CS 577: Introduction to Algorithms — DP Review

Solutions

1. i is the index of the city, 0 is Madison, n is Seattle.
   $l_{j\to i}$ represents the time require from vertex i to j. $OPT(i,k)$ represents minimum path length from Madison to node i, and visited k nodes.
   We have following recursive equation:

$$OPT(i,k) = \min_{j \text{ is the in-neighbor of } i \text{ and } j<i} OPT(j,k-1) + l_{j\to i} \tag{1}$$

   And the base case will be:

   (a) $OPT(0,0) = 0$
   (b) $OPT(i,0) = \infty$ if $i \neq 0$
   (c) $OPT(i,k) = \infty$ if $i < k$

   And the iterative algorithm just sort the list from according to their index, and then find in the 2D array from bottom to top and left to right. And finally, find the largest k such that $OPT(n,k) \leqslant x$

2. We modify the recursive equation of knapsack problem, the original equation $OPT(i,w)$ defined as the maximum value we could obtain using the items from index 1 to i and the total weight remaining is w. The equation is as follows

$$OPT(i,w) = max \begin{cases} v_i + OPT(i-1, w-w_i) \\ OPT(i-1,w) \end{cases} \tag{2}$$

   We modify this equation by changing the role of w and v. $OPT_2(i,v)$ represents the minimum weight of achieving a subset of items $\{1,2,...,i\}$ of value at least v.

$$OPT_2(i,v) = min \begin{cases} w_i + OPT(i-1, max(0, v-v_i)) \\ OPT(i-1,v) \end{cases} \tag{3}$$

   Then we could make iterative algorithm based on the recursive algorithm. Construct a 2D array from top to bottom and left to right.

3. let $OPT(i)$ represent the maximum sum of contiguous numbers end at index i, define v_i as the number of index i.

$$OPT(i) = max \begin{cases} OPT(i-1) + v_i \\ v_i \end{cases} \tag{4}$$

   base case is $OPT(1) = v_1$ Then use above recursive equation to fill in the 1D array. And compare all cells in 1D array and return the largest.

4. The bus have two choices, go left or go right. And let the students get off when the bus reach a stop.
   The define $OPT(i,j,L)$ be the cost that the bus in on the left hand side closer to i, which means that bus is at stop $i+1$ and left nearest stop for this bus is i and right nearest stop for this bus is j
   Similarly, let $OPT(i,j,R)$ be the cost that the bus stop at $j-1$ and the left nearest bus station is i and right nearest bus station is j.
   Let $d(i,j)$ represents the distance between station i and station j.
   Let $s[i]$ represent the number of students need to get off at stop i.

Thus the bus have two choices: go left or go right. The the following recursive equation follows:

$$OPT(i,j,L) = min \begin{cases} OPT(i-1,j,L) + d(i-1,i) \times (\sum_{k=1}^{i} S[k] + \sum_{k=j}^{n} S[k]) \\ OPT(i,j+1,R) + d(i+1,j) \times ((\sum_{k=1}^{i} S[k] + \sum_{k=j}^{n} S[k]) \end{cases} \tag{5}$$

$$OPT(i,j,R) = min \begin{cases} OPT(i-1,j,L) + d(j-1,i) \times (\sum_{k=1}^{i} S[k] + \sum_{k=j}^{n} S[k]) \\ OPT(i,j+1,R) + d(j-1,j) \times ((\sum_{k=1}^{i} S[k] + \sum_{k=j}^{n} S[k]) \end{cases} \tag{6}$$

Assume the location of school is between stop a and stop b.
The the base case is $OPT(0,n,L)$ which means the bus is at stop 1 and there is one one stop n left, then $OPT(0,n,L) = s[n] \times d(1,n)$. Another base case is $OPT(1,n+1,R) = s[1] \times d(1,n)$.
Assume the school is between stop a and stop b.
The the target instance is:

$$OPT(a,b,L) = OPT(a,b,L) = min \begin{cases} OPT(a-1,b,L) + d(school,a) \times totalstudents \\ OPT(a,b+1,L) + d(school,b) \times totalstudents \end{cases} \tag{7}$$

5. The trick for this problem is we need to track both max and min. Because the product of two negative number could give you max. We have two equations: Define $Max(i,j)$ as max value over index from i to j.
let $o_{i,j}$ indicates the operator between i and j

$$Max(i,j) = max_{i \leqslant k < j} \begin{cases} MAX(i,k)\ o_{k,k+1}\ MAX(k+1,j) \\ MAX(i,k)\ o_{k,k+1}\ MIN(k+1,j) \\ MIN(i,k)\ o_{k,k+1}\ MAX(k+1,j) \\ MIN(i,k)\ o_{k,k+1}\ MIN(k+1,j) \end{cases} \tag{8}$$

Define $Max(i,j)$ as min value over index from i to j.

$$Max(i,j) = min_{i \leqslant k < j} \begin{cases} MAX(i,k)\ o_{k,k+1}\ MAX(k+1,j) \\ MAX(i,k)\ o_{k,k+1}\ MIN(k+1,j) \\ MIN(i,k)\ o_{k,k+1}\ MAX(k+1,j) \\ MIN(i,k)\ o_{k,k+1}\ MIN(k+1,j) \end{cases} \tag{9}$$

Base case:

(a) $MIN(i,i) = MAX(i,i) = A[i]$ for $1 \leqslant i \leqslant n$.

(b) $MIN(i,j) = \infty$ if $i > j$.
$MAX(i,j) = -\infty$ if $i > j$.

6. Let $OPT(i,j)$ be minimum insertion from i to j
Let $A[i]$ represent the letter at index i

$$OPT(i,j) = min \begin{cases} OPT(i+1,j-1) & \text{only if } A[i] = A[j] \\ OPT(i+1,j) + 1 \\ OPT(i,j-1) + 1 \end{cases} \tag{10}$$

(a) $OPT(i,i) = 0$

(b) $OPT(i,i+1) = \begin{cases} 0 & \text{if } A[i] = A[i+1] \\ 1 & \text{Otherwise} \end{cases}$.

(c) $OPT(i, j) = \infty$ if $i > j$

The main instance we want is $OPT(1, n)$
Time complexity is $O(n^2)$.

7. Let $OPT(v)$ be the minimum number of coins we need to achieve a value of v

$$OPT(v) = min_{1 \leqslant i \leqslant n} OPT(v - a_i) \text{ if } v - a_i \geqslant 0 \tag{11}$$

The base case is $O(0) = 0$.

8. First we try to find the property that set of precincts is susceptible.
Define A_i as number of voters vote for part A in precinct i. Let A indicate the total number of voter for party A. Suppose we divide A into two districts, then $A_1 + A_2 = A$. If A holds a majority in both districts, then the following must be true:

(a) $A_1 \geqslant \frac{mn}{4} + 1$

(b) $A - A_1 \geqslant \frac{mn}{4} + 1$

Solve both equation, we have: $\frac{mn}{4} + 1 \leqslant A_1 \leqslant A - \frac{mn}{4} - 1$

(a) Check if we could divide the n precincts into to a set that satisfied: $\frac{mn}{4} + 1 \leqslant A_1 \leqslant A - \frac{mn}{4} - 1$.
Define $OPT(i, k, a)$ as if there is a way that using k precincts from 1 to i and the number of voters vote for party A is a, then this value is true. Otherwise, it is false.

$$OPT(i, k, a) = OR \begin{cases} OPT(i-1, k, a) \\ OPT(i-1, k-1, a - A_i) \end{cases} \tag{12}$$

Base case: $OPT(1, 1, a) = \begin{cases} True & \text{if } a = A_i \\ False & \text{Otherwise} \end{cases}$

(b) Then we could check $OPT(n, \frac{n}{2}, a)$ for $\frac{mn}{4} + 1 \leqslant a \leqslant A - \frac{mn}{4} - 1$. If any value is true, then it is susceptible. Otherwise, it is not.

The complexity is $O(An^2)$.