CMSC498K Homework 5 Solutions

Richard Matthew McCutchen

Problem 1

The coach should simply keep a sum of the numbers of the players who have left the train so far. The sum will not exceed $O(n^2)$, so it can be kept in $O(\log n)$ bits. At any time, the coach can compare her sum to $n(n+1)/2$, the sum of all the players, to determine the sum of the players still on the train. In this way, if the coach knows that at most one player remains on the train, she can determine that player's number or that no player remains.

Problem 2

In the United States, there are about 300 million people[1] and about 6 million pregnancies per year[2]. Assuming each pregnancy is visible for 6 months, there are 3 million visibly pregnant women at a time, which is 1% of the population. Assume that these women do not visit the mall substantially more or less than other people do.

Now, suppose that we spend 30 minutes of the visit walking in the hallways, during which we spot an average of one new person every 10 seconds, and the remaining 90 minutes shopping in stores, during which we spot an average of one new person every minute. Then we will see 270 people on average; assume the mall is large so that all of these people are distinct. It is reasonable to assume the number of people seen follows a Poisson distribution with mean 270. Each person seen is pregnant with probability .01, so the number W of pregnant women seen follows a Poisson distribution with mean 2.7. We can then calculate:

$$P(W \geq 3) = 1 - \sum_{x=0}^{2} \frac{2.7^x e^{-2.7}}{x!} \approx .51.$$

So, according to this analysis, it is not at all rare to spot three visibly pregnant women on a 2-hour visit to the mall.

Problem 3

Suppose for the sake of contradiction that some edge (x, y) is in the minimum spanning tree but not the relative neighbor graph. Its absence from the relative neighbor graph means that there exists a vertex z such that $d(z, x) < d(x, y)$ and $d(z, y) < d(x, y)$. Suppose we delete (x, y) from the MST, splitting it into two trees T_x and T_y containing x and y respectively, and then reconnect these trees by adding (z, x) if z is in T_y or (z, y) if z is in T_x. The result is a spanning tree that has strictly lower cost than our assumed MST, a contradiction. Therefore, the MST is a subgraph of the relative neighbor graph, as desired.

Problem 4

Observe that, given a partition of the input points into buckets, it is optimal to set the h value of each bucket to the midrange of the values in that bucket (the average of the minimum and the maximum), and the resulting cost

[1]`http://en.wikipedia.org/wiki/United_States`
[2]`http://www.americanpregnancy.org/main/statistics.html`

is half the greatest range of any bucket. Thus, it is enough to compute a partition that minimizes the greatest bucket range.

We can do this with dynamic programming. For each i and j, let $R(i,j)$ be the minimum greatest bucket range of a partition of the first i points into at most j buckets. We initialize $R(0,j) = 0$ and compute the remaining values from the following recurrence, remembering the operative u in each case:

$$R(i,j) = \min_{u=0}^{i} \max\left(R(u,j-1), \left(\max_{v=u+1}^{i} x_i \right) - \left(\min_{v=u+1}^{i} x_i \right) \right)$$

(u represents the number of points in the first $j-1$ buckets.) If we precompute the maximum and minimum of every interval of input points (which takes $O(n^2)$ time), then it takes $O(n)$ time to calculate each value $R(i,j)$. There are $O(kn)$ values, so the total time to compute R is $O(kn^2)$. Then, we go back and use the saved u values to construct the actual partition and choose the h values as mentioned earlier; this is faster.

A much faster approximate algorithm that does not use dynamic programming is also possible. Given an upper bound on the allowed bucket range, we can construct a k-bucket partition satisfying that bound (if one exists) in $O(n)$ time using a greedy left-to-right scan, extending each bucket as far as possible to the right without violating the bound. We use this procedure in a binary search to find a partition of minimum greatest bucket range. If M is the ratio of the range of the entire dataset to the smallest cost increment that we care about, then the binary search will find a good solution within $O(\log M)$ tries for a total running time of $O(n \log M)$.