HW5 Solutions

#1 Solution

We slightly modify the original TSP algorithm such that the tours whose total weight exceeds W will be discarded by the recursion. We set up the recursion as follows, where $TSP(v_i, V', weight)$ is similar to the original except that it returns the minimum distance of the tours with total weight at most $weight$. $d(v_i, v_j)$ denotes the distance between v_i, v_j, and $w(v_i, v_j)$ denotes the weight of the path from v_i to v_j.

$$TSP(v_i, V', weight)$$
$$= \begin{cases} \min_{v \in V'} \left\{ d(v_i, v) + TSP(v, V' \backslash v, weight - w(v_i, v)) \right\} & V' \neq \emptyset, weight \geq 0 \\ d(v_i, v_1) & V' = \emptyset \\ +\infty & weight < 0 \end{cases}$$

Then, the algorithm looks like (memoization omitted)

DP-WEIGHTED-TSP

1. $result \leftarrow +\infty$
2. For all $v \in V$, do
   (a) $result \leftarrow \min(result, TSP(v, V, W))$
3. return $result$

In order to get the sequence of vertices in the tour, we also store each v that is removed from V' (because it leads to the minimum value) **in order**, and always keep the ordered vertices of the minimum path in the recursion. As analyzed in the lecture note, the time complexity is $O(n2^n)$

#2 Solution

We can find the recursion as follows, where $CHNG(x) = T$ if it is possible to make change for bill of x and F otherwise.

$$CHNG(x) = \begin{cases} \vee_{i=1}^{n} CHNG(x - d_i) & x > 0 \\ T & x = 0 \\ F & x < 0 \end{cases}$$

Then, a simple call to $CHNG(v)$ (with memoization) will return the result. If it returns T, it is possible to make changes, and it cannot make the changes if it returns F. If one wishes to know the exact selections of denominations, he can keep storing the selections every time $CHNG(x)$ returns T, and discard all corresponding historical selections at the time the product evaluates as F.

Essentially, the memoization table T has v entries to fill up, and the final answer is at $T[v]$. To reach $T[v]$, all entries need to be filled up in the worst case, and each entry involves evaluation of $\vee_{i=1}^{n}$ which is $O(n)$. Then, the final complexity is $O(nv)$.

#3 Solution

We get the following recurrence: (The calories it gains by eating the cheese at i,j and the maximum calories it can get from moving to either the cell on the right or the cell below)

$$W(i,j) = \max\{W(i+1,j) + c(w(i+1,j)), W(i,j+1) + c(w(i,j+1))\}, i,j \leq n$$

$$W(i,j) = 0, i > n \text{ or } j > n$$

where we define $C(w(i,j)) = 0$ if $i > n$ or $j > n$.

Start with W(1,1)+c(w(1,1) to obtain the solution.

Memoing uses a two dimensional array. resulting in a time complexity of $O(n^2)$.

Algorithm 1 CheeseEater

for $i = n - 1$ to 1 **do**
    $W(i, n) = W(i + 1, n) + c(w(i, n))$
end for
for $j = n - 1$ to 1 **do**
    $W(n, j) = W(n, j + 1) + c(w(n, j))$
end for
for $i = n - 1$ to 1 **do**
    for $j = n - 1$ to 1 **do**
        $W(i, j) = \max\{W(i + 1, j) + c(w(i + 1, j)), W(i, j + 1) + c(w(i, j + 1))\}, i, j \leq n$
    end for
end for