Robberies http://acm.hdu.edu.cn/showproblem.php?pid=2955   
    背包;第一次做的时候把概率当做背包(放大100000倍化为整数):在此范围内最多能抢多少钱  最脑残的是把总的概率以为是抢N家银行的概率之和… 把状态转移方程写成了f[j]=max{f[j],f[j-q[i].v]+q[i].money}(f[j]表示在概率j之下能抢的大洋);  
      
    正确的方程是:f[j]=max(f[j],f[j-q[i].money]\*q[i].v)  其中,f[j]表示抢j块大洋的最大的逃脱概率,条件是f[j-q[i].money]可达,也就是之前抢劫过;  
    始化为:f[0]=1,其余初始化为-1  (抢0块大洋肯定不被抓嘛)  
      
最大报销额 http://acm.hdu.edu.cn/showproblem.php?pid=1864   
    又一个背包问题,对于每张发票,要么报销,要么不报销,0-1背包,张数即为背包;  
    转移方程:f[j]=max(f[j],f[j-1]+v[i]);  
    恶心地方:有这样的输入数据 3 A:100 A:200 A:300  
      
最大连续子序列 http://acm.hdu.edu.cn/showproblem.php?pid=1231  
    状态方程:sum[i]=max(sum[i-1]+a[i],a[i]);最后从头到尾扫一边  
    也可以写成:  
                Max=a[0];  
                Current=0;  
                for(i=0;i<n;i++)  
                {  
                    if(Current<0)  
                        Current=a[i];  
                    else  
                        Current+=a[i];  
                    if(Current>Max)  
                        Max=Current;  
                }  
      
max sum http://acm.hdu.edu.cn/showproblem.php?pid=1003   
    同上,最大连续子序列      
      
Largest Rectangle http://acm.hdu.edu.cn/showproblem.php?pid=1506  
    对于每一块木板,Area=height[i]\*(j-k+1)  其中,j<=x<=k,height[x]>=height[i];找j,k成为关键,一般方法肯定超时,利用动态规划,如果它左边高度大于等于它本身,那么它左边的左边界一定满足这个性质,再从这个边界的左边迭代下去  
    for(i=1;i<=n;i++)  
        {              
            while(a[l[i]-1]>=a[i])  
                l[i]=l[l[i]-1];  
                  
        }  
      
    for(i=n;i>=1;i--)  
        {  
            while(a[r[i]+1]>=a[i])  
                r[i]=r[r[i]+1];  
        }  
      
City Game http://acm.hdu.edu.cn/showproblem.php?pid=1505  
    1506的加强版,把2维转换化成以每一行底,组成的最大面积;(注意处理连续与间断的情况);  
      
Bone Collector http://acm.hdu.edu.cn/showproblem.php?pid=2602   
    简单0-1背包,状态方程:f[j]=max(f[j],f[j-v[i]]+w[i])  
      
Super Jumping  http://acm.hdu.edu.cn/showproblem.php?pid=1087   
    最大递增子段和,状态方程:sum[j]=max{sum[i]}+a[j]; 其中,0<=i<=j,a[i]<a[j]      
      
命运http://acm.hdu.edu.cn/showproblem.php?pid=2571  
    状态方程:sum[i][j]=max{sum[i-1][j],sum[i][k]}+v[i][j];其中1<=k<=j-1,且k是j的因子      
      
Monkey And Banana     http://acm.hdu.edu.cn/showproblem.php?pid=1069  
    状态方程:f[j]=max{f[i]}+v[j];其中,0<=i<=j,w[i]<w[j],h[i]<h[j]      
      
Big Event in HDU http://acm.hdu.edu.cn/showproblem.php?pid=1171   
    一维背包,逐个考虑每个物品带来的影响,对于第i个物品:if(f[j-v[i]]==0) f[j]=0;  
    其中,j为逆序循环,且j>=v[i]      
      
数塔http://acm.hdu.edu.cn/showproblem.php?pid=2084  
    自底向上:dp[i][j]=max(dp[i+1][j],dp[i+1][j+1])+v[i][j];      
      
免费馅饼http://acm.hdu.edu.cn/showproblem.php?pid=1176  
    简单数塔  
    自底向上计算:dp[i][j]=max(dp[i+1][j-1],dp[i+1][j],dp[i+1][j+1])+v[i][j];处理边界  
      
I Need A Offer http://acm.hdu.edu.cn/showproblem.php?pid=1203  
    简单0-1背包,题目要求的是至少收到一份Offer的最大概率,我们得到得不到的最小概率即可,状态转移方程:f[j]=min(f[j],f[j-v[i]]\*w[i]);其中,w[i]表示得不到的概率,(1-f[j])为花费j元得到Offer的最大概率      
      
FATE http://acm.hdu.edu.cn/showproblem.php?pid=2159   
    二维完全背包,第二层跟第三层的要顺序循环;(0-1背包逆序循环);状态可理解为,在背包属性为 {m(忍耐度), s(杀怪个数)} 里最多能得到的经验值,之前的背包牺牲体积,这个背包牺牲忍耐度跟个数  
    注意: 最后扫的时候 外层循环为忍耐度,内层循环为杀怪个数,因为题目要求出剩余忍耐度最大,没有约束杀怪个数,一旦找到经验加满的即为最优解;  
    状态转移方程为: f[j][k]=max(f[j][k],f[j-v[i]][k-1]+w[i]); w[i]表示杀死第i个怪所得的经验值,v[i]表示消耗的忍耐度  
      
How To Type http://acm.hdu.edu.cn/showproblem.php?pid=2577       
    用两个a,b数组分别记录Caps Lock开与关时打印第i个字母的最少操作步骤;  
    而对于第i个字母的大小写还要分开讨论:  
    Ch[i]为小写: a[i]=min(a[i-1]+1,b[i-1]+2);不开灯直接字母,开灯则先关灯再按字母,最后保持不开灯;    b[i]=min(a[i-1]+2,b[i-1]+2);不开灯则先按字母再开灯,开灯则Shift+字母(比关灯,按字母再开灯节省步数),最后保持开灯;  
    Ch[i]为大写: a[i]=min(a[i-1]+2,b[i-1]+2); b[i]=min(a[i-1]+2,b[i-1]+1)  
      
    最后,b[len-1]++,关灯嘛O(∩\_∩)O~       
      
Coins http://acm.hdu.edu.cn/showproblem.php?pid=2844  
    类似于HDU1171 Big Event In HDU,一维DP,可达可不达      
      
Beans http://acm.hdu.edu.cn/showproblem.php?pid=2845   
    横竖分别求一下不连续的最大子段和;  
    状态方程: Sum[i]=max(sum[j])+a[i];其中,0<=j<i-1;      
      
Largest Submatrix http://acm.hdu.edu.cn/showproblem.php?pid=2870   
    枚举a,b,c 最大完全子矩阵,类似于HDU1505 1506      
      
Matrix Swapping II http://acm.hdu.edu.cn/showproblem.php?pid=2830   
最大完全子矩阵,以第i行为底,可以构成的最大矩阵,因为该题可以任意移动列,所以只要大于等于height[i]的都可以移动到一起,求出height>=height[i]的个数即可,这里用hash+滚动,先求出height[i]出现的次数,然后逆序扫一遍hash[i]+=hash[i+1];      
      
最少拦截系统http://acm.hdu.edu.cn/showproblem.php?pid=1257  
    两种做法,一是贪心,从后往前贪;二是DP;  
    if(v[i]>max{dp[j]})  (0<=j<len)  
    dp[len++]=v[i];      
      
Common Subsequence http://acm.hdu.edu.cn/showproblem.php?pid=1159   
    经典DP,最长公共子序列  
    Len[i][j]={len[i-1][j-1]+1,(a[i]==b[j]); max(len[i-1][j],len[i][j-1])}  
    初始化的优化:   
    for(i=0;i<a;i++)  
            for(j=0;j<b;j++)  
                len[i][j]=0;  
        for(i=1;i<=a;i++)   
            for(j=1;j<=b;j++)   
                if(ch1[i-1]==ch2[j-1])   
                    len[i][j]=len[i-1][j-1]+1;  
                else   
                    len[i][j]=max(len[i-1][j],len[i][j-1]);      
      
★ 搬寝室http://acm.hdu.edu.cn/showproblem.php?pid=1421   
    状态Dp[i][j]为前i件物品选j对的最优解  
    当i=j\*2时,只有一种选择即 Dp[i-2][j-1]+(w[i]-w[i-1])^2  
    当i>j\*2时,Dp[i][j] = min(Dp[i-1][j],Dp[i-2][j-1]+(w[j]-w[j-1])^2)      
      
★ Humble Numbers http://acm.hdu.edu.cn/showproblem.php?pid=1058   
    如果一个数是Humble Number,那么它的2倍,3倍,5倍,7倍仍然是Humble Number  
    定义F[i]为第i个Humble Number  
    F[n]=min(2\*f[i],3\*f[j],5\*f[k],7\*f[L]), i,j,k,L在被选择后相互移动  
    (通过此题理解到数组有序特性)      
      
★ Doing Homework Again http://acm.hdu.edu.cn/showproblem.php?pid=1789   
    这题为贪心,经典题;  
    切题角度,对于每个任务要么在截至日期前完成要么被扣分;所以考虑每个人物的完成情况即可;由于每天只能完成一个任务,所以优先考虑分值较大的任务,看看该任务能不能完成,只要能完成,即使提前完成,占了其他任务的完成日期也没关系,因为当前任务的分值最大嘛,而对于能完成的任务能拖多久就拖多久,以便腾出更多时间完成其他任务;      
      
How Many Ways http://acm.hdu.edu.cn/showproblem.php?pid=1978   
    两种D法,一是对于当前的点,那些点可达;二是当前点可达那些点;  
    明显第二种方法高,因为第一种方法有一些没必要的尝试;  
    Dp[i][j]+=Dp[ii][jj]; (map[ii][jj]>=两点的曼哈顿距离)  
    值得优化的地方,每两点的曼哈顿距离可能不止求一次,所以预处理一下直接读取      
      
珍惜现在 感恩生活http://acm.hdu.edu.cn/showproblem.php?pid=2191   
    每个物品最多可取n件,多重背包;  
    利用二进制思想,把每种物品转化为几件物品,然后就成为了0-1背包      
      
Piggy-Bank http://acm.hdu.edu.cn/showproblem.php?pid=1114   
    完全背包;常规背包是求最大值,这题求最小值;  
    只需要修改一下初始化,f[0]=0,其他赋值为+∞即可;  
    状态转移方程:f[i][V]=max{f[i-1][V],f[i-1][V-k\*v[i]]+k\*w[i]},其中0<=k\*v[i]<=V  
      
★ Max Sum Plus Plus http://acm.hdu.edu.cn/showproblem.php?pid=1024  
    1. 对于前n个数, 以v[n]为底取m段:   
    当n==m时,Sum[m][n]=Sum[m-1][n-1]+v[n],第n个数独立成段;  
当n>m时, Sum[m][n]=max{Sum[m-1][k],Sum[m][n-1]}+v[n]; 其中,m-1<=k<j,解释为,v[n]要么加在Sum[m][n-1],段数不变,要么独立成段接在前n-1个数取m-1段所能构成的最大值后面  
2. 空间的优化:  
        通过状态方程可以看出,取m段时,只与取m-1段有关,所以用滚动数组来节省空间  
      
FatMouse’s Speed http://acm.hdu.edu.cn/showproblem.php?pid=1160   
    要求:体重严格递增,速度严格递减,原始顺序不定  
    按体重或者速度排序,即顺数固定后转化为最长上升子序列问题  
    Dp[i]表示为以第i项为底构成的最长子序列,Dp[i]=max(dp[j])+1,其中0<=j<i , w[i]>w[j]&&s[i]<s[j] 用一个index数组构造最优解:记录每一项接在哪一项后面,最后用max找出最大的dp[0…n],dex记录下标,回溯输出即可      
      
Cstructing Roads http://acm.hdu.edu.cn/showproblem.php?pid=1025   
    以p或者r按升序排列以后,问题转化为最长上升子序列  
    题目数据量比较大,只能采取二分查找,n\*log(n)的算法  
用一个数组记录dp[]记录最长的子序列,len表示长度,如果a[i]>dp[len], 则接在后面,len++; 否则在dp[]中找到最大的j,满足dp[j]<a[i],把a[i]接在dp[j]后面;      
      
FatMouse Chees http://acm.hdu.edu.cn/showproblem.php?pid=1078   
    Dp思想,用记忆化搜索;简单题,处理好边界;      
      
To the Max http://acm.hdu.edu.cn/showproblem.php?pid=1081  
    最大子矩阵  
    把多维转化为一维的最大连续子序列;(HDU1003)      
      
龟兔赛跑http://acm.hdu.edu.cn/showproblem.php?pid=2059   
未总结      
      
★ Employment Planning http://acm.hdu.edu.cn/showproblem.php?pid=1158   
    状态表示:    Dp[i][j]为前i个月的留j个人的最优解;Num[i]<=j<=Max{Num[i]};  
                j>Max{Num[i]}之后无意义,无谓的浪费 记Max\_n=Max{Num[i]};  
    Dp[i-1]中的每一项都可能影响到Dp[i],即使Num[i-1]<<Num[i]  
    所以利用Dp[i-1]中的所有项去求Dp[i];  
    对于Num[i]<=k<=Max\_n,    当k<j时, 招聘;  
                            当k>j时, 解雇  然后求出最小值  
    Dp[i][j]=min{Dp[i-1][k…Max\_n]+(招聘,解雇,工资);      
      
Dividing http://acm.hdu.edu.cn/showproblem.php?pid=1059   
    一维Dp  Sum为偶数的时候判断Dp[sum/2]可不可达      
      
Human Gene Factions http://acm.hdu.edu.cn/showproblem.php?pid=1080   
状态转移方程:  
f[i][j]=Max(f[i-1][j-1]+r[a[i]][b[j]], f[i][j-1]+r[‘-‘][b[j]],f[i-1][j]+r[a[i]][‘-‘]);  
  
★ Doing Homework http://acm.hdu.edu.cn/showproblem.php?pid=1074   
    这题用到位压缩;  
    那么任务所有的状态有2^n-1种  
    状态方程为:Dp[next]=min{Dp[k]+i的罚时} 其中,next=k+(1<<i),k要取完满足条件的值 k>>i的奇偶性决定状态k  
具体实现为: 对每种状态遍历n项任务,如果第i项没有完成,则计算出Dp[next]的最优解      
      
Free DIY Tour http://acm.hdu.edu.cn/showproblem.php?pid=1224   
    简单的数塔Dp,考察的是细节的处理;  
    Dp[i]=Max{Dp[j]}+v[i]  其中j->i为通路;  
    v[n+1]有没有初始化,Dp数组有没有初始化  
    这题不能用想当然的”最长路”来解决,这好像是个NP问题 解决不了的  
      
      
重温世界杯http://acm.hdu.edu.cn/showproblem.php?pid=1422   
这题的状态不难理解,状态表示为,如果上一个城市剩下的钱不为负,也就是没有被赶回杭电,则再考虑它对下一个城市的影响;如果上一个城市剩下的前加上当前城市的前大于当前城市的生活费,那么Dp[i]=Dp[i-1]+1;  
值得注意的而是这题的数据为100000;不可能以每个城市为起点来一次Dp,时间复杂度为n^2;足已超时;  
我是这样处理的,在保存的数据后面再接上1…n的数据,这样扫描一遍的复杂度为n;再加一个优化,当Dp[i]==n时,也就是能全部游完所有城市的时候,直接break;  
  
Pearls http://acm.hdu.edu.cn/showproblem.php?pid=1300   
    Dp[i]=min{Dp[j]+V},  0<=j<i, V为第j+1类珠宝到第i类全部以i类买入的价值;      
      
Zipper http://acm.hdu.edu.cn/showproblem.php?pid=1501  
    Dp[i][j]=       
      
★Fast Food http://acm.hdu.edu.cn/showproblem.php?pid=1227  
    这里需要一个常识:在i到j取一点使它到区间每一点的距离之和最小,这一点为(i+j)/2用图形即可证明;  
    Dp[i][j]=max{Dp[i-1][k]+cost[k+1][j]  其中,(i-1)<=k<j状态为前j个position建i个depots      
      
Warcraft http://acm.hdu.edu.cn/showproblem.php?pid=3008  
    比赛的时候这道DP卡到我网络中心停电!!! 卧槽~   
    因为你没有回血效应,所以你挂掉的时间是一定的;  
    用Dp[i][j]表示第i秒剩余j个单位的MP时怪物所剩的血量; 注意必须是剩余,也就是说,初始化的时候,DP[0][100]=100;  其他Dp[0]状态都不合法,因为没有开战的时候你的MP是满的  
    状态转移方程为:  
    Dp[i+1][j-sk[k].mp+x]=min(Dp[i+1][j-sk[k].mp+x],Dp[i][j]+sk[k].at; 释放第K种技能,物理攻击可以看成是at=1,mp=0 的魔法;  
      
Regular Words http://acm.hdu.edu.cn/showproblem.php?pid=1502   
    F[a][b][c]=F[a-1][b][c]+F[a][b-1][c]+F[a][b][c-1];  
    a>=b>=c;      
      
Advanced Fruits http://acm.hdu.edu.cn/showproblem.php?pid=1503   
    最长公共子序列的加强版