**1.Bài toán “ Đổi tiền”**

**a. Đề bài**:

Kho bạc có N loại tiền có mệnh giá lần lượt là D[1], D[2],..., D[N] đơn vị.

Muốn đổi một lượng tiền M đơn vị.

Hỏi phải đổi như thế nào để có số tờ tiền là ít nhất.

**b. Phân tích**

* **Công thức truy hồi**

C(i,j) = số tờ tiền ít nhất để đổi j đơn vị tiền chỉ dùng các loại tiền có mệnh giá lần lượt là D[1],D[2],…,D[N].

Yêu cầu của bài toán là đổi M đơn vị tiền thành N loại tiền có mệnh giá D[1],D[2],…,D[N] sao cho số tờ tiền là ít nhất.Vì vậy số tờ tiền cần tìm là C(N,M).

Nếu không dùng tờ tiền mệnh giá D[i] thì ta có:

C(i,j)= C(i -1, j)

Nếu dung tờ tiền mệnh giá D[i] thì

C(i,j) = C(i, j – D[i]) + 1

Số tờ tiền là ít nhất nên C(i,j) = min(C(i -1, j), C(i, j – D[i] + 1))

Bài toán có thể vô nghiệm.Nên ta coi trường hợp bài toán không có lời giải số tờ tiền ít nhất là .

* **Cơ sở quy hoạch động**

C(i,j) = C(i,0) số tờ tiền ít nhất đổi 0 đơn vị tiền = 0.

C(i,j) = C(0,i) số tờ tiền ít nhất đổi j đơn vị tiền = .

* **Tìm kết quả tối ưu**

C(N, M) = số tờ tiền ít nhất để đổi M đơn vị tiền dùng các tờ tiền mệnh giá D[1]…D[N]

* **Truy vết**

Bắt đầu từ C(N, M), kết thúc khi đã đến cột 0.

Nếu C(i, j) = C(i-1, j) không sử dụng tờ tiền D[i]

Nếu sử dụng tờ tiền mệnh giá Di 🡪 in Di và chuyển sang xét ô C[i, j-Di]

* Thiết kế

//cơ sở quy hoạch động.OXFFFFFFF =

for i=0…m

c[0][i] = OX7FFFFFFF

for i=0…n

c[i][0] = 0

//Bảng phương án

for i = 1…n

for j = 1…m

c[i][j] = c[i-1][j] //D[i] không được chọn

if(d[i]<j)

if(c[i][j-d[i]] + 1 < c[i][j]) //D[i] được chọn và tốt nhất

c[i][j] = c[i][j-d[i]] + 1

//Kết quả tối ưu

if(c[n][m] < OX7FFFFFFF)

print c[n][m]

else

print “Không có cách nào đổi”

//Truy vết

i = n, j = m

while(j > 0) //Khi vẫn còn tiền để đổi

if(c[i][j] = c[i-1][j]) //Nếu không dùng d[i] thì xét mệnh giá khác

i = i-1

else //Nếu không in ra d[i] và vẫn xét tiếp mệnh giá đó

print d[i]

j = j – d[i]

**2. Bài toán bố trí phòng họp**

**a. Đề bài**

Có N cuộc họp cùng đăng kí tại một phòng họp.

Cuộc họp thứ i bắt đầu vào thời điểm ai và kết thúc ở thời điểm bi.

Mỗi thời điểm phòng họp chỉ phục vụ được một cuộc họp.

Hãy bố trí để phòng họp phục vụ được nhiều cuộc họp nhất

**b. Phân tích**

* **Công thức truy hồi**

Gọi L[i] là số cuộc họp nhiều nhất có thể tổ chức nếu chọn cuộc họp i là cuộc họp cuối cùng .Vì vậy số cuộc họp nhiều nhất có thể tổ chức là giá trị lớn nhất của các L[i] khi đã hoàn thành bảng phương án

Nếu có thể tổ chức 1 số cuộc họp khác trước cuộc họp i thì số cuộc họp tối đa có thể tổ chức là max{L[1]…L[i-1]}+1.

max phải thỏa mãn điều kiện cuộc họp cuối cùng của dãy cuộc họp đó phải kết thúc trước hoặc đúng lúc cuộc họp i bắt đầu. Hay L[j] thỏa mãn phải có

b[j] <= a[i]

Nếu không thể tổ chức 1 cuộc họp nào trước cuộc họp i thì L[i] = 1

Công thức thức truy hồi:

L[i] = max(i-1) + 1

* **Cơ sở quy hoạch động**

L[1] = số cuộc họp có thể tổ chức nếu chọn cuộc họp số 1 là cuộc họp cuối cùng = 1

* **Tìm kết quả tối ưu**

Số cuộc họp nhiều nhất có thể tổ chức là max{L[i]}

* **Truy vết**

Bắt đầu từ L[i] là giá trị lớn nhất của mảng L

Kết thúc khi tới giá trị L[j] = 1

Với mỗi L[i] đang xét, tìm ra L[j] đứng trước L[i] đầu tiên thỏa mãn

L[j] = L[i] – 1

b[j] <= a[i]

In cuộc họp j và chuyển sang xét L[j]

c. Thiết kế

//cơ sở quy hoạch động

L[1] = 1

//Bảng phương án

for i = 2…n

L[i] = max(i-1) + 1

//Kết quả tối ưu

maxi= 1 //Giả sử L[1] là giá trị lớn nhất

for i = 2…n

if L[i] > L[maxi]

maxi = i //Cập nhật lại nếu tìm thấy giá trị lớn hơn

print L[maxi] //In ra giá trị lớn nhất

//truy vết

i = maxi //Bắt đầu từ vị trí giá trị lớn nhất

while(L[i] != 1) //Khi chưa xét tới phần tử đầu tiên của dãy

print i, a[i], b[i] //In thông tin cuộc họp i

for j = i-1…1 //Duyệt ngược tìm cuộc họp trước cuộc họp i

if(L[j] = L[i] – 1 && b[j] <= a[i])

i = j

break

print i, a[i], b[i] //Cuộc họp đầu tiên

**3. Bài toán “Biến đổi xâu”**

**a. Đề bài:**

Cho 2 xâu X, Y.

Có 3 phép biến đổi với xâu X:

* Chèn một ký tự.
* Thay thế một ký tự.
* Xoá một ký tự.

Hãy tìm dãy các phép biến đổi ít nhất để biến xâu X thành xâu Y.

**b. Phân tích:**

* Công thức truy hồi:

Gọi C(i, j) = số phép biến đổi ít nhất khi phải biến đổi X[1]…X[i] thành Y[1]..Y[j].

0<=i<=M, 0<=j<=N (0 tương ứng với xâu rỗng)

Khi đó, số phép biến đổi ít nhất biến X thành Y là C(M, N).

Công thức truy hồi:

Nếu X[i] = Y[j] thì C(i, j) = C(i-1, j-1)

Nếu không, C(i, j) = min{C(i, j-1), C(i-1, j), C(i-1, j-1)}+1

* Cơ sở quy hoạch động:

C(i, 0) = số phép biến đổi ít nhất biến thành X[1]…X[i] xâu rỗng = i lần sử dụng phép xóa = i

C(0, j) = số phép biến đổi ít nhất biến xâu rỗng thành Y[1]..Y[j].= j lần sử dụng phép chèn = j

* Tìm kết quả tối ưu

C(M,N)là số phép biến đổi ít nhất biến X thành Y.

d) Truy vết

Bắt đầu từ C(M, N), kết thúc tại C(0, 0)

Giả sử đang xét ô C[i, j].

Nếu Xi = Yj thì không có phép biến đổi nào được dùng tại bước này, chuyển sang xét ô C[i-1, j-1]

Nếu không, chọn 1 trong 3 ô C[i, j-1], C[i-1, j], C[i-1, j-1] xem ô nào có giá trị = C[i, j] – 1 thì chuyển sang xét ô đó

Nếu đó là ô C[i, j-1] thì dùng phép chèn Yj  vào sau vị trí Xi

Nếu đó là ô C[i-1, j-1] thì dùng phép thay thế Xi bới Yj

Nếu đó là ô C[i-1, j] thì dùng phép xóa Xi

c. Thiết kế:

//Cơ sở quy hoạch động

for i = 0…M

c[i][0] = i

for i = 0…N

c[0][i] = i

//Bảng phương án

for i = 0…M

for j=0…N

if(X[i] = Y[j]) //Nếu X[i] = Y[j] thì

C[i][j] = C[i-1][j-1] //chỉ biến đổi phần trước i, j

else //Nếu không thì chọn 1 trong 3 phép biến đổi

C[i][j] = min(C[i][j-1], C[i-1][j-1], C[i-1][j]) +1

//Kết quả tối ưu

print C[M][N]

//Truy vết

i = m, j = n

while(i>0 || j>0)

if(X[i] = Y[j])

i = i-1

j = j-1

else if(C[i][j] = C[i][j-1]+1)

print Chèn Y[j] vào sau vị trí i

j = j-1

else if(C[i][j] = C[i-1][j-1]+1)

print Thay thế ký tự tại vị trí i bằng Y[j]

i = i-1

j = j-1

else if(C[i][j] = C[i-1][j]+1)

print Xóa ký tự tại vị trí i

i = i-1