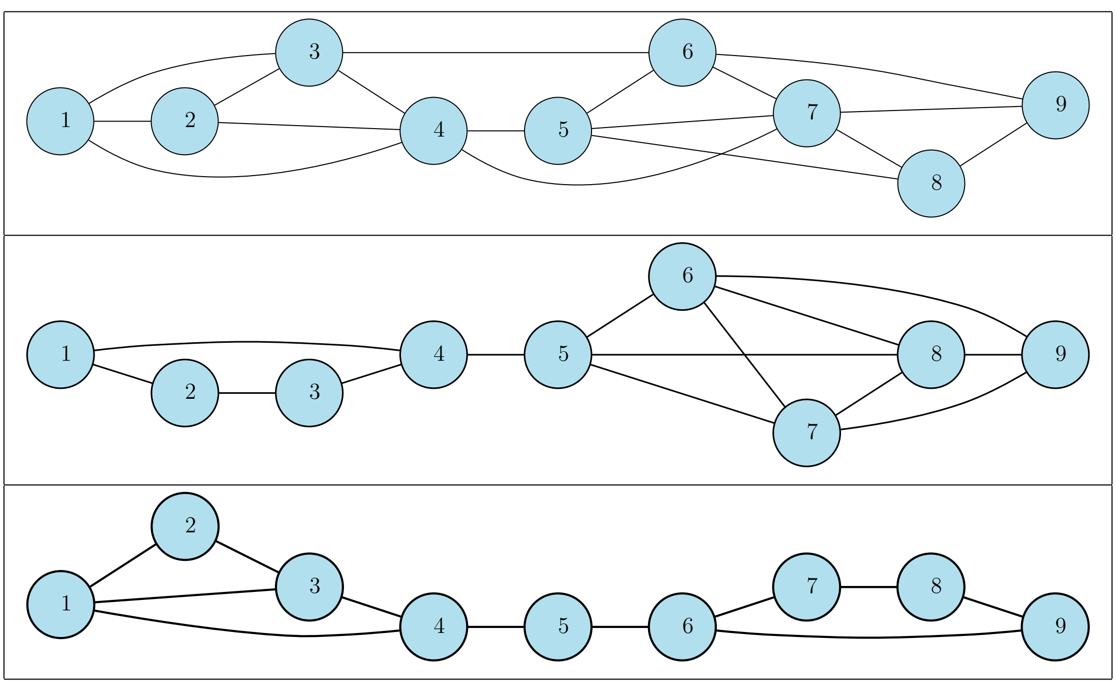
For example, the following shows a tree with a possible snake packing.



(A) The tree | (B) Input snakes | (C) Possible solution.

2. (50 PTS.) Coloring silly graphs.

A graph G with $\mathsf{V}(\mathsf{G}) = \{1, \ldots, n\}$ is k-silly, if for every edge $ij \in \mathsf{E}(\mathsf{G})$, we have that $|i - j| \leq k$ (note, that it is not true that if $|i - j| \leq k$ then ij must be an edge in the graph!). Here are a few examples of a 3-silly graph:



Note, that only the last graph in the above example is 3-colorable.

Consider the decision problem 3COLORSillyGraph of deciding if a given k-silly graphs is 3-colorable.

(A) (20 PTS.) Prove that 3COLORSillyGraph is NP-COMPLETE.

(B) (30 PTS.) Provide an algorithm, as fast as possible, for solving 3COLORSillyGraph. What is the dependency of the running time of your algorithm on the parameter k?

In particular, for credit, your solution for this problem should be have polynomial time for k which is a constant. For full credit, the running time of your algorithm should be $O(f(k)n)$, where $f(k)$ is some function of k.

Hint: (A) Think about the vertices as ordered from left to right as above. Start with $k = 2$. Then, solve the problem for $k = 3, 4, \ldots$. Hopefully, by the time you hit $k = 5$ you would be able to describe an algorithm for the general case.