CHUYÊN ĐỀ NÂNG CAO 6

**CHUYÊN ĐỀ 6: QUY HOẠCH ĐỘNG**

**C. LỚP BÀI TOÁN CÁI TÚI**

**1. Mô hình**

Trong siêu thị có n đồ vật (n≤1000), đồ vật thứ i có trọng lượng là W[i]≤10­00 và giá trị V[i] ≤1000. Một tên trộm đột nhập vào siêu thị, tên trộm mang theo một cái túi có thể mang được tối đa trọng lượng M (M≤1000). Hỏi tên trộm sẽ lấy đi những đồ vật nào để được tổng giá trị lớn nhất.

Giải quyết bài toán trong các trường hợp sau:

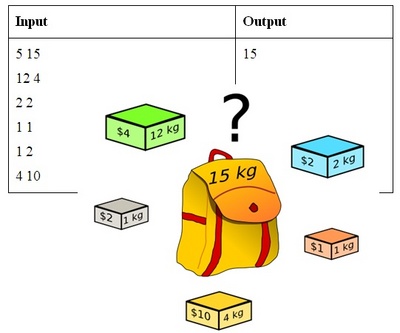
* Mỗi vật chỉ được chọn một lần.
* Mỗi vật được chọn nhiều lần (không hạn chế số lần)

**InputData:** file văn bản **Bag.inp**

* Dòng 1: n, M cách nhau ít nhất một dấu cách
* n dòng tiếp theo: Mỗi dòng gồm 2 số Wi, Vi, là chi phí và giá trị đồ vật thứ i.

**OutputData:** file văn bản **bag.out**: Ghi giá trị lớn nhất tên trộm có thể lấy

**Example**



**2. Hướng dẫn giải**

**Trường hợp mỗi vật được chọn 1 lần: Vali B**

Ta nhận thấy rằng: Giá trị của cái túi phụ thuộc vào 2 yếu tố: Có bao nhiêu vật đang được xét và trọng lượng còn lại cái túi có thể chứa được, do vậy chúng ta có 2 đại lượng biến thiên. Cho nên hàm mục tiêu sẽ phụ thuộc vào hai đại lượng biến thiên. Do vậy bảng phương án của chúng ta sẽ là bảng 2 chiều.

Gọi F[i,j] là tổng giá trị lớn nhất của cái túi khi xét từ vật 1 đến vật i và trọng của cái túi chưa vượt quá j. Với giới hạn j, việc chọn tối ưu trong số các vật {1,2,…,i-1,i} để có giá trị lớn nhất sẽ có hai khả năng:

- Nếu không chọn vật thứ i thì F[i,j] là giá trị lớn nhất có thể chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng là j, tức là: **F[i,j]:=F[i-1,j].**

- Nếu có chọn vật thứ i (phải thỏa điều kiện W[i] ≤ j) thì F[i,j] bằng giá trị vật thứ i là V[i] cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng j-W[i] tức là về mặt giá trị thu được: **F[i,j]:=V[i]+F[i-1,j-W[i]]**

Vậy chúng ta phải xem xét xem nếu chọn vật i hay không chọn vật i thì sẽ tốt hơn. Từ đó chúng ta có công thức truy hồi như sau.

* **F[0,j] = 0** (hiển nhiên) – Bài toán con nhỏ nhất.
* **F[i,j]= max(F[i-1,j], V[i]+F[i-1,j-W[i]]**

**Trường hợp mỗi vật được chọn nhiều lần**: Tương tự như suy luận ở trên ta xét: **Vali A**

- Nếu không chọn vật thứ i thì F[i,j] là giá trị lớn nhất có thể chọn trong số các vật {1,2,…,i-1} với giới hạn trọng lượng là j, tức là: **F[i,j]:=F[i-1,j].**

- Nếu có chọn vật thứ i (phải thỏa điều kiện W[i] ≤ j) thì F[i,j] bằng giá trị vật thứ i là V[i] cộng với giá trị lớn nhất có thể có được bằng cách chọn trong số các vật {1,2,…,i} (vì vật i vẫn có thể được chọn tiếp) với giới hạn trọng lượng j-W[i] tức là về mặt giá trị thu được:

**F[i,j]:=V[i]+F[i,j-W[i]]**

Do vậy chúng ta có **công thức truy hồi** như sau:

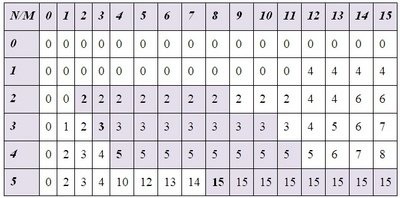
* **F[0,j] = 0** (hiển nhiên) – Bài toán con nhỏ nhất.
* **F[i,j]= max(F[i-1,j], V[i]+F[i,j-W[i]]**

**3. Bảng phương án**

Ta xây dựng bảng phương án dựa trên công thức truy hồi trên. Để kiểm tra kết quả có chính xác hay không (nếu không chính xác chúng ta xây dựng lại hàm mục tiêu). Thông qua cách xây dựng hàm mục tiêu và bảng phương án chúng ta sẽ định hướng việc truy vết.

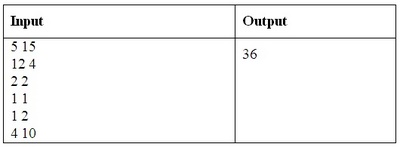
**Example (**trường hợp mỗi vật chỉ được chọn 1 lần)

**Bảng phương án:**

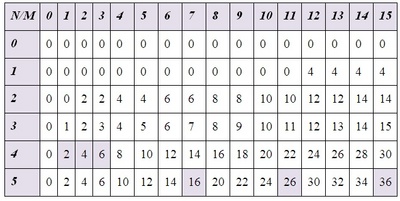


Vậy chúng ta có thể chọn vật 2, 3, 4, 5.

**Example (**trường hợp mỗi vật được chọn nhiều lần)



**Bảng phương án:**



Chúng ta có thể chọn vật 4 (3 lần) và vật 5 (3 lần).

**4. Truy vết**

**Trường hợp 1**: Trong bảng *phương* án F[n,m] chính là giá trị lớn nhất thu được khi chọn trong cả n vật với giới hạn trọng lượng là M.

Nếu f[n,M]=f[n-1,M] thì tức là không chọn vật thứ n, ta truy về f[n-1,M]. Còn nếu f[n,M]≠f[n-1,M] thì ta thông báo rằng phép chọn tối ưu có chọn vật thứ n và truy về f[n-1,M-Wn].

Procedure truyvet(n,m:longint);

Begin

if F[n,m]=0 then exit;

if F[n,m]=F[n-1,m] then truyvet(n-1,m) else

Begin

write(n,' ');

truyvet(n-1,m-w[n]);

End;

End;

**Trường hợp 2:** Trong bảng phương án F[n,m] chính là giá trị lớn nhất thu được khi chọn trong cả n vật với giới hạn trọng lượng là M.

Nếu f[n,M]=f[n-1,M] thì tức là không chọn vật thứ n, ta truy về f[n-1,M]. Còn nếu f[n,M]≠f[n-1,M] thì ta thông báo rằng phép chọn tối ưu có chọn vật thứ n và truy về f[n,M-Wn].

Procedure truyvet(n,m:longint);

Begin

if F[n,m]=0 then exit;

if F[n,m]=F[n-1,m] then truyvet(n-1,m) else

Begin

d:=d+1;

if **F[n,m-b[n]]=F[n-1,m-W[n]]** then

Begin inc(t); V1[t]:= n; v2[t]:= d; d:=0; end;

truyvet(**n,m-W[n]**);

End;

End;

Mảng v1 lưu lại chỉ số các vật được chọn, mảng v2 lưu lại số lượng chọn của từng vật. Sau khi truy vết thì ta có 2 mảng này lưu lại vật nào được chọn bao nhiêu lần. Ta chỉ cần in mảng này ra là xong:

For i:=t downto 1 do writeln(v1[i],#32,v2[i]);

**5.** **Cài đặt**

Bài Vali A:

Procedure process;

Begin

for i:=1 to n do

for j:=1 to m do

if w[i]<=j then F[i,j]:=max(F[i-1,j],**F[i-1,j-w[i]]** + v[i])

else F[i,j]:=F[i-1,j];

End;

Bài vali b: cài tương tự

Các em cài đặt và làm online tại: Bài Catu2 <http://laptrinh.ntu.edu.vn/Problem/Details/1148>

**6. Một số bài toán khác tương tự Vali B**

**a) Dãy con có tổng bằng S:**

Cho dãy a1,a2,..an. Tìm một dãy con của dãy đó có tổng bằng S.

**Hướng dẫn**

Đặt L[i,t)=1 nếu *có thể tạo ra tổng t từ một dãy con của dãy gồm các phần tử a1,a2,..ai*. Ngược lại thì L[i,t)=0. Nếu L[n,S)=1 thì đáp án của bài toán trên là “có”.

Ta có thể tính L[i,t] theo công thức: L[i,t]=1 nếu L[i-1,t]=1 hoặc L[i-1,t-a[i]]=1.

**Cài** **đặt**

Nếu áp dụng luôn công thức trên thì ta cần dùng bảng phương án hai chiều. Ta có thể nhận xét rằng để tính dòng thứ i, ta chỉ cần dòng i-1. Bảng phương án khi đó chỉ cần 1 mảng 1 chiều L[0..S] và được tính như sau:

L[t]:=0; L[0]:=1;

for i := 1 to n do

for t := S downto a[i] do

if (L[t]=0) and (L[t-a[i]]=1) then L[t]:=1;

Dễ thấy chi phí không gian của cách cài đặt trên là O(m), chi phí thời gian là O(nm), với m là tổng của n số. Hãy tự kiểm tra xem tại sao vòng for thứ 2 lại là for downto chứ không phải là for to.

**b) Chia kẹo**

Cho n gói kẹo, gói thứ i có ai viên. Hãy chia các gói thành 2 phần sao cho chênh lệch giữa 2 phần là ít nhất.

**Hướng dẫn**: Gọi T là tổng số kẹo của n gói. Chúng ta cần tìm số S **lớn nhất** thoả mãn:

• S≤T/2.

• Có một dãy con của dãy a có tổng bằng S.

Khi đó sẽ có cách chia với chênh lệch 2 phần là **T-2S** là nhỏ nhất và dãy con có tổng bằng S ở trên gồm các phần tử là các gói kẹo thuộc phần thứ nhất. Phần thứ hai là các gói kẹo còn lại.

**c) Market (Olympic Balkan 2000)**

Người đánh cá Clement bắt được n con cá, khối lượng mỗi con là ai, đem bán ngoài chợ. Ở chợ cá, người ta không mua cá theo từng con mà mua theo một **lượng** nào đó. Chẳng hạn 3 kg, 5kg…

Ví dụ: có 3 con cá, khối lượng lần lượt là: 3, 2, 4. Mua lượng 6 kg sẽ phải lấy con cá thứ 2 và và thứ 3. Mua lượng 3 kg thì lấy con thứ nhất. Không thể mua lượng 8 kg.

Nếu bạn là người đầu tiên mua cá, có bao nhiêu lượng bạn có thể chọn?

**Hướng dẫn**: Thực chất bài toán là tìm các số S mà có một dãy con của dãy a có tổng bằng S.

Ta có thể dùng phương pháp đánh dấu của bài chia kẹo ở trên rồi đếm các giá trị t mà L[t]=1.

**d) Điền dấu**

Cho n số tự nhiên a1,a2, ...,an. Ban đầu các số được đặt liên tiếp theo đúng thứ tự cách nhau bởi dấu "?": a1?a2?...?an. Cho trước số nguyên S, có cách nào thay các dấu "?" bằng dấu + hay dấu − để được một biểu thức số học cho giá trị là S không?

**Hướng dẫn**: Đặt L(i,t)=1 nếu có thể điền dấu vào i số đầu tiên và cho kết quả bằng t. Ta có công thức sau để tính L:

• L(1,a[1]) =1.

• L(i,t)=1 nếu L(i-1,t+a[i])=1 hoặc L(i-1,t-a[i])=1.

Nếu L(n,S)=1 thì câu trả lời của bài toán là có. Khi cài đặt, có thể dùng một mảng 2 chiều (lưu toàn bộ bảng phương án) hoặc 2 mảng một chiều (để lưu dòng i và dòng i-1). Chú ý là chỉ số theo t của các mảng phải có cả phần âm (tức là từ -T đến T, với T là tổng của n số), vì trong bài này chúng ta dùng cả dấu - nên có thể tạo ra các tổng âm.

Bài này có một biến thể là *đặt dấu sao cho kết quả là một số chia hết cho k*. Ta có thuật giải tương tự bài toán trên bằng cách thay các phép cộng, trừ bằng các *phép cộng và trừ theo* *môđun k* và dùng mảng đánh dấu với các giá trị từ 0 đến k-1 (là các số dư có thể có khi chia cho k). Đáp số của bài toán là L(n,0).

**e) Expression (ACM 10690)**

Cho n số nguyên. Hãy chia chúng thành 2 nhóm sao cho tích của tổng 2 nhóm là lớn nhất.

**Hướng dẫn**: Gọi T là tổng n số nguyên đó. Giả sử ta chia dãy thành 2 nhóm, gọi S là tổng của một nhóm, tổng nhóm còn lại là T-S và tích của tổng 2 nhóm là S\*(T-S). Bằng phương pháp đánh dấu ta xác định được mọi số S là tổng của một nhóm (như bài Market) và tìm số S sao cho S\*(T-S) đạt max.

**7. Một số bài toán khác tương tự mô hình Vali A**

**a) Farmer (IOI 2004)**

Một người có N mảnh đất và M dải đất. Các mảnh đất có thể coi là một tứ giác và các dải đất thì coi như một đường thẳng. Dọc theo các dải đất ông ta trồng các cây bách, dải đất thứ i có ai cây bách. Ông ta cũng trồng các cây bách trên viền của các mảnh đất, mảnh đất thứ j có bj cây bách. Cả ở trên các mảnh đất và dải đất, xen giữa 2 cây bách ông ta trồng một cây ôliu.

Ông ta cho con trai được chọn các mảnh đất và dải đất tuỳ ý với điều kiện tổng số cây bách không vượt quá Q. Người con trai phải chọn thế nào để có nhiều cây ôliu (loài cây mà anh ta thích) nhất.

**Hướng dẫn**: Dễ thấy mảnh đất thứ i có ai cây ôliu và dải đất thứ j có bj-1 cây ôliu. Coi các mảnh đất và dải đất là các “đồ vật”, đồ vật thứ k có khối lượng wk và giá trị vk (nếu k là mảnh đất i thì wk=vk=ai, nếu k là dải đất j thì wk=bj, vk=bj-1). Ta cần chọn các “đồ vật”, sao cho tổng “khối lượng” của chúng không vượt Q và tổng “giá trị” là lớn nhất. Đây chính là bài toán xếp balô đã trình bày ở trên.

**b) Đổi tiền**

Ở đất nước Omega người ta chỉ tiêu tiền xu. Có N loại tiền xu, loại thứ i có mệnh giá là ai đồng. Một người khách du lịch đến Omega du lịch với số tiền M đồng. Ông ta muốn đổi số tiền đó ra tiền xu Omega để tiện tiêu dùng. Ông ta cũng muốn số đồng tiền đổi được là ít nhất (cho túi tiền đỡ nặng khi đi đây đi đó). Bạn hãy giúp ông ta tìm cách đổi tiền.

**Hướng dẫn**: Bài toán này khá giống bài toán xếp balô (*“khối lượng*” là mệnh giá, *“giá trị”* là 1), chỉ có một số thay đổi nhỏ: *số* *đồng xu mỗi loại* *được chọn tuỳ ý* (trong bài toán xếp balô mỗi đồ vật chỉ được chọn 1 lần) và *tổng giá trị yêu cầu là nhỏ nhất*. Do đó ta cũng xây dựng hàm QHĐ một cách tương tự: Gọi L(i,t) là số đồng xu ít nhất nếu đổi t đồng ra i loại tiền xu (từ 1 đến i). Công thức tính L(i,t) như sau:

• L(i,0)=0;

• L(0,t)= ∞ với t>0.

• L(i,t)=L(i-1,t) nếu t<ai.

• L(i,t)=min(L(i-1,t), L(i,t-ai)+1) nếu t ≥ai.

Công thức này khác công thức của bài xếp balô ở chỗ: dùng hàm **min** chứ không phải hàm **max** (vì cần tìm cách chọn ít hơn) và nếu chọn đồng xu thứ i thì L(**i**,t)=L(**i**,t-ai)+1 (vì ta vẫn còn được chọn đồng xu thứ i đó nữa), khác với khi xếp balô là: nếu chọn đồ vật thứ i thì L(**i**,t)=L(**i-1**,t-ai)+bi vì đồ vật i chỉ được chọn một lần.