### 穷举法★

最朴素的算法是穷举法，但不能通过全部数据。

因为n个魔法石中的每个都有装和不装两种选择，故可以1和0表示。穷举所有的n位二进制数进行统计即可。

参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | //简单背包问题２  #include <stdio.h>  int a[30+1];//状态为0-1，装入为1，不装入为0  long v,n,i,j,s,f;  long t[30+1];//保存每物品的体积  int main()  {  scanf("%ld%ld",&v,&n);  for(i=1;i<=n;i++)  scanf("%ld",&t[i]);  memset(a,0,sizeof(a));  s=0;  while(a[0]==0)//穷举  {  f=0;  for(i=1;i<=n;i++)  f=f+a[i]\*t[i];//计算不同状态的总重量  if((f>s)&&(f<=v))  s=f;  j=n;  while(a[j]==1)//进位  j=j-1;  a[j]=1;  for(i=j+1;i<=n;i++)  a[i]=0;  }  printf("%ld\n",v-s);  return 0;  } |

### 动态规划１★

使用二维数组f[i][j]，表示前i个物品装入容量为j的背包能获得的最大体积，则动态转移方程为： f[i][j]=max{f[i-1][j]，f[i-1][j-a[i]]+a[i]}。该方程的意思是，要么第i件物品不放，即f[i-1][j]；要么放第i件物品，这样就必须从j容量的背包里腾出第i件物品的空间才行，即f[i-1][j-a[i]]，然后加上a[j]的重量，即f[i-1][j-a[i]]+a[i]。

参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //简单背包问题２－动规１  #include <iostream>  using namespace std;  int f[31][20001];  int main()  {  freopen("pack2.in","r",stdin);  freopen("pack2.out","w",stdout);  int V,n,i,j;  cin>>V>>n;  int a[n+1];  for(i=1;i<=n;i++)  cin>>a[i];  for(i=1;i<=n;i++)  for(j=1;j<=V;j++)  if(j<a[i])  f[i][j]=f[i-1][j];  else  f[i][j]=max(f[i-1][j],f[i-1][j-a[i]]+a[i]);  cout<<V-f[n][V]<<'\n';  return 0;  } |

### 动态规划２★

前一种方法占用内存空间较大，可以改用一维数组f[j]，仍表示前i个物品装包能获得的最大体积。

　　参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //简单背包问题２-动规２  #include <iostream>  #include <cstdlib>  using namespace std;  int f[20001];//f[j]表示前i个物品装入获得的最大体积  int main()  {  freopen("pack2.in","r",stdin);  freopen("pack2.out","w",stdout);  int V,n,i,j;  cin>>V>>n;  int a[n+1];  for(i=1;i<=n;i++)  cin>>a[i];  for(i=1;i<=n;i++)  for(j=V;j>=1;j--)  if(j<a[i])  f[j]=max(f[j-1],f[j]);//虽然f[j]总是大于f[j-1],但这句不能少  else  f[j]=max(f[j],f[j-a[i]]+a[i]);  cout<<V-f[V];  return 0;  } |

### 动态规划３★

　继续压缩空间，设布尔数组f[20001]，f[i]=1表示容量为i的背包空间可被装满，f[i]=0即表示容量为i的背包空间无法装满。若f[j]=1，下一个物品为a[i]，若该物品可装，显然f[j+a[i]]也为1。

　　参考程序如下所示：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | //简单背包问题２－动规３  #include <iostream>  #include <fstream>  using namespace std;  ifstream fin("pack2.in");  ofstream fout("pack2.out");  bool f[20001];  int main()  {  int v,n,a[31],i,j,d;  fin>>v>>n;    for(i=1;i<=n;i++)  fin>>a[i];    for(i=1;i<=n;i++)//对n种物品  {  for(j=v;j>0;j--)//j为剩余空间  if(f[j]==1 && j+a[i]<=v)  f[j+a[i]]=1;  f[a[i]]=1;  }  d=v;  while(d>0 && f[d]!=1)//找出最大装载空间  d--;    fout<<v-d<<endl;  fin.close();  fout.close();  return 0;  } |