解决的问题是：规定的总重量下，找到能装下不同物品的组合，使得装起来的价值最高

对于简便版动态规划（由于01背包问题的递推式只与dp矩阵上一行有关，所以可以压缩）

1.01背包不重复，那么每一行的就应该从右往前区更新

（原因：利用上一行的同位置和上一行的左边的位置进行比较，所以如果早更新前面的，那么上一行的值就会被替换了）

Eg.

 for(int j = half\_weight ; j >= 0 ; j--){

                if(j >= stones[i]){

                    dp[j] = (dp[j] > (dp[j-stones[i]] + stones[i]))?dp[j]:(dp[j-stones[i]] + stones[i]);

                }

            }

}

2.01背包可重复，那么每一行就应该从前往后去更新