```
//**************************************
//        Matrix Chain Order
//**************************************
Matrix-Chain-Order(p)
   n = length[p] - 1  // n is the number of matrix multiplies needed
   // Initialize results array
   for i = 1 to n
      do m[i,i] = 0
   for l = 2 to n  // l is the diagonal to fill in
      do for i = 1 to n - l + 1   // work down the diagonal; i is the row;
           do j = i + l - 1     // column j depends on diagonal and row
              m[i,j] = INFINITY  // initialize result so we can find min
              for k = i to j - 1 // check all possible split points k
                 do q = m[i,k] + m[k+1,j] + p[i-1]*p[k]*p[j]
                    if q < m[i,j]
                       then m[i,j] = q
                            s[i,j] = k  // save split point so
                                        // multiplication order can be
                                        // reconstructed
   return m and s
```

```
Memoized-Matrix-Chain(p)
    // Initialize results array and call auxilliary recursive function
    n = length[p] - 1  // n is the number of matrix multiplies needed
    for i = 1 to n
              do for j = i to n
               do m[i,i] = INFINITY
   return Lookup-Chain(p, 1, n)

Lookup-Chain(p, i, j)
    if m[i,j] < INFINITY
       then return m[i,j]
    if i == j
       then m[i,j] = 0
       else for k = i to j - 1 // check all possible split points k
                do q = Lookup-Chain(p,i,k) + Lookup-Chain(k+1,j)
                              + p[i-1]*p[k]*p[j]
                       if q < m[i,j]
                          then m[i,j] = q
    return m[i,j]
```