Chain Matrix Multiplication

- Suppose you are to multiply 4 matrices, A, B, C, D of dimensions 40×20, 20×400, 400×10, 10×100.

- Multiplying a $m \times n$ matrix and a $n \times p$ matrix requires mnp multiplications.

- $(((A \times B) \times C) \times D)$ needs $40.20.400 + 40.400.10 + 40.10.100 = 520,000$ multiplications

- $((A \times ((B \times C) \times D))$ needs $20.400.10 + 20.10.100 + 40.20.100 = 180,000$ multiplications

- $((A \times (B \times C)) \times D)$ needs $20.400.10 + 40.20.10 + 40.10.100 = 128,000$ multiplications

How to find the best sequence of multiplications? Examining all possible sequence will take O(2^n). In fact there are $C(n)$ of such combinations, where $C(n)$ called the *Catalan* number

Use Dynamic Programming!

Suppose that $M = m_1 m_2 \ldots m_q$
where m_i is a $r_i \times r_{i+1}$ matrix, $1 \leq i \leq q$
Let $M_{ij} = m_i \times \ldots \times m_j$, where $i \leq j$
Let c_{ij} be the cost of the optimal way to compute M_{ij}
Let d_{ij} be the position, k, of the optimal computation of M_{ij} where $M_{ij} = M_{ik} \times M_{k+1j}$

What is the recurrence relation?

$C(i,i) = 0, \quad 1 \leq i \leq q$
$C(i,i+1) = r_i.r_{i+1}.r_{i+2}, \quad d_{ij} = i, \quad 1 \leq i \leq q$

$C(i,i+s) =$
$\min_{i \leq k < i+s}\{C(i,k) + C(k+1,i+s) + r_i.r_{k+1}.r_{i+s+1}\}$

where $1 \leq i \leq q-s, \;\; 1 < s < q, \quad d_{ij} = k$ which results in the minimum $C(i,i+s)$

An Example

$q = 4,\ (r_1, r_2, r_3, r_4) = (10, 5, 1, 10, 2)$

$C(1,4) = \min\{C(1,1) + C(2,4) + 100, C(1,2) + C(3,4) + 20, C(1,3) + C(4,4) + 200\}$

$C(1,1) = 0, C(1,2) = 50, C(3,4) = 20, C(4,4) = 0$

$C(2,4) = \min\{C(2,2) + C(3,4) + 10, C(2,3) + C(4,4) + 100\}$

$C(2,4) = \min\{0 + 20 + 10, 50 + 0 + 100\} = 30$

$C(1,3) = \min\{C(1,1) + C(2,3) + 500, C(1,2) + C(3,3) + 100\}$

$C(1,3) = \min\{0 + 50 + 500, 50 + 0 + 100\} = 150$

$C(1,4) = \min\{C(1,1) + C(2,4) + 100, C(1,2) + C(3,4) + 20, C(1,3) + C(4,4) + 200\}$

$C(1,4) = \min\{0 + 30 + 100, 50 + 20 + 20, 150 + 0 + 200\} = 90$

Solution: Table Method

```
static int[][] c = new int[100][100];
static int[][] d = new int[100][100];
static int[]   r = new int[101];

static int optimal(int q){
  for (int i=1; i<=q; i++){
      c[i][i]=0;
      c[i][i+1]= r[i]*r[i+1]*r[i+2];
  }
  for (int s=2; s<q ; s++)
    for (int i=1; i<=q-s; i++){
       int mymin=0x7FFFFFFF;
       for (int k=i; k<i+s; k++){
          int temp=c[i][k]+c[k+1][i+s]+
                   r[i]*r[k+1]*r[i+s+1];
          if (temp<mymin){
             mymin=temp; d[i][i+s]=k;
          }
       }
       c[i][i+s]=mymin;
   }
  return c[1][q];
}
```

Solution: Memoization Method

```
static int[][] c = new int[100][100];
static int[][] d = new int[100][100];
static int[]   r = new int[101];

static int C(int i, int j){
  if (c[i][j]>0) //already computed
     return c[i][j];
  //not computed, compute now!
  if (i==j) return 0; //one matrix
  if (i==j-1){ //two matrices
     c[i][j]=r[i]*r[i+1]*r[i+2];
     d[i][j]=i;
     return c[i][j];
  }
  //more than two matrices
  int u=C(i,i)+C(i+1,j)+r[i]*r[i+1]*r[j+1];
  d[i][j]=i;
  //compute the remaining min terms
  for (int k=i+1; k<j; k++){
      int t=C(i,k)+C(k+1,j)+r[i]*r[k+1]*r[j+1];
      if (t<u){//smaller term
         u=t; d[i][j]=k;
      }
  }
  c[i][j]=u;
  return u;
}
```