Family Vacation

- Set of cities denoted by $P_1, P_2, \ldots, P_n$.
- $d_i$: Distance from $P_{i-1}$ to $P_i$ ($1 < i \leq n$).
- $d_1 = 0$
- $c_i$: Cost of dinner, lodging and breakfast when staying at city $P_i$ ($1 < i < n$).
- $c_1 = c_n = 0$.
- $w$: maximum number of miles the family may drive each day.
- Assume that for each $i$, $d_i \leq w$.

Problem: Find the cities the family needs to stay overnight when driving from city $P_1$ to $P_n$ (using the route $P_1, P_2, \ldots, P_n$) such that every day the family drives at most $w$ miles and the total cost of the overnight stays is least possible.
Greedy Method 1

Drive as much as possible each day.

Method does not always generate an optimal solution. Counterexample ($w = 300$).

<table>
<thead>
<tr>
<th>Cities</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>0</td>
<td>100</td>
<td>200</td>
<td>100</td>
<td>200</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>$c_i$</td>
<td>0</td>
<td>20</td>
<td>500</td>
<td>20</td>
<td>500</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

- Greedy solution: Stay at $P_3$ and $P_5$. Total Cost is $500 + 500 = 1000$.
- Optimal Solution: Stay at $P_2$, $P_4$ and $P_6$. Total cost is $20 + 20 + 20 = 60$.
- Therefore this greedy method does not generate an optimal solution.
- However this greedy method minimizes the number of overnight stays.
Greedy Method 2

Stay overnight at the least expensive place within \( w \) miles from the previous overnight stay. (Ties?) Stay at the city with least cost that is farthest from the previous stay.

Method does not always generate an optimal solution. Couterexample (\( w = 50 \))

<table>
<thead>
<tr>
<th>Cities</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
<th>( P_5 )</th>
<th>( P_6 )</th>
<th>( P_7 )</th>
<th>( P_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_i )</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>( c_i )</td>
<td>0</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>0</td>
</tr>
</tbody>
</table>

- Greedy solution: Stay at \( P_2, P_3, P_4, P_5, P_6 \), and \( P_7 \). Total cost is \( 50 + 51 + 52 + 53 + 54 + 55 = 315 \).

- Better Solution: Stay at \( P_6 \) and \( P_7 \). Cost = 109.

- This greedy method is not optimal.
Greedy Method 3

Stay overnight at the city (at a distance at most $w$ from previous stay) where the cost of the overnight stay divided by the miles driven is least possible. (Ties?) Stay at the city with least cost that is farthest from the previous stay.

Method does not always generate an optimal solution. Counterexample ($w = 50$)

<table>
<thead>
<tr>
<th>Cities</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
<th>$P_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>50</td>
</tr>
<tr>
<td>$c_i$</td>
<td>0</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>0</td>
</tr>
</tbody>
</table>

- $$/dist: 50/10=5, 51/20=2.55, 52/30=1.73, 53/40=1.32, 54/50=1.08.$
• This greedy method is not optimal.
Dynamic Programming Approach

- $M_i$: Minimum cost for the overnight stays when driving from $P_i$ to $P_n$ with the restriction that the family must drive at most $w$ miles. This assumes that the family starts at $P_i$.
- Objective is to compute $M_1$ and then figure out the places where the family stays overnight in an optimal solution with cost $M_i$.
- Clearly, $M_n = 0$. 
Recurrence Relation

\[ M_i = \min \begin{cases} 
    c_{i+1} + M_{i+1} & \text{if } d_{i+1} \leq w \\
    c_{i+2} + M_{i+2} & \text{if } d_{i+1} + d_{i+2} \leq w \\
    \vdots & \vdots \\
    c_j + M_j & \text{if } \sum_{k=i+1}^{j} d_k \leq w 
\end{cases} \]

where either \( j = n \) or \( \sum_{k=i+1}^{j+1} d_k > w \)

Note that the above equations means that one would drive and stay at \( P_{i+1} \), or \( P_{i+2} \), or \( \ldots \), or \( P_j \).
Example

<table>
<thead>
<tr>
<th>Cities</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
<th>$P_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i$</td>
<td>0</td>
<td>150</td>
<td>100</td>
<td>100</td>
<td>70</td>
<td>100</td>
<td>150</td>
<td>90</td>
</tr>
<tr>
<td>$c_i$</td>
<td>0</td>
<td>40</td>
<td>100</td>
<td>60</td>
<td>120</td>
<td>30</td>
<td>70</td>
<td>0</td>
</tr>
</tbody>
</table>

$w$ is equal to 300.

By definition $M_8 = 0$

\[
M_7 = \min \left\{ c_8 + M_8 \ (90 \leq w = 300) \right\}
\]

\[
M_7 = 0 \leftarrow \min \left\{ 0 + 0 \right\}
\]

\[
M_6 = \min \left\{ \begin{array}{l}
  c_7 + M_7 \ (150 \leq w = 300) \\
  c_8 + M_8 \ (150 + 90 \leq w = 300)
\end{array} \right\}
\]
\[ M_6 = 0 \leftarrow \min \left\{ \begin{array}{l} 70 + 0 \\ 0 + 0 \end{array} \right\} \]
\[ M_5 = \min \left\{ \begin{array}{l} c_6 + M_6 \quad (100 \leq w = 300) \\ c_7 + M_7 \quad (100 + 150 \leq w = 300) \end{array} \right. \]

\[ M_5 = 30 \left\{ \begin{array}{l} 30 + 0 \\ 70 + 0 \end{array} \right. \]

\[ M_4 = \min \left\{ \begin{array}{l} c_5 + M_5 \quad (70 \leq w = 300) \\ c_6 + M_6 \quad (70 + 100 \leq w = 300) \end{array} \right. \]

\[ M_4 = 30 \left\{ \begin{array}{l} 120 + 30 \\ 30 + 0 \end{array} \right. \]
\[ M_3 = \min \begin{cases} 
  c_4 + M_4 & (100 \leq w = 300) \\
  c_5 + M_5 & (100 + 70 \leq w = 300) \\
  c_6 + M_6 & (100 + 70 + 100 \leq w = 300) 
\end{cases} \]

\[ M_3 = 30 \leftarrow \min \begin{cases} 
  60 + 30 \\
  120 + 30 \\
  30 + 0 
\end{cases} \]

\[ M_2 = \min \begin{cases} 
  c_3 + M_3 & (100 \leq w = 300) \\
  c_4 + M_4 & (100 + 100 \leq w = 300) \\
  c_5 + M_5 & (100 + 100 + 70 \leq w = 300) 
\end{cases} \]

\[ M_2 = 90 \leftarrow \min \begin{cases} 
  100 + 30 \\
  60 + 30 \\
  120 + 30 
\end{cases} \]
\[ M_1 = \min \begin{cases} 
  c_2 + M_2 & (150 \leq w = 300) \\
  c_3 + M_3 & (150 + 100 \leq w = 300) 
\end{cases} \]

\[ M_1 = 130 \leftarrow \min \begin{cases} 
  40 + 90 \\
  100 + 30 
\end{cases} \]

Optimal Solutions: Stay Overnight at:

- \( P_1, P_2, P_4, P_6, P_8 \). Cost is 40 + 60 + 30 = 130.
- \( P_1, P_3, P_6, P_8 \). Cost is 100 + 30 = 130.
Exponential Time Algorithm

Main program invokes M(1). Note that $c_j$ is written as $c[j]$ and $d_j$ as $d[j]$ in the following procedures.

```
int Procedure M(i) //Exponential time algorithm
    if i == n return 0
    cost = +infinity
    dist = 0
    for j=i+1 to n do
        dist = dist + d[j]
        if dist <= w
            then cost = min{cost, M(j)+c[j]}
            else exit j-loop
    endfor
    return cost
end procedure
```
Polynomial Time Algorithm

Main program invokes M(1). Note that $c_j$ is written as $c[j]$ and $d_j$ as $d[j]$ in the following procedures. Avoid recomputation of $M$ values.

define $M[i] = +\text{infinity}$ \hspace{1cm} 1 <= i < n

define $M[n] = 0$

```c
int Procedure M(i) //Polynomial time algorithm
    if $M[i]$ is finite then return $M[i]$
    dist = 0
    for j=i+1 to n do
        dist = dist + d[j]
        if dist <= w
            then $M[i] = \min\{M[i], M(j)+c[j]\}$
        else exit j-loop
    endfor
    return $M[i]$
end procedure
```

UCSB TG
Iterative Algorithm

Main program invokes M(1). Compute the $M$ values iteratively.

```c
int Procedure Iterative-Compute
    define M[n] = 0
    for i=n-1 to 1 by -1 do
        M[i] = +infinity
        dist = 0
        for j=i+1 to n do
            dist = dist + d[j]
            if dist <= w
                then M[i] = min{M[i], M[j]+c[j]}
            else exit j-loop
        endfor
    endfor
end procedure
```

The time complexity of the inside loop is $O(n)$. Since it is executed $n - 1$ times, then the time complexity $O(n^2)$. 
Printing the Overnight Locations

Main program invokes M(1). Note that \( c_j \) is written as \( c[j] \) and \( d_j \) as \( d[j] \) in the following procedure. Compute the \( M \) values iteratively.

```
int Procedure Iterative-Compute-Print
    Invoke the iterative algorithm.
    Assume we have the M[i] values.

    i = 1
    While i <> n do
        for j=i+1 to n do
            if M[i] == M[j]+c[j]
                then print j
                    i = j
                    exit j-loop
        endfor
    endwhile
end procedure
```

The time complexity for the print part is \( O(n^2) \).
Without Recomputation

Main program invokes \( M(1) \). Compute the \( M \) and \( kay \) values iteratively. \( kay[i] \) stores the next city for the overnight stay in an optimal solution for \( M_i \).

```c
int Procedure Iterative-Compute-Print
    define M[n] = 0
    for i=n-1 to 1 by -1 do
        M[i] = +infinity
        dist = 0
        for j=i+1 to n do
            dist = dist + d[j]
            if dist <= w
                then if M[j]+c[j] < M[i]
                    then kay[i] = j
                    M[i] = min{M[i], M[j]+c[j]}
                else exit j-loop
        endfor
    endfor
endfor
```
/* Printing Part */

i = 1
While i <> n do
    print kay[i]
    i = kay[i]
endwhile
end procedure

The time complexity for the print part is $O(n)$
and the computation of the $M[i]$ and $kay[i]$ values
is $O(n^2)$. 