Bitmask DP

Helena Liu

6 March 2020

1 Traveling Salesman Problem

1.1 Problem

There are N cities ($2 \leq N < 20$). Given the distance between each pair of cities, find the shortest possible path that visits every city and returns to the origin city.

1.2 Brute Force Solution

Try every possible route that goes through every city. The complexity of this naive algorithm is $O(N!)$, which is too inefficient for the bounds of $2 \leq N < 20$, as 15! is already over 1 billion.

How can we make a faster algorithm? Assume we are comparing two different ways to go from City A to City B, both of which visit the same intermediate cities but in a different order. Logically, whichever one of these two paths is shorter will always be better than the other, and will always be the preferred path to take. Therefore, there is no reason to continue adding cities onto the longer path. Unlike the naive solution, the dynamic programming solution for this problem takes advantage of this.

1.3 Bitmask DP Solution

A bitmask is an N-length binary string that represents a possible state. In the case of the Traveling Salesman problem, the bitmask represents every city that we've visited so far. For example, if $N = 5$ and the bitmask is 10110, the cities we've visited are cities 1, 3, and 4.

Let's make a dp array of size 2^N, since the decimal value of the largest possible bitmask is $2^N - 1$. We also need another dimension of length N, because we need to record which city we are currently at. In the $N = 5$ example, if the bitmask is 10110, which is 22 in decimal, dp[22][4] will contain the shortest length possible of all paths that visit cities 1, 3, and 4, and end at city 4. We can iterate from 0 to $2^N - 1$ without missing any possible cases. This is because we need all subsets of a bitmask to be processed beforehand, and all subsets are guaranteed to be a smaller number, since they are less by some power of 2.

Algorithm 1 Traveling Salesman Bitmask DP

```
for i ← 0...2^N − 1 do
    for j ← 0...N do
        bitmask ← i
        for k ← 0...N do
            if (bitmask&1) == 0 then
                dp[i + 2^k][N − k − 1] ← min(dp[i + 2^k][N − k − 1], dp[i][j] + dist[j][N − k − 1])
            bitmask ← (bitmask >> 1)
```

2 Time Complexity

For the Traveling Salesman Problem, the time complexity is $O(N^2 * 2^N)$, because we iterate 2^N times, and with each iteration we try all N cities as our current city, and for each city we iterate through the N-length

bitmask to find the zeroes, which represent the cities that have not yet been visited. Bitmask DP always contains an $O(2^N)$ factor because each bit in the N-length string can be a 0 or a 1, and it always contains an $O(N)$ factor because the entire length of the bitmask must be processed to find the indexes of the zeros. Therefore, a general indicator to use Bitmask DP to solve a problem is when the bounds on N are very low, such as $N \leq 20$.

3 Problems

1. (USACO Gold December 2014, Guard Mark) Farmer John and his herd are playing frisbee. Bessie throws the frisbee down the field, but it's going straight to Mark the field hand on the other team! Mark has height H ($1 \leq H \leq 1,000,000,000$), but there are N cows on Bessie's team gathered around Mark ($2 \leq N \leq 20$). They can only catch the frisbee if they can stack up to be at least as high as Mark. Each of the N cows has a height, weight, and strength. A cow's strength indicates the maximum amount of total weight of the cows that can be stacked above her. Given these constraints, Bessie wants to know if it is possible for her team to build a tall enough stack to catch the frisbee, and if so, what is the maximum safety factor of such a stack. The safety factor of a stack is the amount of weight that can be added to the top of the stack without exceeding any cow's strength.

2. (USACO Gold January 2015, Moovie Mooving) Bessie is out at the movies. Being mischievous as always, she has decided to hide from Farmer John for L ($1 \leq L \leq 100,000,000$) minutes, during which time she wants to watch movies continuously. She has N ($1 \leq N \leq 20$) movies to choose from, each of which has a certain duration and a set of showtimes during the day. Bessie may enter and exit a movie at any time during one if its showtimes, but she does not want to ever visit the same movie twice, and she cannot switch to another showtime of the same movie that overlaps the current showtime. Help Bessie by determining if it is possible for her to achieve her goal of watching movies continuously from time 0 through time L. If it is, determine the minimum number of movies she needs to see to achieve this goal (Bessie gets confused with plot lines if she watches too many movies).

3. (USACO Gold November 2013, No Change) Farmer John is at the market to purchase supplies for his farm. He has in his pocket K coins ($1 \leq K \leq 16$), each with value in the range 1..100,000,000. FJ would like to make a sequence of N purchases ($1 \leq N \leq 100,000$), where the ith purchase costs $c(i)$ units of money ($1 \leq c(i) \leq 10,000$). As he makes this sequence of purchases, he can periodically stop and pay, with a single coin, for all the purchases made since his last payment (of course, the single coin he uses must be large enough to pay for all of these). Unfortunately, the vendors at the market are completely out of change, so whenever FJ uses a coin that is larger than the amount of money he owes, he sadly receives no changes in return! Please compute the maximum amount of money FJ can end up with after making his N purchases in sequence. Output -1 if it is impossible for FJ to make all of his purchases.