CS41 Lab 6: Greedy Algorithms

October 6, 2022

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

1. **Cell service on the Appalachian Trail.**
   The Appalachian Trail is an approximately 2100 mile trail that runs north-south from Maine to Georgia. Recently, several hikers have started to demand cell service along the trail. Other hikers object, assuming that cell phones will ruin the hiking experience. Leaders of the Appalachian Trail Conservatory, who manage the trail, have decided on a compromise – they plan to install cell phone base stations, but only for service at one of the camground areas on the trail. You've been hired to help decide where to place the base stations. Your goal is to place cell phone base stations at certain points on the trail, so that every campground is within five miles of the base stations.

   Give an efficient algorithm that achieves this goal, using as few base stations as possible. Your algorithm's input should be a List of campground locations, and you should output a List of base station locations that covers all campground locations using the minimal number of base stations.

2. **Dijkstra's Algorithm.** In CS35, you likely saw the *shortest path on weighted graphs* problem. In this problem, each edge e has an edge length ℓ_e, and the length of a path $s \rightsquigarrow t$ is the sum of the edge lengths along the path. Your goal is to find, for a specific start vertex, the length of the shortest $s \rightsquigarrow v$ path for all vertices v.

   Problem: Shortest Paths

   Inputs:

   - A graph $G = (V, E)$
   - For each edge e a positive *edge length* ℓ_e
   - A start vertex $s \in V$

   Output: an array of distances d, where $d[v]$ is the length of the shortest $s \rightsquigarrow v$ path.

   Below is pseudocode for Dijkstra's Algorithm, which finds the shortest path in a graph G between a start vertex s and any other vertex.

```
DIJKSTRA(G, s, $\ell$)
1  $S = \{s\}$.
2  $d[s] = 0$.
3  while $S \neq V$
4        pick $v \in V \setminus S$ to minimize $\min_{e=(u,v):u \in S} d[u] + \ell_e$.
5        add $v$ to $S$.
6        $d[v] = d[u] + \ell_e$
7  Return $d[\ldots]$.
```

(a) Show that Dijkstra's Algorithm solves the shortest path problem. (Hint: use the stays ahead method)

(b) What is the asymptotic running time of Dijkstra's algorithm?
If you were to implement it (in, say, C++) what data structures would you need? Would you need any additional data structures beyond structures you've seen from CS35? If so, try to design an implementation for them.

Note: the pseudocode given above is *high-level* pseudocode. One reason why this is high-level is because it doesn't specify how to compute the edge $e = (u, v)$ such that $u \in S$ that minimized $d[u] + \ell_e$. You'll need to understand how to compute this edge efficiently.

3. **Making change (foreign edition)**

Consider the problem of making change for n cents out of the fewest number of coins. Assume that n and the coin values are positive integers (cents).

(a) Describe a greedy algorithm to solve the problem using the EU coin denominations of 50 cents, 20 cents, 10 cents, 5 cents, 2 cents, and 1 cent. Prove your algorithm is optimal.

(b) Describe a greedy algorithm to solve the problem using the US coin denominations of quarters (25), dimes (10), nickels (5), and pennies (1). Prove your algorithm is optimal.

(c) Suppose the country of Algorithmland uses denominations that are powers of c for some integer c. This country uses $k+1$ denominations of $c^0, c^1, \ldots, c^k$. Show that your greedy algorithm works in Algorithmland as well.

(d) Design a currency system of your choosing with at least three coin denominations such that a greedy algorithm does *not* yield a minimum number of coins for some amount of m cents. Assume that one of your denominations has value one, so a solution exists for all values of m.