**Họ tên: Trần Đức Hào**

**Bài toán đổi tiền (Bài toán máy ATM)**

**Yêu cầu:** Giả sử trong 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.

- Công thức truy hồi:  
Gọi B[i][j] là số tờ tiền ít nhất để đổi j tiền xét đến loại tiền tứ i (tức đã xét đến tiền xu có mệnh giá ti)  
  
Xét việc sử dụng đồng xu thứ i:  
  
B[i][j] = min {B[i-1][j], 1+B[i][j-t[i]}  
  
- Cơ sở quy hoạch động:  
  
B[0][j] = VO\_CUNG\_LON;  
B[i][0] = 0;

Code:

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

#define max 100

int n, soTien;

int T[max];

int pa[max];

int B[max][max];

void DocDL()

{

FILE \*f;

f = fopen("DoiTien.txt", "r");

if(!f)

{

puts("Loi doc tep");

getch();

exit(0);

}

fscanf(f, "%d", &n);

fscanf(f, "%d", &soTien);

int i;

for(i=1; i<=n; i++)

fscanf(f, "%d", &T[i]);

fclose(f);

}

int Min(int a, int b)

{

if(a<b)

return a;

return b;

}

void QHD()

{

int i, j;

for(j=1; j<=soTien; j++)

B[0][j] = 32767;

for(i=1; i<=n; i++)

for(j=1; j<=soTien; j++)

{

if(j<T[i])

B[i][j] = B[i-1][j];

else

B[i][j] = Min(B[i-1][j], 1+ B[i][j-T[i]]);

}

}

void TruyVet()

{

int i, j;

i = n; j = soTien;

int k;

k = 1;

while(i && (j>0))

{

if(B[i][j] == B[i-1][j])

i--;

if(B[i][j] == (1 + B[i][j-T[i]]))

{

j -= T[i];

pa[k] = i;

k++;

}

}

}

void InPA()

{

int i;

i = 1;

while(pa[i])

{

printf("%3d", T[pa[i]]);

i++;

}

}

int main()

{

DocDL();

QHD();

TruyVet();

InPA();

getch();

}

**Input:**

Code:

8 20

3 5 7 4 8 6 9 10

**Output:**

Code:

10 10  
**Chứng Minh**  
**Thuật toán như mô tả cho lời giải tối ưu**

**Ta mô tả bài toán tổng quát như sau:**

*Các đồng tiền có mệnh giá là 1, p, p2, ..., pn, trong đó p, n là các số nguyên dương, p >= 2, n >= 2*  
- **Nhận xét**:

Việc sử dụng nhiều nhất đồng tiền mệnh giá cao nhất tương đương với việc lấy L chia cho mệnh

giá đó, phần thương chính là số lượng đồng tiền cần dùng ứng với mệnh giá này, phần dư sẽ

được tiếp tục quy đổi theo quy tắc này.

Ví dụ như có các đồng tiền mệnh giá 1, 3, 32, ..., 31000, có lượng tiền L = 50

L = 50 < 34

Chia L cho 33 => được 1 (sử dụng 1 đồng loại 33), dư 23.

Tiếp tục chia 23 cho 32 => được 2 (sử dụng 2 đồng 32), dư 5

Tiếp tục chia 5 cho 31 => được 1 (sử dụng 1 đồng 31), dư 2

Tiếp tục chia 2 cho 30 => được 2 (sử dụng 2 đồng 30), hết.

Cuối cùng ta có biểu diễn

L = 50 = 1.33 + 2.32 + 1.31 + 2.30,

với số lượng đồng tiền cần dùng là 1 + 2 + 1 + 2 = 6  
  
- Giả sử, lời giải tối ưu cho ta biểu diễn

L = bn.pn + bn-1.pn-1 + ... + b1.p1 + b0

Ta phải có bk < p, với mọi 0 <= k <= n-1

Thật vậy, nếu bk > p, với k nào đó (0 <= k <= n-1), khi đó ta có thể biến đổi

bk+1.pk+1 + bk.pk

= (bk+1 + 1).pk+1 + (bk - p).pk

trong khi đó:

bk+1 + bk > (bk+1 + 1) + (bk - p) (do p > 1)

tức là ta có thể sử dụng ít số lượng đồng tiền hơn nữa bằng cách sử dụng bk+1 + 1 đồng mệnh giá

pk+1 và bk - p đồng mệnh giá pk.

Điều này trái với giả thiết dãy {b} là phương án tối ưu.

Vậy bk < p, lưu ý rằng bk và p đều là các số nguyên cho nên

**bk <= p - 1, với mọi 0 <= k <= n-1**  
  
(**Nếu x, y là các số nguyên và x < y, thì x <= y-1, xảy ra dấu "=" khi và chỉ khi x, y là 2 số**

**nguyên liên tiếp**. Đây là một đánh giá đơn giản nhưng ta nên lưu ý!)  
  
- Ở trên ta đã có  
bk <= p - 1, với mọi 0 <= k <= n-1, nên:  
bi-1.pi-1 + ... + b1.p1 + b0.p0  
<= (p-1).pi-1 + ... + (p-1).p1 + (p-1).p0  
= (p-1)(pi-1 + ... + p1 + p0)  
= pi - 1 < pi, với mọi 1 <= i <= n-1  
  
=> **bi-1.pi-1 + ... + b1.p1 + b0.p0 < pi, với mọi 1 <= i <= n-1 (\*)**  
=> bn-1.pn-1 + ... + b1.p1 + b0.p0 < pn  
Vậy biểu diễn   
L = bn.pn + bn-1.pn-1 + ... + b1.p1 + b0  
chứng tỏ rằng bn chính là thương trong phép chia L cho pn, còn bn-1.pn-1 + ... + b1.p1 + b0.p0 chính là phần dư trong phép chia này.  
  
Sau khi có được bn, ta tiếp tục có  
L - bn.pn = bn-1.pn-1 + bn-2.pn-2 + ... + b1.p1 + b0.p0  
và tiếp tục lập luận như trên (do bất đẳng thức (\*) đúng với mọi 1 <= i <= n-1)  
  
Dãy bn, bn-1, ..., b1, b0 thu được như trên cũng chính là kết quả của thuật toán tham lam.  
  
**Lưu ý**:  
ta hoàn toàn có thể chứng minh theo hướng "thuận", tức là:  
Giả sử với thuật toán tham lam, ta có biểu diễn   
L = an.pn + an-1.pn-1 + ... + a1.p1 + a0  
sau đó bạn lần lượt chứng minh   
an = bn, an-1 = bn-1, ..., a1 = b1, a0 = b0  
(vẫn sử dụng bất đẳng thức (\*) ở trên)