**Кумулятивные суммы**

Кумулятивные суммы – динамика, позволяющая быстро высчитывать сумму элементов на определенных промежутках массива.

*Одномерный массив:*

*//asum – массив сумм*

*//a – массив элементов*

asum[i]=asum[i-1]+a[i]; при чем asum[0]=a[0];

fsum(i, j) = asum[j]-asum[i-1];

*Двумерный массив:*

asum[i][j]=asum[i-1][j]+asum[i][j-1]+a[i][j]-asum[i-1][j-1];

fsum(x1,y1,x2,y2) = asum[x2][y2]-asum[x2][y1-1]-asum[x1-1][y2]+asum[x1-1][y1-1];

**Задача БАНКОМАТ**

Рассмотрим следующую задачу. В обороте находятся банкноты *k*  различных номиналов: *a*1, *a*2, ..., *a*k рублей. Банкомат должен выдать сумму в *N* рублей при помощи минимального количества банкнот или сообщить, что запрашиваемую сумму выдать нельзя. Будем считать, что запасы банкнот каждого номинала неограничены.

Рассмотрим такой алгоритм: будем выдавать банкноты наибольшего номинала, пока это возможно, затем переходим к следующему номиналу. Например, если имеются банкноты в 10, 50, 100, 500, 1000 рублей, то при *N* = 740 рублей такой алгоритм выдаст банкноты в 500, 100, 100, 10, 10, 10, 10 рублей. Подобные алгоритмы называют «жадными», поскольку каждый раз при принятии решения выбирается тот вариант, который кажется наилучшим в данной ситуации (чтобы использовать наименьшее число банкнот каждый раз выбирается наибольшая из возможных банкнот).

Но для решения данной задачи в общем случае жадный алгоритм оказывается неприменимым. Например, если есть банкноты номиналом в 10, 60 и 100 рублей, то при *N* = 120 жадный алгоритм выдаст три банкноты: 100 + 10 + 10, хотя есть способ, использующий две банкноты: 60 + 60. А если номиналов банкнот только два: 60 и 100 рублей, то жадный алгоритм вообще не сможет найти решения.

Но эту задачу можно решить при помощи метода динамического программирования. Пусть *F*(*n*) -- минимальное количество банкнот, которым можно заплатить сумму в *n* рублей. Очевидно, что *F*(0) = 0, *F*(*a*1) = *F*(*a*2) =...= *F*(*a*k) = 1. Если некоторую сумму *n* невозможно выдать, будем считать, что *F*(*n*) = $ \infty$ (бесконечность).

Выведем рекуррентную формулу для *F*(*n*), считая, что значения *F*(0), *F*(1), ..., *F*(*n* - 1) уже вычислены. Как можно выдать сумму *n*? Мы можем выдать сумму *n* - *a*1, а потом добавить одну банкноту номиналом *a*1. Тогда нам понадобится *F*(*n* - *a*1) + 1 банкнота. Можем выдать сумму *n* - *a*2 и добавить одну банкноту номиналом *a*2, для такого способа понадобится *F*(*n* - *a*2) + 1 банкнота и т. д. Из всевозможных способов выберем наилучший, то есть:

*F*(*n*) = min(*F*(*n* - *a*1), *F*(*n* - *a*2),..., *F*(*n* - *a*k)) + 1.

Теперь заведем массив F[n+1], который будем последовательно заполнять значениями выписанного рекуррентного соотношения. Будем предполагать, что количество номиналов банкнот хранится в переменной int k, а сами номиналы хранятся в массиве int a[k].

const int INF=1000000000; // Значение константы }бесконечность}   
 int F[n+1];   
 F[0]=0;   
 int m, i;   
 for(m=1; m<=n; ++m)   // заполняем массив F   
 {                     // m - сумма, которую нужно выдать   
   F[m]=INF;           // помечаем, что сумму m выдать нельзя   
   for(i=0; i<k; ++i)  // перебираем все номиналы банкнот   
   {   
     if(m>=a[i] && F[m-a[i]]+1<F[m])   
       F[m] = F[m-a[i]]+1; // изменяем значение F[m], если нашли   
   }                       // лучший способ выдать сумму m   
 }

После окончания этого алгоритма в элементе F[n] будет храниться минимальное количество банкнот, необходимых, чтобы выдать сумму n. Как теперь вывести представление суммы n при помощи F(n) банкнот? Опять рассмотрим все номиналы банкнот и значения *n* - *a*1, *n* - *a*2, ..., *n* - *a*k. Если для какого-то *i* окажется, что *F*(*n* - *a*i) = *F*(*n*) - 1, значит, мы можем выдать банкноту в *a*i рублей и после этого свести задачу к выдаче суммы *n* - *a*i, и так будем продолжать этот процесс, пока величина выдаваемой суммы не станет равна 0:

if (F[n]==INF)  
   cout<<"Требуемую сумму выдать невозможно"<<endl;  
else  
 while(n>0)  
 for(i=0;i<k;++i)  
 if (F[n-a[i]]==F[n]-1)  
 {  
 cout<<a[i]<<" ";  
 n-=a[i];  
 break;  
 }

**Задача о рюкзаке**

Грабитель, проникший в банк, обнаружил в сейфе *k* золотых слитков, массами *w*1, *w*2, ..., *w*k килограмм. При этом грабитель может унести не более *W* килограмм. Определите набор слитков, который должен взять грабитель, чтобы унести как можно больше золота.

Эта задача является частным случаем задачи об укладке рюкзака. Сформулируем ее в общем случае.

Дано *k* предметов, *i*-й предмет имеет массу *w*i > 0 и стоимость *p*i > 0. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины *W* (вместимость рюкзака), а суммарная стоимость была максимальна. Другими словами, нужно определить набор бинарных величин (*b*1, *b*2,..., *b*k), такой, что

*b*1*w*1 + *b*2*w*2 +...+ *b*k*w*k$\displaystyle \le$*W*,

а величина *b*1*p*1 + *b*2*p*2 +...+ *b*k*p*k -- максимальная. Величина *b*i равна 1, если *i*-й предмет включается в набор, и равна 0 в противном случае.

Задача укладки рюкзака очень сложна. Если перебирать всевозможные подмножества данного набора из *k* предметов, то получится решение сложности не менее чем *O*(2k). В настоящее время неизвестен (и, скорее всего, вообще не существует) алгоритм решения этой задачи, сложность которого является многочленом от *k*.

Мы рассмотрим решение данной задачи для случая, когда все входные данные -- целочисленные, сложность которого будет *O*(*kW*).

Рассмотрим следующую функцию. Пусть *A*(*s*, *n*) есть максимальная стоимости предметов, которые можно уложить в рюкзак максимальной вместимости *n*, если можно использовать только первые *s* предметов из заданных *k*.

Зададим краевые значения функции *A*(*s*, *n*).

Если *s* = 0, то *A*(0, *n*) = 0 для всех *n* (ни один предмет нельзя брать, поэтому максимальная стоимость равна 0).

Если *n* = 0, то *A*(*s*, 0) = 0 для всех *s* (можно брать любые из первых *s* предметов, но вместимость рюкзака равна 0).

Теперь составим рекуррентное соотношение в общем случае. Необходимо из предметов с номерами 1, ..., *s* составить рюкзак максимальной стоимости, чей вес не превышает *n*. При этом возможно два случая: когда в максимальный рюкзак включен предмет с номером *s* и когда предмет *s* не попал в максимальный рюкзак.

Если предмет *s* не попал в максимальный рюкзак массы *n*, то максимальный рюкзак будет составлен только из предметов с номерами 1, ..., *s* - 1, следовательно, *A*(*s*, *n*) = *A*(*s* - 1, *n*).

Если же в максимальный рюкзак включен предмет *s*, то масса оставшихся предметов не превышает *n* - *w*s, а от добавления предмета *s* общая стоимость рюкзака увеличивается на *p*s. Значит, *A*(*s*, *n*) = *A*(*s* - 1, *n* - *w*s) + *p*s. Теперь из двух возможных вариантов составить рюкзак массы, не превосходящей *n*, из предметов 1, ..., *s* нужно выбрать наилучший:

*A*(*s*, *n*) = max$\displaystyle \left(\vphantom{A(s-1,n),A(s-1,n-w_s)+p_s}\right.$*A*(*s* - 1, *n*), *A*(*s* - 1, *n* - *w*s) + *p*s$\displaystyle \left.\vphantom{A(s-1,n),A(s-1,n-w_s)+p_s}\right)$.

Теперь составим программу. Будем считать, что веса предметов хранятся в массиве w[1], ..., w[k], а их стоимости в массиве p[1], ..., p[k]. Значения функции *A*(*s*, *n*), где 0$ \le$*s*$ \le$*k*, 0$ \le$*n*$ \le$*W*, будем хранить в массиве A[k+1][W+1].

int A[k+1][W+1];

for(n=0;n<=W;++n) // Заполняем нулевую строчку

A[0][n]=0;

for(s=1;s<=k;++s) // s - максимальный номер предмета,

{ // который можно использовать

for(n=0;n<=W;++n) // n - вместимости рюкзака

{

A[s][n]=A[s-1][n];

if ( n>=w[s] && ( A[s-1][n-w[s]]+p[s] > A[s][n]) )

A[s][n] = A[s-1][n-w[s]]+p[s];

}

}

В результате исполнения такого алгоритма в элементе массива A[k][W] будет записан ответ на поставленную задачу. Легко видеть, что сложность этого алгоритма, равно как и объем используемой им памяти, являются величиной *O*(*kW*).

Рассмотрим пример работы этого алгоритма. Пусть максимальная вместимость рюкзака *W* = 15, количество предметов *k* = 5, их стоимости и массы таковы:   
*w*1 = 6, *p*1 = 5,    *w*2 = 4, *p*2 = 3,    *w*3 = 3, *p*3 = 1,   
*w*4 = 2, *p*4 = 3,    *w*5 = 5, *p*5 = 6.

В приведенной ниже таблице указаны значения заполненного массива A[k+1][W+1].

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |  |
| *s* = 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| *s* = 1 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |  |
| *s* = 2 | 0 | 0 | 0 | 0 | 3 | 3 | 5 | 5 | 5 | 5 | 8 | 8 | 8 | 8 | 8 | 8 |  |
| *s* = 3 | 0 | 0 | 0 | 1 | 3 | 3 | 5 | 5 | 5 | 6 | 8 | 8 | 8 | 9 | 9 | 9 |  |
| *s* = 4 | 0 | 0 | 3 | 3 | 3 | 4 | 6 | 6 | 8 | 8 | 8 | 9 | 11 | 11 | 11 | 12 |  |
| *s* = 5 | 0 | 0 | 3 | 3 | 3 | 6 | 6 | 9 | 9 | 9 | 10 | 12 | 12 | 14 | 14 | 14 |  |

Первая строка массива соответствует значениям *A*(0, *n*). Поскольку ни одного предмета брать нельзя, то строка заполнена нулями: из пустого множества предметов можно составить рюкзак нулевой массы.

Вторая строка массива соответствует значению *s* = 1, то есть рюкзак можно составлять только из первого предмета. Вес этого предмета *w*1 = 6, а его стоимость *p*1 = 5. Поэтому при *n* < 6 мы не можем включить этот предмет в рюкзак и значение *A*(1, *n*) равно 0 при *n* < 6. Если *n*$ \ge$*w*1, то мы можем включить первый предмет в рюкзак, а поскольку других предметов нет, то *A*(1, *n*) = 5 (так как *p*1 = 5).

Рассмотрим третью строку массива, соответствующую двум предметам (*s* = 2). Добавляется второй предмет, более легкий и менее ценный, чем первый (*w*2 = 4, *p*2 = 3). Поэтому *A*(2, *n*) = 0 при *n* < 4 (ни один предмет взять нельзя), *A*(2, *n*) = 3 при *n* = 4 и *n* = 5 (в рюкзак включается предмет номер 2 ценности 3), *A*(2, *n*) = 5 при 6$ \le$*n*$ \le$9 (при данном *n* выгоднее в рюкзак включить предмет 1, поскольку его ценность выше) и, наконец, *A*(2, *n*) = 8 при *n*$ \ge$10 (при данной вместимости рюкзака можно взять оба предмета).

Аналогично заполняются остальные строки массива, при заполнении элемента *A*(*s*, *n*) рассматривается две возможности: включать или не включать предмет с номером *s*.

Как теперь вывести на экран тот набор предметов, который входит в максимальный рюкзак? Сравним значение A[k][W] со значением A[k-1][W]. Если они равны, то максимальный рюкзак можно составить без использования предмета с номером k. Если не равны, то предмет c номером k обязательно входит в максимальный рюкзак. В любом случае, задача печати рюкзака сводится к задаче печати рюкзака для меньшего числа предметов. Напишем это в виде рекурсивной функции Print(int s, int n), которая по параметрам s и n печатает номера предметов, входящих в максимальный рюкзак массой не более n и составленный из предметов 1, ..., s:

void Print(int s, int n)

{

if (A[s][n]==0) // максимальный рюкзак для параметров (s,n)

return; // имеет нулевую ценность,

// поэтому ничего не выводим

else if (A[s-1][n] == A[s][n])

Print(s-1,n); // можно составить рюкзак без предмета s

else

{

Print(s-1,n-w[s]); // Предмет s должен обязательно

cout<<s<<endl; // войти в рюкзак

}

}  
Для печати искомого рюкзака необходимо вызвать функцию с параметрами (k,W).

В приведенном примере для печати максимального рюкзака вызовем функцию Print(5,15). Поскольку *A*(5, 15) = 14, а *A*(4, 15) = 12 (с использованием только первых 4 предметов мы можем собрать рюкзак максимальной стоимости 12, а с использованием всех 5 предметов -- стоимости 14), предмет номер 5 обязательно входит в рюкзак. Далее рассмотрим *A*(4, 10) (общая вместимость рюкзака уменьшилась на вес включенного предмета). Поскольку *A*(4, 10) = *A*(3, 10) = *A*(2, 10) = 8, то мы можем исключить из рассмотрения предметы номер 4 и 3 -- можно собрать рюкзак вместимости 10 и стоимости 8 только из первых двух предметов. Для этого необходимо включить оба этих предмета. Таким образом, оптимальный рюкзак будет состоять из предметов 1, 2, 5, его масса будет равна 6 + 4 + 5 = 15, а стоимость -- 5 + 3 + 6 = 14.

Можно составить рюкзак и по-другому. Поскольку вес предмета 4 равен двум, его стоимость *p*4 равна трём, а *A*(4, 10) = *A*(3, 8) + *p*4, то мы можем включить в наш рюкзак предмет 4. Теперь рассмотрим *A*(3, 8) = 5 -- как составить рюкзак массы не более 8 и стоимости 5 из первых трех предметов. Поскольку *A*(3, 8) = *A*(2, 8) = *A*(1, 8) = 5, то мы исключаем из рассмотрения предметы 3 и 2, но включаем предмет 1. Получим рюкзак из предметов 1, 4, 5, его масса будет 6 + 2 + 5 = 13 и стоимость также равна 5 + 3 + 6 = 14. Поскольку стоимость обоих полученных рюкзаков получилась одинаковой, а масса в каждом случае не превосходит максимально допустимого значения *W* = 15, то оба решения подходят.