CS 325 – HW3Sol

1. Canoe Rental Problem (10 pts total) : *There are n trading posts numbered 1 to n as you travel downstream. At any trading post i you can rent a canoe to be returned at any of the downstream trading posts j, where j ≥ i. You are given an array R[i, j] defining the costs of a canoe which is picked up at post i and dropped off at post j, for 1 ≤ i ≤ j ≤ n. Assume that R[i,i] = 0 and that you can't take a canoe upriver. Your problem is to determine a sequence of rentals which start at post 1 and end at post n, and that has the minimum total cost.*

a) Describe verbally and give pseudo code for a DP algorithm to compute the cost of the cheapest sequence of canoe rentals from trading post 1 to n. Give the recursive formula you used to fill in the table *or array.* **(5 pts)**

Define a 1 dimensional table *C*[1…*n*] where $C[i]$ is the cost of an optimal sequence of canoe rentals that starts at post 1 and ends at post *i*, for $1 \le i \le n$. When this table is filled, we simply return the value $C[n]$. *Note could use a different variable name for C[n].*

Clearly $C[1] = 0$.

Define $C[i]$ in terms of earlier table entries. Indeed its clear that $C[i] = C[k] + R[k,i]$. Since we do not know the post *k* beforehand, we take the minimum of this expression over all *k* in the range $1 \le k < i$. Define

$$C[i] = \begin{cases} 0 & i = 1 \\ \min\limits_{1 \le k < i}\left(C[k] + R[k,i]\right) & 1 < i \le n \end{cases}$$ **(2 pts)**

With this formula, the algorithm for filling in the table is straightforward.

<u>Below is a strictly bottom-up approach. Give full credit for memorized DP.</u>

<u>CanoeCost(*R*)</u> **Algorithm (3 pts)**

```
n ← #rows[R]
C[1] ← 0
for i ← 2 to n
    min ← R[1,i]
    for k ← 2 to i−1
        if C[k] + R[k,i] < min
            min ← C[k] + R[k,i]
    C[i] ← min
return C[n]
```

b) Print Sequence **(3pts)**

Explanation **(1 pt)**
To determining the actual sequence of canoe rentals which minimizes cost alter the CanoeCost() algorithm so as to construct the optimal sequence while the table *C*[1…*n*] is being filled. In the following algorithm we maintain an array *P*[1…*n*] where $P[i]$ is defined to be the post *k* at which

the last canoe is rented in an optimal sequence from 1 to *i*. Note that the definition of $P[1]$ can be arbitrary since it is never used. Array *P* is then used to recursively print out the sequence.

CanoeSequence(*R*) **Modifications of original algorithm (1 pt)**

```
n ← #rows[R]
C[1] ← 0 , P[1] ← 0
for i ← 2 to n
    min ← R[1,i]
    P[i] ← 1
    for k ← 2 to i−1
        if C[k] + R[k,i] < min
            min ← C[k] + R[k,i]
            P[i] ← k
    C[i] ← min
return P
```

PrintSequence(*P*, *i*) (Pre: $1 \le i \le \text{length}[P]$) **(1 pt)** *Note can use an iterative method*

```
if i > 1
    PrintSequence(P, P[i])
    print "Rent a canoe at post " P[i] " and drop it off at post " i
```

c) What is the running time of your algorithm to find the minimum cost and to find the sequence?

(2 pts)

CanoeCost() runs in time $\Theta(n^2)$, since the inner for loop performs $i-2$ comparisons in order to determine $C[i]$, and $\sum_{i=2}^{n}(i-2) = \frac{(n-1)(n-2)}{2} = \Theta(n^2)$.

The top level call to PrintSequence(*P*, *n*) has a cost that is the depth of the recursion, which is in turn, the number of canoes rented in the optimal sequence from post 1 to post *n*. Thus in worst case, PrintSequence() runs in time $\Theta(n)$.