CS3230
Tutorial 8

1. Give an argument to show why dynamic programming algorithm gives optimal solution for coin changing problem.

   Ans: By induction on k, we show that $C[n-k,j]$ is optimal (that is, it gives the optimal number of coins using denominations $d[i]$, $i \geq n-k$).

   Clearly, for $k=0$, $C[n,j]$ is optimal, for $j \leq S$.

   Suppose $C[n-k+1,j]$ is optimal, for $j \leq S$.

   We then show that $C[n-k,j]$ is optimal for $j \leq S$, by induction on j.

   Clearly, this holds if $j=0$.

   Suppose $C[n-k,j]$ is optimal for $j \leq m$.

   Then, consider $C[n-k,m+1]$. Consider two cases, when optimal change contains $d[n-k]$ or not. In the first case, the optimal change contains $1+C[n-k,m+1-d[n-k]]$ coins (by induction). In the latter case, the optimal change contains $C[n-k+1,m+1]$ coins (by induction).

   As the algorithm chooses the minimal of these two cases, we have that $C[n-k,m+1]$ as the correct optimal answer.

2. Build a table to show how the dynamic programming algorithm will work for finding the optimal algorithm for following matrix multiplication.

   $M_1 \times M_2 \times M_3 \times M_4 \times M_5$, where

   M_1 is a matrix of size 6×6

   M_2 is a matrix of size 6×3

   M_3 is a matrix of size 3×4

   M_4 is a matrix of size 4×4

   M_5 is a matrix of size 4×8

   $F(1,1)=0$, $F(2,2)=0$, $F(3,3)=0$, $F(4,4)=0$, $F(5,5)=0$.

   $F(1,2)=6*6*3=108$, $F(2,3)=72$, $F(3,4)=48$, $F(4,5)=128$.

   $F(1,3)=min(0+72+144,108+0+72)=180$, $F(2,4)=min(0+48+72,72+0+96)=120$, $F(3,5)=min(0+128+96,48+0+96)=144$.

   $F(1,4)=min(0+120+144,108+48+72,180+0+96)=228$, $F(2,5)=min(0+144+144,72+128+192,120+0+192)=288$.

   $F(1,5)=min(0+288+288,108+144+144,180+128+192,228+0+192)=396$.

3. Give a counterexample to show that the following conjecture is false:

   To minimize the number of scalar multiplications of the product $M_1 \times M_2 \times \ldots \times M_n$, it should be grouped as

$$(M_1 \times M_2 \times \ldots M_k) \times (M_{k+1} \times \ldots \times M_n),$$

   where M_k has minimum number of columns.

   Ans: Consider $M_1 \times M_2 \times M_3$, where size of M_1 is (1×1), M_2 is (1×2) and M_3 is (2×100). Then,

   Optimal answer is by doing $(M_1 \times M_2) \times M_3$ which gives number of multiplications as $1 \times 1 \times 2 + 1 \times 2 \times 100$, which is 202. On the other hand, the method suggested by the question gives

   $\mathbf{M}_1 \times (M_2 \times M_3)$ which uses $(1 \times 2 \times 100) + (1 \times 2 \times 100)$, which is 400.

4. Modify the algorithm given in the class to show how the order of matrix multiplication can be obtained along with the optimal number of multiplications needed.

   Ans:

```
1. For $i = 1$ to $n$ { $F(i,i) = 0$ } (* Initialization *)
2. For $r = 1$ to $n - 1$ {
3.  For $i = 1$ to $n - r$ {
4.  $j = i + r$.
    5.   $F(i,j) = min_{k=i}^{j-1}[F(i,k) + F(k+1,j) + r_i c_k c_j]$.
    6.   Let $C(i,j) = k$, where $k$ is the value which gives the minimal in the above
           computation.
    }
}
```

   MatOrder is initially called with $i = 1$ and $j = n$. Below $H_{r,p}$ denotes the matrix obtained by $M_r \times M_{r+1} \times \ldots \times M_p$.

```
MatOrder(i,j)
    If $i = j$, then return.
    Otherwise,
          MatOrder($i, C(i,j)$),
          MatOrder($C(i,j) + 1, j$)
          Print("Muliply $H_{i,C(i,j)}$ and $H_{C(i,j)+1,j}$")
End
```

5. Give a dynamic programming algorithm to compute a^n given the following formula:

   $a^0 = 1$

   $a^n = a^{n/2} * a^{n/2}$ if n is even.

   $a^{n+1} = a^{n/2} * a^{n/2} * a$ if n is odd.

   Ans:

   The equations are:

   $F(0) = 1$.

   $F(n) = F(n/2) * F(n/2)$, if n is even.

   $F(n) = F((n-1)/2) * F((n-1)/2) * a$, if n is odd.

   The algorithm is:

```
$F(0) = 1$
For $i = 0$ to $\lfloor (n-1)/2 \rfloor$ {
    $F(2i+1) = F(i) * F(i) * a$
    $F(2i+2) = F(i+1) * F(i+1)$.
}
```