Section 5: Advanced Dynamic Programming Solutions

CS 124

Spring 2021

1 Problem 1: Optimal Taxation

You are the king of a kingdom that's set up like a tree, and you want to collect some taxes. You know that each city has a "tax potential" t_i that you can collect. However, you don't want to tax adjacent cities, because they might talk to each other and begin a revolt. Find the maximum tax you can collect in linear time.

Solution: Let $\text{dp}[i][0]$ be the most revenue we can collect from the subtree at i when i is not used and let $\text{dp}[i][1]$ be the most revenue we can collect from the subtree at i when i is used.

Then, note that we have

$$\text{dp}[i][0] = \sum_{j \in N(i), j \neq p} \max\left(\text{dp}[j][0], \text{dp}[j][1]\right)$$

$$\text{dp}[i][1] = i + \sum_{j \in N(i), j \neq p} \text{dp}[j][0]$$

and now we simply use the DFS algorithm as mentioned before.

2 Problem 2: Stack the Blocks

We have n blocks, each with weight w_i and strength b_i. We stack the blocks on top of each other in some order.

The *power* of a block is its strength minus the sum of the weights which lie on it.

The *strength factor* of a stack is the minimum power of any particular block.

Give a $\mathcal{O}(n \cdot 2^n)$ algorithm to find the maximum possible strength factor across all permutations containing all of the blocks (note: answer might be negative).

Solution: First, we write the dynamic programming solution in the language of sets. Note that

$$\text{dp}[S] = \max_{i \in S} \min\left(\text{dp}[S \setminus \{i\}], b_i - \sum_{k \in S \setminus \{i\}} w_k\right) = \max_{i \in S} \min\left(\text{dp}[S \setminus \{i\}], b_i + w_i - \sum_{k \in S} w_k\right)$$

and $\text{dp}[] = 0$

which we can see by noticing that the strength factor of $S \setminus \{i\}$ stacked on top of i is given by the minimum of the strength factor of $S \setminus \{i\}$ and the power of the bottom block. Implementing this in bitsets is not much more complicated, as one can simply iterate through each $i \in [0, 1, \ldots n-1]$ to see if its in S, and then perform the calculation.

Note: A variant of this problem appeared as USACO 2014 December Gold Contest Problem 1.

3 Problem 3: Longest Paths in a Tree

Find the length of the longest path in a tree, where the edges are weighted (and possibly negative).

Solution: Each path on a tree has a unique "highest" point v. We will essentially calculate the longest path whose highest point is given by v. First, however, since paths whose highest points are at v consist of two disjoint "vertical paths", we first compute the longest vertical paths that end at each vertex. $\mathrm{dp}[i]$ is the length of the longest "vertical path" whose highest point is i.

Then, note that we have

$$\mathrm{dp}[i] = \max\left(0, \max_{j \in N(i), j \neq p} (\mathrm{dp}[j] + w_{ij})\right)$$

Now, for each vertex v, to find the longest path whose highest point is v, we simply compute

$$\ell[i] = \max\left(\mathrm{dp}[i], \max_{j,k \in N(i), j,k \neq p, j \neq k} ((\mathrm{dp}[j] + w_{ij}) + (\mathrm{dp}[k] + w_{ik}))\right).$$

which we can find by simply finding the two largest values of $\mathrm{dp}[j] + w_{ij}$ across all j. Both of these problems fit nicely into our framework, and we may conclude.