```
#-------------------------------------------------------------------------
#                    Exemplos de uso da Programação Dinâmica
#                         (programas em python)
#-------------------------------------------------------------------------


#-------------------------------------------------------------------------
#                        Produto óptimo de matrizes
#-------------------------------------------------------------------------



#   Mi......Mj
#   Mk tem dimensoes d[k]*d[k+1]
#   c[i][j] = min_(i<=p<j ) {c[i][p] + c[p+1][j] +
#                            d[i]d[p+1]d[j+1]
#                           }

# d e' a lista das dimensoes
# para M1[10,20]*M1[20,5]*M1[5,50], d=[10,20,5,50]

def matmin(d):
    infty = 1E10
    m=len(d)-1    # num de matrizes
    c     = [[0]*m  for i in range(m)]    # criar matriz m*m
    inter = [[-1]*m for i in range(m)]    # criar matriz m*m, não definido
    for k in range(2,m+1):
        for i in range(m-k+1):
            j=i+k-1
            cmin=infty
            for p in range(i,j):          # para cada possível produto de topo:
                custo=c[i][p] + c[p+1][j] + d[i]*d[p+1]*d[j+1]
                if custo<cmin:
                    cmin=custo
                    inter[i][j]=p
            c[i][j]=cmin
    return (c[0][m-1],inter)



#-------------------------------------------------
# imprime o produto óptimo

def expr(inter,a,b):
    m=inter[a][b]
    #print "a b mrd",a,b,m
    if a==b:
        printm(a),
        return
    #print "inter",m
    #print "*** a,b,m",a,b,m
    if m<0:
        return
    print "(",
    expr(inter,a,m)
    print "*",
    expr(inter,m+1,b)
    print ")",

def printm(a):
    print "M"+str(a),

def expressao(d):
    (_,inter)=matmin(d)
    expr(inter,0,len(d)-2)
```

```
def expressao1(d):
    (_,inter)=topdown(d)
    expr(inter,0,len(d)-2)

def next(n,t):
    return t*2*(2*n-1)/(n+1.0)

#--------------------------- testes ----------------------------
for d in [[2,3],[3,4,5],[100,20,100,20],[100,100,10,2],[10,50,1,42,8],[10,100,50,1,42,8]]:
    print
    print "----------------------------"
    print "d = ",d
    print matmin(d)[0]
    expressao(d)
    print
```

```
Produto:        (2x3)
d:              [2,3]
Custo mínimo    0
Produto óptimo: M0
----------------------------
Produto:        (3x4)(4x5)
d:              [3,4,5]
Custo mínimo    60
Produto óptimo: (M0*M1)
----------------------------
Produto:        (100x29)(20x100)(100x20)
d:              [100,20,100,20]
Custo mínimo    80000
Produto óptimo: (M0*(M1*M2))
----------------------------
Produto:        (100x100)(100*10)(10*2)
d:              [100,100,10,2]
Custo mínimo    22000
Produto óptimo: (M0*(M1*M2))
----------------------------
Produto:        (2x3)
d:              [2,3]
Custo mínimo    0
Produto óptimo: M0
----------------------------
d =  [100, 100, 10, 2]
22000
( M0 * ( M1 * M2 ) )
```

```
#-------------------------------------------------------------------------
#               Máxima sub-sequência (não contígua) comum
#-------------------------------------------------------------------------
#-----------------------------------------------------------------
#  Versao top down e sua ineficiencia
#  Versao bottom up
#-----------------------------------------------------------------

def maxsubdeq(s,t):
    m=len(s)
    n=len(t)
    ms = [[0]*(n+1) for i in range(m+1)] # matriz iniciada com 0's
    for i in range(1,m):
        for j in range(1,n):
            if s[i] != t[j]:
                ms[i][j] = max(ms[i-1][j],ms[i][j-1])
            else:
                ms[i][j] = 1+ms[i-1][j-1]
        print ms[i]
    return ms[m-1][n-1]

#-----------------------------------------------------------------
# Uso: índices a começar em 1, exemplo s=" abcd"  m=5   indices 1..4
# Com determinacao das sequências máximas comuns
# ms[i][j] = (comprimento,ultimo_indice)
#-----------------------------------------------------------------

def maxsub(s,t):
    m=len(s)                                    # indices: 1,2,...,m
    n=len(t)                                    # indices: 1,2,...,n
    ms  = [[0]*(n+1) for i in range(m+1)]       # matriz iniciada com 0's
    seq = [[""]*(n+1) for i in range(m+1)]      # seq. óptima do inicio até aqui
    for i in range(1,m):
        for j in range(1,n):
            if s[i] != t[j]:                    # símbolos diferentes:
                if ms[i-1][j] >= ms[i][j-1]:
                    ms[i][j]  = ms[i-1][j]
                    seq[i][j] = seq[i-1][j]     # = seq. em (i-1,j)
                else:
                    ms[i][j]  = ms[i][j-1]
                    seq[i][j] = seq[i][j-1]     # = seq. em (i,j-1)
            else:                               # símbolos iguais:
                ms[i][j]  = 1+ms[i-1][j-1]
                seq[i][j] = seq[i-1][j-1]+s[i] # = seq. em (i-1,j-1) + s[i]
        print ms[i]                             # (debug: imprime a matriz)
    print
    return (ms[m-1][n-1], seq[m-1][n-1])        # (comprimento,sequência)

#-------------------------------------------------------------
# testes
#-------------------------------------------------------------
s0=" abazdc"
t0=" aacbad"

s1=" aabbbc"
t1=" baacbb"

s2=" aabbbc"
t2=" bbaacb"

s3=" aa"
t3=" xbabba"

s4=" abaecc"
t4=" bacbace"
```

```
for (s,t) in [(s0,t0),(s1,t1),(s2,t2),(s3,t3),(s4,t4)]:
    print s,t
    # print maxsubdeq(s,t)
    print maxsub(s,t)

#-------------------------------------------------------------
 abazdc  aacbad
[0, 1, 1, 1, 1, 1, 1, 0]
[0, 1, 1, 1, 2, 2, 2, 0]
[0, 1, 2, 2, 2, 3, 3, 0]
[0, 1, 2, 2, 2, 3, 3, 0]
[0, 1, 2, 2, 2, 3, 4, 0]
[0, 1, 2, 3, 3, 3, 4, 0]

(4, 'abad')
#-------------------------------------------------------------
 aabbbc  baacbb
[0, 0, 1, 1, 1, 1, 1, 0]
[0, 0, 1, 2, 2, 2, 2, 0]
[0, 1, 1, 2, 2, 3, 3, 0]
[0, 1, 1, 2, 2, 3, 4, 0]
[0, 1, 1, 2, 2, 3, 4, 0]
[0, 1, 1, 2, 3, 3, 4, 0]

(4, 'aabb')
#-------------------------------------------------------------
 aabbbc  bbaacb
[0, 0, 0, 1, 1, 1, 1, 0]
[0, 0, 0, 1, 2, 2, 2, 0]
[0, 1, 1, 1, 2, 2, 3, 0]
[0, 1, 2, 2, 2, 2, 3, 0]
[0, 1, 2, 2, 2, 2, 3, 0]
[0, 1, 2, 2, 2, 3, 3, 0]

(3, 'aab')
#-------------------------------------------------------------
 aa  xbabba
[0, 0, 0, 1, 1, 1, 1, 0]
[0, 0, 0, 1, 1, 1, 2, 0]

(2, 'aa')
#-------------------------------------------------------------
 abaecc  bacbace
[0, 0, 1, 1, 1, 1, 1, 1, 0]
[0, 1, 1, 1, 2, 2, 2, 2, 0]
[0, 1, 2, 2, 2, 3, 3, 3, 0]
[0, 1, 2, 2, 2, 3, 3, 4, 0]
[0, 1, 2, 3, 3, 3, 4, 4, 0]
[0, 1, 2, 3, 3, 3, 4, 4, 0]

(4, 'abae')
```