EECS 336: Algorithms	Homework 3 Solutions

1 Matroids

Assume $(E, \mathcal{I})$ is not a matroid. Then either the subset property does not hold, or the subset property does hold, but the augmentation property does not hold. We will argue these two cases separately.

Subset property does not hold: We know that there exists $J \in \mathcal{I}$ and $I \subset J$ such that $I \notin J$. Give all elements of I weight 2, all elements of $J \setminus I$ weight 1, and all other elements weight 0. The greedy algorithm will consider the elements of I, then the remaining elements of J, and the the remaining elements of E. Because the greedy algorithm maintains feasibility, it will not take all the elements of I. But then the greedy algorithm cannot take all the elements of J either. This means that greedy isn't optimal, as every optimal solution must contain every element of J.

Subset property does hold, but augmentation property does not: Then there exist $I, J \in \mathcal{I}$ such that $|I| < |J|$ but there is no $j \in J \setminus I$ such that $I \cup \{j\} \notin \mathcal{I}$. Assign all elements of I weight $1+\epsilon$, and all elements of J weight 1. Assign all other elements of E weight 0. The greedy algorithm will consider all elements of I first, and will add them all, by downward closure. However, it will not add any elements of $J \setminus I$, as the augmentation property is violated for I and J. It follows that greedy will return a set with total weight $|I|(1+\epsilon)$. For sufficiently small values of ϵ, the optimal solution will be J, with total weight $|J| > |I|(1+\epsilon)$. It follows that greedy is not optimal in this case.

2 Travel Time

modDijkstra(G,f,s)

> S = {}.
>
> for all v, insert(v, ∞).
>
> decreasekey(s,0)
>
> While Q not empty
>
> > (v,d) = deletemin()
> >
> > add v to S
> >
> > dist(v) = d
> >
> > for each u adjacent to v
> >
> > > decreasekey($u, f_{(v,u)}(d)$)

We'll show two things by induction. These claims together imply correctness.

1. On the jth iteration of the while loop, dist(v) is the travel time from s to v for all $v \in S$.

2. In addition, for all pairs in Q at that time, the key is the one-hop distance from s, where one-hop distance is the shortest travel time along one-hop paths.

Proof:

Base Case Before the first iteration, the claim is clearly true.

Inductive Hypothesis Assume the claim holds for all $j \leq k$

Inductive Case Let (v, d) be the k+1st pair removed from Q.

- Assume there's path P from s to v with travel time less than d.
- It can't be a one-hop path, by IH.
- Let u be first vertex where P leaves S.
- Let P' be P truncated at u
  * P' is the shortest path from s to u.
  * It is one-hop.
  * Hence u would be returned by deletemin()
- Contradiction
- To see part 2 of our claim, note that by the induction hypothesis, d is the one-hop distance of v, so if any vertex u has its key changed to $f_{(v,u)}(d)$ it is precisely because there is a one-hop path through v with that length.

Runtime: $O(m \log n)$. Initializing and filling the priority queue costs $O(n \log n)$. The remaining work comes from calls to decreasekey(). There are $O(m)$ such calls, resulting in a runtime of $O(m \log n)$. (This require sthe assumption of strong connetivity, in which case $n = O(m)$.