CS325 Winter 2017

Assignment 4

February 27, 2017

This set of questions focuses on:

- greedy algorithms, MST and hoffman coding
- the proof techniques for proving the optimality of the greedy algorithm (arguing that greedy stay ahead). The exchange argument. Proof by contradiction.

1. Prove (by contradiction) that if the weights of the edges of G are unique then there is a unique MST of G. Also, show that the converse is not true by giving a counterexample.
   Given a graph G with unique edge weights, for the sake of contradiction, we assume there are two MSTs T_1 and T_2 that have different sets of edges.
   Let $e = (v, w)$ be the cheapest edge that is only in one of the MSTs (say it's in T_1, not in T_2).
   Now T_2 must contain a path from v to w. Adding e to T_2 will create a cycle.
   On this cycle, there must be another edge f that is not in T_1 (otherwise T_1 would contain a cycle). So f is in T_2 but not in T_1.
   By the construction of e (being the cheapest edge in only one of the two MST's), we know the weight of e must be smaller than that of f. So if we replace f with e, we will have a cheaper spanning tree than T_2 contradicting to the optimality of T_2. As such, we can conclude that it is not possible to have two different MST's for G.
   To show the converse to be false, consider a graph with 2 edges, both of weight 1. It has a unique MST, but its edge weights are not unique.

2. The following statements may or may not be correct. In each case, either prove it (if it is correct) or give a counterexample (if it isn't correct). Always assume that the graph $G = (V; E)$ is undirected. Do not assume that edge weights are distinct unless this is specifically stated. Note that lightest = cheapest, heaviest = most expensive.

   a. If graph G has more than $|V| - 1$ edges, and there is a unique heaviest edge, then this edge cannot be part of a minimum spanning tree.

   b. If G has a cycle that has a unique heaviest edge e, then e cannot be part of any MST.

   c. Let e be any edge of minimum weight in G. Then e must be part of some MST.

   d. If the lightest edge in a graph is unique, then it must be part of every MST.

   Solutions:

   a. *False, consider the case where the heaviest edge is a bridge (is the only edge connecting two connected components of G.*

   b. *True. Assuming false, removing e from the MST will break the tree into two parts, and adding another edge across the two parts will re-connect the tree and we get a new tree with less total weight, a contradiction.*

   c. *True, e will belong to the MST produced by Kruskal.*

   d. *True. Assuming there is a MST that does not include this edge (e), adding e to the tree will cause a cycle, removing another edge in the cycle will produce a lighter tree.*

3. Textbook 5.14

   a $a \leftarrow 0$, $b \leftarrow 10$, $c \leftarrow 110$, $d \leftarrow 1110$, $e \leftarrow 1111$.

   b *length*= $\frac{1000{,}000}{2} \times 1 + \frac{1000{,}000}{4} \times 2 + \frac{1000{,}000}{8} \times 3 + \frac{1000{,}000}{16} \times 4 + \frac{1000{,}000}{4} \times 4 = 1875000$

4. Textbook 5.15

   a *$f_a = 1/2, f_b = 1/4, f_c = 1/4$ gives this code.*

   b *This is not a prefix free code.*

   c *This is not a full binary tree, thus cannot be an optimal code.*

5. Given a set of points $\{x_1 \leq x_2 \leq ... \leq x_n\}$ on the real line. The goal is to use a smallest set of unit-length closed intervals to cover all of the points. For example, for inputs $\{0.5 \leq 1.4 \leq 1.55 \leq 1.6 \leq 2.5\}$. An optimal solution contains two intervals $[0.45, 1.45]$ and $[1.55, 2.55]$. The first interval covers the first two points, whereas the remaining points are covered by the second interval. Consider the following greedy algorithm.

   > The greedy algorithm simply works by starting the first interval at x_1. It then removes all the points in $[x_1, x_1 + 1]$, then repeat this process on the remaining points.

   For example input $\{0.5 \leq 1.4 \leq 1.55 \leq 1.6 \leq 2.5\}$, the first interval will be $[0.5, 1.5]$. We then skip the second the point and start a new interval at the third point $[1.55, 2.55]$.

   Suppose our greedy algorithm output a solution with p unit-length closed intervals: $I_1, I_2, ..., I_p$, ordered on the real line. Consider an optimal solution $J_1, J_2, ..., J_q$ ordered on the real line.

   - Prove that our algorithm stays ahead. That is, for $1 \leq k \leq \min(p, q)$, our first k intervals covers at least as many points as covered the $J_1, J_2, ..., J_k$. (Hint: use induction)
     We prove this by induction.
     Base case: *$k = 1$. By construction of our algorithm, among all unit-length intervals that covers x_1, I_1 has the largest possible starting position, thus its ending position will be no smaller than that of J_1 and thus cover no fewer points than J_1.*
     Inductive hypothesis: *Suppose the statement is true for $k = 1, ..., n$ for some $n \geq 1$*
     Inductive step: *Here we need to prove that the statement is true for $k = n + 1$. Consider the first point that is covered by I_{n+1}, let's call it x. Consider the first point covered by J_{n+1} and call it y. Based on the inductive assumption, we $I_1, ..., I_n$ covers more points than $J_1, ..., J_n$, thus we have $x \geq y$. Therefore, I_{n+1}'sending position must be greater than or equal to J_{n+1}'s ending position. As such we can conclude that $I_1, ..., I_{n+1}$ must cover more points than $J_1, ..., J_{n+1}$.*

     Q. E. D.

   - Prove that our solution is optimal building on the stay-ahead lemma.
     Suppose for the sake of contradiction that our solution is not optimal, that is $p > q$. Consider the first q intervals of our solution, based on our stay-ahead lemma, we know all the points that are covered by $J_1, ..., J_q$ must be covered by $I_1, ..., I_q$. This means that $I_{q+1}, ..., I_p$ covers points that the optimal solution does not cover, which is not possible because the optimal solution must cover all the points. So we must have $p \leq q$ and our algorithm must be optimal.