CS 4820, Spring 2017 Homework 4, Problem 2

Name: Yuxiang Peng

NetID: yp344

Collaborators: jl3455, zl542

(2) (12 points)

Tired from working on 4820 problem sets, you've decided to get a job this summer loading stacks of boxes into trucks. Although you were hoping this would take your mind off of dynamic programming at least for the summer, you soon discover that your hopes were in vain.

Here's the situation. A typical task consists of the following: there are n boxes at locations numbered 1, 2, . . . , n, and a truck at location 0. Box i starts at location i and has weight wi > 0. You need to move the boxes into a stack sitting in the truck, with the boxes stacked in order from 1 to n. (Box 1 is at the bottom of the stack and box n is at the top.) You could carry the boxes from their original locations to the truck one by one, but this would involve a lot of walking to and from the truck. It's more efficient to carry entire stacks of boxes at the same time if they're not too heavy.

A basic operation consists of the following: before the operation is performed, there is a (possibly empty) stack of boxes numbered i, i + 1, . . . , j − 2, j − 1 at location x, and a (nonempty) stack of boxes numbered j, j + 1, . . . , k − 2, k − 1 at location y. You lift up the second stack, carry it from location y to location x, and deposit it on top of the first stack. This operation results in a stack of boxes numbered i, i + 1, . . . , k − 2, k − 1, located at x, replacing the two stacks (one of which was possibly empty) that existed at the start of the operation. The contents of all other locations are unchanged. We'll denote the cost of this basic operation by c(x, y, w). It depends on how far apart locations x and y are, and on the total weight, w, of the stack that you're carrying from j to i. You can assume that

c(x, y, w) ≥ c(x, y, w0

) ≥ 0

for all locations x, y and weights w ≥ w

0 ≥ 0, and you can assume the triangle inequality

c(x, y, w) + c(y, z, w) ≥ c(x, z, w)

for all locations x, y, z and weights w. During a sequence of basic operations, you are allowed to store more than one stack of boxes at a single location.

Give an efficient algorithm to find the minimum cost of a sequence of basic operations that results in the boxes being stacked in the truck in the required order. You can assume that the input specifies the weight wi and initial location i of each box, and that it also contains a table listing the value of c(x, y, w) for every pair of locations x, y and weight w. Note that the problem does not ask you to output the optimal sequence of operations, only its cost.

Answer: Set F(i,k,x,label) as the minimum cost of moving box i to k-1 to the location x. Label is binary which has the value of 0 meaning the previous step could not have boxes i to k-1 as a whole in one location other than x and 1 meaning the previous step could have. The algorithm is as follows:

```
Algorithm:
Let t = k – 1
Initialize F(i,i+1,x,1)=0, if x=i; F(i,i+1,x,1)=c(x,i,wi), otherwise.
For t=1 to n
For i=1 to n+1-t
For x=0 to n
.F(i,k,x,0)=$\mathrm{Min}_y \mathrm{Min}_{i<j<k}$ {F(i, j, x, 1) + F(j, k, y, 1) + c(x, y, sum of wi(i is from j to k - 1)}
F(i,k,x,1)=Min{$\mathrm{Min}_y$ {F(i, k, y, 0)}, F(j, k, y, 1) + c(x, y, sum of wi(i is from i to k - 1)}
end
Return F(1,n+1,0,1)
```

Runtime: It is O(N^3) for the whole loop structure without considering the inner details. Inside the loop, we need to traverse n from both j and y so the complexity is O(N^2). So the total runtime is O(N^5).

Correctness:

For the recursive process, we have only two situations which are moving the stack of boxes of i to j-1 from location y to location x at the location of x and not having this done. The scenario of F(i,k,x,0) and F(i,k,x,1) are connected to the optimal choice of j and y both moving and not to make this choice relative to the last sentence of two situations. According to the question, the base case is k−i = 1, F(i,k,x,1) is right because F(i,i+1,x,1)=0, if x=i; F(i,i+1,x,1)=c(x,i,wi). When k − i <= n, F(i,k,x,1) which refers to not moving is right and it is the current optimal choice based on the question. As mentioned before, the algorithm is correct because of the correctness of both base case and recursive process.