CSC373 Fall'19
Tutorial 2
Mon. Sept. 23, 2019

Q1 Interval scheduling and partitions revisited

Given m machines and a set of intervals S with start times s_i and finishing times f_i, output a feasible mapping $\sigma : S \to \{0, 1, 2, \ldots, m\}$ where $\sigma(I) = k > 0$ means that interval I is schedule on machine k and $\sigma(I) = 0$ means that interval I is not scheduled. The mapping is feasible if $\sigma(I_i) = \sigma(I_j) = k > 0$ implies I_i and I_j do not intersect (the intervals are compatible). The objective is to maximize $|I : \sigma(I) > 0|$.

Q2 Making change

Consider the problem of making change for some amount of money n, given denominations $1 = C[1] < C[2] < ... < C[k]$ (for example, Canadian coins come in denominations $C[1] = 1, C[2] = 5, C[3] = 10, C[4] = 25, C[5] = 100, C[6] = 200$). Say that we want the output to be a sequence of coins $S = u_1, u_2, ..., u_m$ (where each $u_i = C[j]$ for some j) such that we use as few coins as possible (i.e., $n = u_1 + u_2 + ... + u_m$ and m is minimal).

Give a greedy algorithm that makes change for any amount $n >= 0$ using as few coins as possible, for the Canadian denominations. What is the runtime of your algorithm? Prove that your algorithm always returns an optimal answer.

Does your algorithm still work correctly if we add a 75 cents coin, i.e., for denominations $C[1] = 1, C[2] = 5, C[3] = 10, C[4] = 25, C[5] = 75, C[6] = 100, C[7] = 200$? What if we add a 30 cents coin instead of a 75 cents coin?

Q3 MSTs

(a) Prove or disprove: If e is a minimum-weight edge in connected graph G (where not all edge weights are necessarily distinct), then every minimum spanning tree of G contains e.

(b) Does your answer change if w(e) is unique (no other edge in G has the same weight as e, but w(e) is still the smallest)? Again, prove your answer.

Q4 Cops and robbers

Given an array of size n that has the following specifications: Each element in the array contains either a cop or a robber. Each cop can catch only one robber. A cop cannot catch a robber who is more than K units away from the cop. Write an algorithm to find the maximum number of robbers that can be caught.