USACO Review

SCT Officers

December 21 2018

1 USACO Gold, Teamwork

1.1 Problem Statement

We are given a list of numbers, representing the skills of up to N ($N \leq 10^4$) cows. Farmer John wants to split the cows into consecutive groups, with no group exceeding K ($K \leq 10^3$) cows. Each group's teamwork value will be the skill of the most skilled cow times the size of the group. What is the maximum total teamwork value of the groups.

1.2 Solution

This problem is a example of a dynamic programming. Note $N * K = 10^7$, which means that a DP solution of $O(NK)$ will work. Let's define a state $f(n)$ to be the best teamwork total value with cow n being the last cow in its group and the $n+1$ cow will start a new group (cow indexing starts at 1). Let's also define $f(0) = 0$ since if we don't have any cows, the teamwork value is 0. We are looking for $f(N)$, as there will be no more cows left at the $N+1$ index.

Now we've defined the states, let's define the transitions. By definition, the rightmost cow of the group ending at $f(n)$ is is fixed at cow n. We now want to determine how many cows are in the group such that $f(n)$ is maximized. Since we can have up to K cows in a group, we just have to iterate backwards, considering all group sizes s from 1 to K (or n if $n < K$). Since we are iterating backwards, the size of the group is increasing, so the max (denoted MX) skill of the group can simply be kept track of and updated. Thus, $f(n)$ will be the max of $f(n-s)$+MX$*s$ for all sizes s.

2 USACO Gold, Dining

2.1 Problem Statement

There are $2 \leq M \leq 50,000$ pastures, connected by $1 \leq M \leq 10^5$ edges. Edges are bidirectional and require time w_i to traverse. The barn is in pasture N. All the cows can and will reach the barn. There are also $1 \leq K \leq N$ haybales in various pastures. Each haybale has a yumminess y_i. Cows want to stop by a haybale on their route back to the barn. The cows can only stop at one haybale. They can only stop if the yumminess of that haybale is greater than or equal to the time the detour added to their trip.

Determine, for each pasture, whether a cow starting there can stop by a haybale on its trip to the barn.

2.2 Solution

Let $d(v)$ be the distance from node v to the barn. We can compute this for all nodes with a Dijkstra. For a cow at node a, and a haybale at node h, the cow can stop at h if

$$dist(a,h) + d(h) - d(a) \leq y_h$$

I will look at how the left side changes when I move to and adjacent node along an edge. I will travel from $u \to v$, with edge weight w. $dist(a,h)$ increases by w. $d(h)$ is unchanged. $d(a)$ changes by $(d(v) - d(u))$. At the haybale, the left side is zero. As I Dijkstra out from the haybale, the changes will accumulate. When the changes in the left side are greater than y_h, cows can no longer stop at that haybale. I will define a new edge weight C. For an edge $u \to v$ with weight w,

$$C = w + d(u) - d(v)$$

Then, I can Dijkstra from each haybale using C as the edge weight. All cows within y_h distance can use that haybale. Instead of running K Dijkstra's, one for each bale, I can use a multi-source Dijkstra. In this Dijkstra, I will set the "distance" to every haybale as $-y_h$. I then push all the haybales into the priority queue and use normal Dijkstra. When the Dijkstra finishes, all nodes with a "distance"≤ 0 can stop by a haybale. All others cannot. The multi-source Dijstra still runs in $O((E+V)\log E)$

3 USACO Gold, Compatibility

3.1 Problem Statement

There are N ($N \leq 50000$) cows, each of whom have a list of their five favorite ice cream flavors (each flavor is represented by an integer ID at most 10^6). Two cows are compatible if they share any flavor of ice cream in common. How many pairs of cows are not compatible?

3.2 Solution

First, lets see what happens if we only keep track of the number of occurrences of each flavor. We create a map from flavor to number of occurrences. For each cow, we add together the occurrences of each of its flavors to determine the number of cows it is compatible with. But wait! We have over-counted when it shares two flavors with a cow. Therefore we must also keep track of the occurrences of pairs of flavors, in order to subtract them. After this subtraction though, we have under-counted cows with triplets in common. Taking this logic to its limit, we must keep track of the occurrences of every subset of flavors (this will require $2^5 N$ map elements at most). For each cow, we will add the singletons, subtract the pairs, add the triplets, subtract the quadruplets, then add the quintuplet. This gives us the number of compatible cows, so we subtract this result from N. repeating this process for all N cows and adding the results, then dividing by two (so we count each pair once), we have our answer. This is an $O(n)$ solution.

4 USACO Platinum, Balance

4.1 Problem Statement

Bessie is performing on a balance beam. Positions are numbered [0,N+1], $1 \leq N \leq 10^5$. Bessie moves back and forth on the beam by flipping a fair coin to move left or right. Bessie can also quit instead of flipping the coin. Bessie will get some reward $f(x)$ for quitting the game at position x. If Bessie moves to positions 0 or N+1, the game is over and Bessie gets zero reward. Find the expected reward Bessie can get, starting from each position [1,N], assuming Bessie plays optimally (to maximize the expected reward).

4.2 Solution

We are trying to find $E(x)$, the expected value starting from position x. $E(x)$ is either $f(x)$, if Bessie quits at x, or $\frac{E(x-1)+E(x+1)}{2}$ if Bessie flips at x. $E(0) = E(N+1) = 0$. Because Bessie will pick the better option, $E(x)$ is at least the average of its neighbors. If $f(x)$ is greater than the average of the neighboring expected values, then Bessie will quit at x. If the were a concave part of $E(x)$, we could flip in the concave area to improve the expected value. Therefore $E(x)$ is convex. Bessie will quit at points in the convex hull of $f(x)$, so $E(x)$ is the convex hull of $f(x)$.

We can easily build the convex hull of $f(x)$ because the points $(x, f(x))$ are already sorted by x. We can keep a stack of the points on the convex hull. We scan across from x=0 to x=N+1. Before putting a point x onto the stack, look at the previous point. If the previous point is concave with respect to the point below it in the stack and x, the remove the previous point from the stack. Repeat this until the previous point is x=0 or it is convex. Then push x onto the stack. This is $O(N)$ because every point is pushed and popped once (or left on the stack).

The stack now has the points of the convex hull, with x=N+1 on top. $E(x)$ is on the line between the convex hull points on either side. A simple sweep can find all the $E(x)$ in $O(N)$ time.