输入：第一行包含一个整数T，例数。  
接着为T的情况下，每一种情况下三行，第一行包含两个整数N，V，（N- <= 1000，V <= 1000）表示骨骼的数目和他的袋子的体积。和第二行包含表示各骨骼的值N的整数。第三行包含代表每个骨骼的体积N个整数。

输出：每表示最大总价值的行一个整数

(详见杭电2602题)

#include <iostream>

using namespace std;

int main()

{

int t;cin>>t;

while(t--)

{

int n,va,w[10005]={0},v[10005]={0},f[10005]={0}; //数组f放入总价值多少

cin>>n>>va; //输入背包容量和物品数量

for(int i=0;i<n;i++) //输入价值

cin>>w[i];

for(int i=0;i<n;i++) //输入重量

cin>>v[i];

for(int i=0;i<n;i++) //判断第i个物品

{

for(int j=va;j>=v[i];j--) //背包重量需逆序不然会重复放置物品

f[j]=max(f[j],f[j-v[i]]+w[i]); //状态转移方程，判断第i个物品是否放入

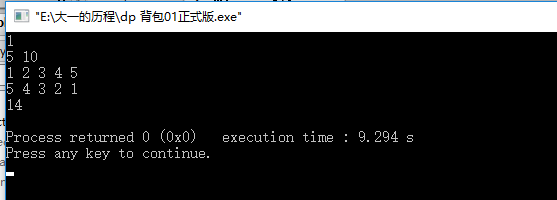
}

cout<<f[va]<<endl; //输出结果

}

return 0;

}



背包01的题主要是要求出前i个物品放入背包，得出最大价值。

假设已经放入第i个物品，首先要判断的就是第i个物品是放还是不放，放的话不能超过容量，且价值最大，不放就是超过了容量，但前i-1个物品已经是最大价值。

我一开始是是设想让重量的数组随机加（价值也跟着加），只要加到背包的容量就将所加价值存储到另一个数组，可惜后来发现没弄出来。。。。

于是我选择上网学习背包01的题，来弥补自己代码的不足。

之后经过网上的解释我改变了自己的想法，设想背包容量逆序循环（容量从大到小），判断第i个物品能否放入，之所以将背包容量逆序是因为如果背包顺序增加会导致物品重复放入。

例如：

当i = 2，我们要求 f [5]：表示检测物品2放入容量为5的背包的最大收益

上图表示，当i = 2，求f[5]时f数组的状况，

橙色为数组现在存储的值，这些值是i = 2时(本次循环)存入数组 f 的。相当于f[i][v]

这是由于，我们是增序遍历数组f的，在求f[v]时，v之前的值(0 ~ v - 1)都已经在第i次循环中求出。

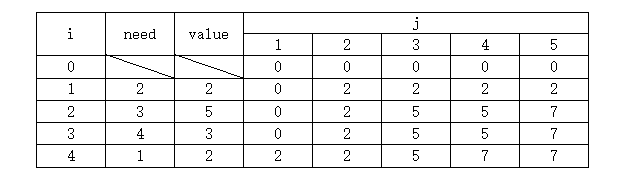
而黄色使我们要求的值，在求f[5]之前，f[5]= 5，即f[i - 1][5] = 5

现在要求 i = 2 时的f[5] = f[5 - 2] + 10 =10+ 10 = 20 >  f[i - 1][5] = 5

故，f[5] = 20；

其中引用的f[3]是相当于f[i][3] 而不是正常的f[i - 1][3]

具体来说就是这幅图：



判断背包容量每种情况下最大价值组合。

心得体会：本次理解背包01题让我深刻意识到二维数组的重要性，虽然实验报告中没提及，但实际上从上方的表格就可以看出此题可以用二维数组表示，通过二维数组优化才得以得到现在的代码。今后我应该更加重视二维数组的应用。

除此之外，我还发现做dp相关的题首先要思考方程式是怎样的，而且不能满足现有的代码，要尝试对代码进行优化。

最后我发现自己对二维数组的运用远远不行，并且思维还不够活跃，需要多多锻炼，多刷杭电。