Winter Camp Test01　Solution

P2652 - 简单的划分数列

分析：经典的最小 k 段问题

30 分，n <= 30, k <= 10

搜索，暴力枚举数列的 k 段，状态数近似于 K^K，配合一些最优性剪枝和可行性剪枝可通过此部分数据。

50 分（n , k<=100）

DP，最小 k 段问题是比较经典动态规划问题，状态转移方程：

state[i][j] = min{ max{ state[i][1]-state[k][1] , state[k][j-1] } }

这里 state[i][j]表示前 i 个数据分成 j 段得最大最小值，state[i][1]表示前 i 个数据分成 1 段的值，这里其实就是前 i 个数据之和。

1 2.....k.....i 这里可以这样理解，前 k 个数据的 j-1 分段的值与 k----i 的值的最大值，其中 k----i 的值就是上面 state[i][1]-state[k][1]的值。

100 分 ， n <= 100000, k <= n

二分答案，在区间 max{ai}~sum{ai}之间二分每段元素和的最大值，对于每个二分得到的值 mid 进行 check，检查每段元素和的最大若为 mid 能否将数列划分成 <=k 段，若能说明 mid 可行继续在左边区间二分，否则在右边区间二分。

P1101 - 软件开发

solution①

爆搜 n 个人同时开工，在 2*m 个模块上工作，由于软件模块完成的先后次序没有要求，实际上可以认为是 n 个人同时分布在两条流水线上加工，爆搜代码怎么实现？？？ 不会....

有错误的贪心的做法吗？ 不会...

DP：

dp[i][j][k]表示前 i 个人，第一种软件生产了 j 个，第二种生产了 k 个，花的最少时间。转移方程为：dp[i][j][k]=min(max(dp[i-1][x][y],(j-x)*d1+(k-y)*d2))。

第一层循环枚举每一个人，二、三层循环枚举第一、二个软件已经做了多少模块，第四、五层循环枚举第 i 个人第一、二个软件做了多少模块。这个方程转移需要的时间复杂度是 O(n^5)，通过不了 n=100 的极限数据。

如何才能减少循环呢？如果要是知道时间就好了，这样只要枚举第一个软件完成的模块就可以计算出第二个模块完成了多少。这道题需要求最短时间，答案具有单调性，可以先二分答案，再通过 DP 来验证答案的正确性。

最大值最小模型，二分答案+DP 验证

首先这题其实满足二分性质，如果第 i 天完成了，那么第 i+1 天必然可以完成，而 i-1 天则不一定能完成。求所有人完成时间最长的最小值-->最大/小 化 最/大小值-->二分答案

如何验证在时间 mid 的范围之内是否能完成工作？

我们发现由于同一个人分配在 1、2 上面的时间不同，会对结果产生影响，所以我们要解决两个工作之间的干扰问题

由于 n,m 不大，试着枚举前 i 个人在总共做了 j 项工作 1 时，剩下还可以做工作 2 份数的最大值，设为 dp[i][j]

枚举第 i 个人做了 k 份工作 i，则状态转移方程为：

dp[i][j]=max{ dp[i][j-k] + (lim-a[i]*k) / b[i] }

最后如果 dp[n][m]<m，则不能完成，反之则可以;

时间复杂度:O(n*m*m*log2(ans))

***需要注意的是状态的处理，初值要赋成-1，表示这个状态没有产生。再把 dp[0,0] 赋成 0，表示这一状态合法。如果用来更新的状态没有产生就不能用这个状态来更新当前状态。

P1497 - 【[国家集训队 2011】跳跳棋

Hzwer's solution ~ http://hzwer.com/4784.html

首先广搜 20 分

对于一个状态

例如 2 3 7

中间可以往两侧跳，即 2 3 7->1 2 7 / 2 3 7->2 7 11

两侧仅有一个能往中间跳，即 2 3 7->3 4 7

那么所有的状态就能表示为一棵二叉树，第一种情况为其两个儿子，第二种为其父亲

问题转换为给定树上的两个结点，求其距离

直接暴力可以得 40 分

但是可以构造这样的数据

1 2 1000000000

99999998 99999999 1000000000

这样左边要一直往中间跳上上亿次

发现若记前两个数差 t1，后两个数差 t2，不妨设 t1<t2

则左边最多往中间跳(t2-1)/t1 次

然后只能右边往中间跳，是一个辗转相除的过程，即在 logK 的时间内我们可以用这种方法得到某个结点它向上 K 次后的结点，或者根节点，同时还可以顺便算下深度

那么只要求始终两个状态的深度 d1,d2,将较深的调整到同一深度

然后二分/倍增求与 lca 的深度差 x

ans=2*x+abs(d1-d2)

solution②

设：s1=b-a s2=c-b

那么 b 可以跳动到 a 左边，或者 c 右边。

如果 s1 < s2，那么 a 可以跳到 bc 中间；

如果 s1 > s2，那么 c 可以跳到 ab 中间。

也就是，如果 s1≠s2，那么一个局面有 3 种跳法。

如果 s1=s2，那么只有 2 种跳法。

如果我们用图来表示状态之间的关系，就很容易发现，状态之间组成的联系实际上是二叉树组成的森林。

每一个 s1=s2 的状态（注意，它们的位置可能不同）都是一棵二叉树的根。

其余的每个状态，a 或 c 往中间跳表示往父亲节点走一步

对于所有状态，中间节点往左右跳分别对应往左右孩子走一步。

原问题转换成了树上最短路问题。我们设起始状态对应节点 p，目标状态对应节点 q。那么问题是：1.p 和 q 是否同根。2.如果同根，求 p 到 q 的距离。

这两个问题都可以用 LCA 的知识来解决。（LCA 最近公共祖先）

如果 p 和 q 不存在 LCA 那么输出 NO。

如果存在，那么计算 LCA(p,q)到 p 和 q 分别的距离，相加即为答案。

当我们在把某状态往父亲走时，每走一步，a 或 c 就会往 b 的方向跳一次。这其实相当于 a 或 c 去了 b 的位置，而 b 向右跳了 s1 格或向左跳了 s2 格。跳完之后，s2 就会递减一个 s1（或 s1 递减一个 s2）。

于是进一步思考，可不可以一次走很多步？？？

可以的，简单的数学运算就能解决问题。

事实上我们可以在 O(logS)的时间内做到把某状态往父亲走任意多步。

（相当于 s1 和 s2 辗转相除，已证明此操作复杂度为对数阶）

通过欧几里得算法，我们可以直接计算 p 和 q 在二叉树中的深度。

为了方便求 LCA，我们首先把 p 和 q 深度调整到相同。h(x)表示 x 的深度。

不妨设 h(p)≤h(q)。我们把 q 往上走 h(q)-h(p)步。

求 2 个深度相同的点的 LCA，我们可以采用二分答案的方法。

对于二分答案：LCA 到 p 的距离 mid，

如果 p 往上走 mid 和 q 往上走 mid 到达的点相同， **(倍增 LCA 思想)**

那么 答案≤mid；

否则 答案＞mid。

时间复杂度：O((logS)^2)。