Problem 1.

a) Let X be the value of first expression, the state space $X = \{0, 4, 5, 6, 7, 8, 9, \cdots, \infty\}$, the action space $U = \{0, 1, 2, \cdots, \infty\}$.

The dynamic programming equation is

$$v_t(x) = \max_{u \in U_t(x)} \{c_t u + v_{t+1}(x_{t+1})\};\ x \in X; t = n, n-1, \ldots, 1;$$

$$v_{n+1}(x) = 0, \qquad x \in X$$

b) $z_j = [3,0,0,0,1]$, and the result of this problem is 18.

In [32]:

```
def z():
    zj=[0,0,0,0,0]
    z=[0,0,0,0,0]
    sum0 = 0
    while zj[0] <=4:
        zj[1] =0
        while zj[1]<=4:
            zj[2] =0
            while zj[2]<=3:
                zj[3] =0
                while zj[3]<=2:
                    zj[4] =0
                    while zj[4]<=2:
                        sum1 = 4*zj[0]+6*zj[1]+5*zj[2]+9*zj[3]+7
*zj[4]
                        sum2 = 4*zj[0]+5*zj[1]+4*zj[2]+8*zj[3]+6
*zj[4]
                        if sum1 < 20:
                            if sum2>sum0:
                                z = zj
                                sum0=sum2
                                print(z)
                                print(sum2)
                        zj[4] =zj[4]+1
                    zj[3]=zj[3]+1
                zj[2]=zj[2]+1
            zj[1]=zj[1]+1
        zj[0]=zj[0]+1

z = z()
```

```
[0, 0, 0, 0, 1]
6
[0, 0, 0, 0, 2]
12
[0, 0, 0, 1, 1]
14
[0, 0, 0, 2, 0]
16
[1, 1, 0, 1, 0]
17
[3, 0, 0, 0, 1]
18
```

Problem 2.

a) Let $A[i : j]$ be the product $A_i A_{i+1} \cdots A_j$. The problem is to choose the order of multiplying $A[1 : n]$ that will minimize the number of multiplications. Suppose this order divide the matrixes between matrix A_p and A_{p+1}, $1 \leq p \leq n$. Thus, this problem is decomposed into two sub-problems. Thus, the number of multiplications of $A[1 : n]$ is equal to the number of multiplications of $A[1 : p]$ plus $A[p + 1 : n]$, and the number of multiplications computing the product $A[1 : p]A[p + 1 : n]$. Hence, we have the following recurrence for the number of multiplications to parenthesize the matrix chain of n matrices. For $1 \leq i \leq j \leq n$, let $m[i, j]$ denote the minimum number of multiplications of $A[1 : n]$.

$$m[i,j] = \begin{cases} 0 & , i = j \\ min_{i \leq p \leq j}\left(m[i,p] + m[p+1,j] + k_{i-1}k_p k_j\right), & i < j \end{cases}$$

b) The order is $((A_1(A_2A_3))A_4)$, and the minimum number of multiplications needed to compute the product is 6800. The printout is on the following page.

In [5]:

```
import random
from pandas import *

matrix = [[10, 30], [30, 70], [70, 2], [2, 100]]
m = [[0] * 4 for i in range(4)]
s = [[0] * 4 for j in range(4)]

def MatrixMultiplication(inp):
    for i in range(inp):
        m[i][i] = 0
    for r in range(1, inp):
        for i in range(inp-r):
            j = i + r
            m[i][j] = m[i+1][j] + matrix[i][0] * matrix[i][1] *
matrix[j][1]
            s[i][j] = i+1
            for k in range(i+1, j):
                judge = m[i][k] + m[k+1][j] + matrix[i][0] * mat
rix[k][1] * matrix[j][1]
                if judge < m[i][j]:
                    m[i][j] = judge
                    s[i][j] = k+1

def printmatrix(left, right):
    if left == right:
        print("A"+str(left+1), end='')
    else:
        print("(", end='')
        printmatrix(left, s[left][right]-1)
        printmatrix(s[left][right], right)
        print(")", end='')

MatrixMultiplication(4)
dm = DataFrame(m, index=list(range(1, 5)), columns=list(range(1,
5)))
ds = DataFrame(s, index=list(range(1, 5)), columns=list(range(1,
5)))
print('Matrix:',matrix)
print("The number of multiplications: \n", dm)
printmatrix(0, 3)
```

```
Matrix: [[10, 30], [30, 70], [70, 2], [2, 100]]
The number of multiplications:
    1      2     3      4
1  0  21000  4800   6800
2  0      0  4200  10200
3  0      0     0  14000
4  0      0     0      0
((A1(A2A3))A4)
```

Problem 3.

The shortest path from node 1 to node 10 is 1-4-5-10.

In [2]:

```
def getPath(i, j):
    if i != j:
        if path[i][j] == -1:
            print('-', j+1, end='')
        else:
            getPath(i, path[i][j])
            getPath(path[i][j], j)


def printPath(i, j):
    print('Path:', i+1, end='')
    getPath(i, j)
    print()

# initialized
vertex=10
edge = 20
inf = 99999999
dis = []  # matrix of the shortest distance
path = []  # record the shortest path
for i in range(vertex):
    dis += [[]]
    for j in range(vertex):
        if i == j:
            dis[i].append(0)
        else:
            dis[i].append(inf)
for i in range(vertex):
    path += [[]]
    for j in range(vertex):
        path[i].append(-1)
table = [[1,2,4],[1,4,7],[1,6,8],[1,8,9],[2,4,7],[4,6,12],[8,6,6
],[2,3,11],
        [4,3,5],[4,5,10],[6,5,16],[6,7,15],[7,8,11],[8,9,12],[3,
5,10],
        [9,7,9],[3,10,16],[5,10,8],[7,10,4],[9,10,14]]

# weight matrix
for i in range(edge):
    u, v, w = table[i][0],table[i][1],table[i][2]
    u, v, w = int(u)-1, int(v)-1, int(w)
```

```python
    dis[u][v] = w

print('the weight matrix is:')
for i in range(vertex):
    for j in range(vertex):
        if dis[i][j] != inf:
            print('%5d' % dis[i][j], end='')
        else:
            print('%5s' % '∞', end='')
    print()


# floyd algorithm
for k in range(vertex):
    for i in range(vertex):
        for j in range(vertex):
            if dis[i][j] > dis[i][k] + dis[k][j]:
                dis[i][j] = dis[i][k] + dis[k][j]
                path[i][j] = k
print('===========================================')
print('v%d ----> v%d  tol_weight:''%3d' % (1, 10, dis[0][9]))
printPath(0, 9)
```

```
the weight matrix is:
    0    4    ∞    7    ∞    8    ∞    9    ∞    ∞
    ∞    0   11    7    ∞    ∞    ∞    ∞    ∞    ∞
    ∞    ∞    0    ∞   10    ∞    ∞    ∞    ∞   16
    ∞    ∞    5    0   10   12    ∞    ∞    ∞    ∞
    ∞    ∞    ∞    ∞    0    ∞    ∞    ∞    ∞    8
    ∞    ∞    ∞    ∞   16    0   15    ∞    ∞    ∞
    ∞    ∞    ∞    ∞    ∞    ∞    0   11    ∞    4
    ∞    ∞    ∞    ∞    ∞    6    ∞    0   12    ∞
    ∞    ∞    ∞    ∞    ∞    ∞    9    ∞    0   14
    ∞    ∞    ∞    ∞    ∞    ∞    ∞    ∞    ∞    0
===========================================
v1 ----> v10  tol_weight: 25
Path: 1- 4- 5- 10
```