再谈01背包

- 适用问题：有n个物品，它们有各自的体积和价值，现有给定容量的背包，如何让背包里装入的物品具有最大的价值总和？
- 例题：Bone Collector
  - 对该题样例的分析可看01背包样例演示，看完PPT再看代码效果更佳哦！
  - 定义状态：
    $max\_Value[i][j]:=$ 用容量为j的背包，考虑前i个物品的取舍，所能取得的最大价值和
  - 初始化状态：

    有0个物品可以取时：$max\_Value[0][j]=0$

    背包容量为0时：$max\_Value[i][0]=0$
  - 得出递推式：

    背包容量为j，考虑第$i+1$个物品时，状态为$(i+1,j)$。如果取了这件物品，那么问题就变成了：背包容量为$j-(物品i的重量)$时，从前i个物品做选择，能够取得最大价值和是？则从$(i+1,j)$状态转移到$(i,j-(物品i的重量))$；如果不取这件物品，那完全可以当这件物品不存在，那么问题就变成了：背包容量为j时，从前i个物品做选择，能够取得最大价值和是？则从$(i+1,j)$状态转移到(i,j)。两种做法选择能获取最大价值的那种。

    递推式：
    $max\_Value[i][j]=max(max\_Value[i-1][j-weight[i]]+value[i],max\_Value[i-1][j])$
  - 按递推方向求解：

    一个一个物品*加进去*考虑，i是递增的；对于固定的i，j遍历所有可能的背包容量，j可以是任意顺序的。

```
#include <bits/stdc++.h>
using namespace std;
const int mx = 1002;    // 最多多少物品。依题目数据，该值同时也是最大容量
int max_value[mx][mx] = {0};
int v[mx]; // value
int w[mx]; // weight
int main()
{
    int T, n, v;    // T组数据，n个物品，背包容量最大为v
    scanf("%d", &T);
    while (T--)
    {
        scanf("%d %d", &n, &v);
        for (int i = 1; i <= n; ++i)
            scanf("%d", &v[i]);
        for (int i = 1; i <= n; ++i)
            scanf("%d", &w[i]);
        memset(max_value, 0, sizeof(max_value));    // 用于把整个数组清0

        for (int i = 1; i <= n; ++i){    // 逐个物品加进去考虑
            for (int j = 0; j <= v; ++j){    // 需要在前i个物品做选择时，任意背包容量所对应的最大价值
                if (j >= w[i]){
```

```
                    max_Value[i][j] = max(max_Value[i - 1][j - w[i]] +
V[i], max_Value[i - 1][j]);
                }else{
                    max_Value[i][j] = max_Value[i - 1][j];
                }
            }
        }

            /*
            for (int i = 1; i <= n; ++i){
                for (int j = v; j >= 0; --j){
                    if (j >= w[i]){
                        max_Value[i][j] = max(max_Value[i - 1][j -
w[i]] + V[i], max_Value[i - 1][j]);
                    }else{
                        max_Value[i][j] = max_Value[i - 1][j];
                    }
                }
            }
            */

         printf("%d\n", max_Value[n][v]);
    }
    return 0;
}
```