Problem Set 10

Data Structures and Algorithms, Fall 2019

Due: November 28, in class.

From CLRS

Exercise 22.5-3, 22.5-7 (solution of 22.5-5 might help you), Problem 22-4. Exercise 23.1-2, 23.1-8, 23.2-4, 23.2-8. Problem 23-1. Exercise 16.1-3, 16.2-4.

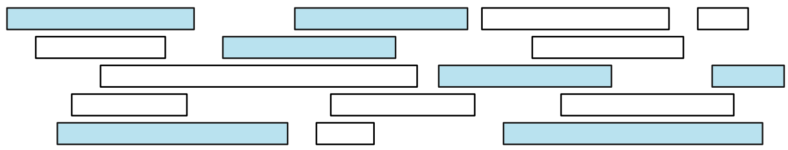
Additional Problem One

A "bite-of-each-cycle edge set" of an undirected graph G is a subset F of the edges such that every cycle in G contains at least one edge in F. In other words, removing every edge in F makes the graph G acyclic. Describe and analyze a fast algorithm to compute a minimum-weight bite-of-each-cycle edge set of a given edge-weighted graph.

Additional Problem Two

Let X be a set of n intervals on the real line. We say that a subset of intervals $Y \subseteq X$ *covers* X if the union of all intervals in Y is equal to the union of all intervals in X. The size of a cover is just the number of intervals in the cover. See following figure for an example.

Describe and analyze an efficient algorithm to compute the smallest cover of X. Assume that your input consists of two arrays $L[1 \cdots n]$ and $R[1 \cdots n]$, representing the left and right endpoints of the intervals in X. (For simplicity, you may assume all $2n$ endpoints are distinct.) If you use a greedy algorithm, you must prove that it is correct.



A set of intervals, with a cover (shaded) of size 7.

Bonus Problem One[1]

Imagine you want to host a party and you have n friends to potentially invite. You have made up a list of which pairs of these people know each other. You want to invite as many people as possible, subject to two constraints: at the party, each person should have at least six other people whom they know and six other people whom they don't know.

Give an efficient algorithm that takes as input the list of n people and the list of pairs who know each other and outputs the best choice of party invitees. Argue your algorithm is correct and give the running time in terms of n. (*Hint: The basic idea of the solution is not complicated.*)

[1] You are *not* required to solve this problem.