

CMPSCI 611 FALL '09: HOMEWORK 2
DUE 2:30 PM, OCTOBER 15TH

- Collaborating with at most two other students is allowed but answers must be written independently. Solutions must be typed (ideally in Latex). Please write with whom you collaborated on your solutions. Unfortunately, you are not allowed to use material from the net or talk about the homework with anybody outside your collaboration group (aside from the lecturer or TA of course.)
- Solutions are due before 2:31 pm (according to the clock in the classroom) on the due date. Marks for solutions that are returned up to 24 hours late will be reduced by 25%. After that no credit will be given. This policy will be strictly enforced.
- Solutions can either
 - Be emailed to `mcgregor@cs.umass.edu` and `smurtagh@cs.umass.edu`. Use the subject line “CMPSCI 611 Homework 2” or
 - Be printed and handed in at the start of class.
- To get full marks, answers must be sufficiently detailed, supported with rigorous mathematical proofs, and be clearly explained.

Question 1 (10 marks). Consider a length n sequence of values $P[1], P[2], \dots, P[n]$. An increasing subsequence of length k has the form $P[i_1]P[i_2] \dots P[i_k]$ where $i_1 < i_2 < \dots < i_k$ and

$$P[i_j] < P[i_{j+1}] \text{ for each } j = 1, 2, \dots, k - 1$$

Design an efficient algorithm that finds the length of the longest increasing subsequence subject to the condition that $i_1 = 1$.

Question 2 (10 marks). Let (E, \mathcal{I}) be a subset system for bipartite matching and let $w : E \rightarrow \mathbb{R}^+$ be a weight function. Let i be the subset returned by the greedy algorithm and let i' be a maximum solution. Prove that $w(i) \leq w(i') \leq 2w(i)$.

Question 3 (10 marks). Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is a positive integer.

- (1) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.
- (2) While traveling you come across a country whose coins have the values c^0, c^1, \dots, c^k for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.
- (3) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution. Your set should include a penny so that there is a solution for every value of n .
- (4) Give an $O(nk)$ time algorithm that makes change for any set of k different coin denominations, assuming that one of the coins is a penny.

Note that the greedy algorithms don't need to be analyzed via matroid techniques.

Question 4 (10 marks). Given a subset system and a weight function, there can be multiple maximum weight solutions.

- (1) Let (E, \mathcal{I}) be a matroid. Prove that if the elements of E are assigned distinct weights (i.e., no two weights are equal), then the maximum weight subset in \mathcal{I} is unique.

- (2) Now consider an assignment of weights such that there is more than one subset in \mathcal{I} that achieves the maximum weight. From the first part, we know that such an assignment must have weights that are not distinct. Thus, when we sort the elements of E by non-increasing weight at the start of the greedy algorithm, there are multiple sorted orders that can result, depending on how we break ties. In fact, how we break ties can effect which of the optimal solutions the greedy algorithm returns. Prove that for every subset $i \in \mathcal{I}$ that achieves the maximum weight, there is some way to break ties between elements of E with the same weight, such that the greedy algorithm returns the solution i .

Question 5 (10 marks). Consider a directed graph $G = (V, E)$ where each edge e has weight w_e and a special node $t \in V$. The weights may be positive and negative. A friend has computed values $d[v]$ for each $v \in V$ and claims that $d[v] = \delta(v, t)$, i.e., that the length of the shortest path from v to t is $d[v]$. All the $d[v]$ values are finite.

- (1) Give an $O(|E|)$ time algorithm that determines whether your friend has correctly computed the values. Note that your algorithm just needs to verify the claimed values, i.e., if your friend is wrong, you just need to identify this fact rather than finding the correct values.
- (2) Assume your friend is correct. Now you need to compute distances to a different node $t' \neq t$. Give an $O(|E| + |V| \log |V|)$ time algorithm for computing $\delta(v, t')$ for all $v \in V$. You may assume that Dijkstra's algorithm runs in $O(|E| + |V| \log |V|)$ times. **Hint:** It may be useful to consider a new weight function defined as follows: for edge $e = (v, w)$, let $w'_e = w_e - d(v) + d(w)$.