Contest 15 By Digimon

Summary

Problem	AC	WA	PE	RTE	FPE	SF	NZEC	TLE	MLE	OLE	CE	Submit
A	0	0	0	0	0	0	0	0	0	0	0	0
B	1	0	0	0	0	0	0	0	0	0	0	1
C	5	10	0	0	0	5	0	4	0	0	0	24
D	19	95	0	0	0	2	0	11	0	0	0	127
E	3	30	0	0	0	0	0	16	0	0	0	49
F	3	1	0	0	0	0	0	0	0	0	0	4
Summary	31	136	0	0	0	7	0	31	0	0	0	205

Problem A: Left 4 Dead

- 给出某一关的总路程，难度系数，受伤害的地点，捡药的地点，求最大总分
- 每个地点至多受一次伤害和捡一种药
- 总分 = 路程分 + 奖励分
- 路程分 = ∑走过的路的长度 * 100 / 总路程，例如总路程为100，两个都到了终点，路程分为200。一个人死在了 50 的地方，另一个人死在了 80 的地方，路程分为 50 + 80 = 130

Problem A: Left 4 Dead

- 奖励分 = 地图难度 * 总血量 * 存活人数
- 总血量为两个人到达终点后吃完所有药的血量和
- 两个人的初始血量为 100，上限为 200，一开始身上没有药，每个人至多带一片药片和一根注射剂
- 一片药片 +50 hp，一根注射剂 +25 hp，血量超过 100 时，每走一米少一滴血，血量为 0 时死亡

Problem A: Left 4 Dead

- 数据不大，可以 dp
- 状态很好表示，dp[i][x][y][k]表示到了在第i个地点前，第一个人血剩 x，第二个人血剩 y，药的状态为 k 时的最大路程分。k 为一个两位的 3 进制数，表示药片和注射剂的数量
- 写起来非常麻烦，如果跳了这题，基本上GG了

Problem A: Left 4 Dead

- 先根据地点顺序排个序，同个地点药排在前面
- 第i个地点为药，那么有两种策略：
- 1.吃药 dp[i][x][y][k] -> dp[i+1][nx][y][k]
- dp[i][x][y][k] -> dp[i+1][x][ny][k]
- 2.储存起来 dp[i][x][y][k] -> dp[i+1][x][y][nk]
- 注意 x 或者 y 为 0 的时候，说明他已经死了，也就不能吃药了
- 药也不能超过限制

Problem A: Left 4 Dead

- 第i个地点为伤害，先枚举吃药情况，再算伤害，这时有两种情况：
- 1.有一个人活着
  - 他死了　　dp[i][0][y][k] -> dp[i+1][0][0][0]
  - 他还活着　　dp[i][0][y][k] -> dp[i+1][0][ny][nk]
- 2.两个都活着
  - 一个人死了　　dp[i][x][y][k] -> dp[i+1][0][ny][nk]
  - 两个都活着　　dp[i][x][y][k] -> dp[i+1][nx][ny][nk]
- 注意两个人都活着，但是受到伤害死了一个人后，这时药最多只能携带一片药和一根注射剂

Problem A: Left 4 Dead

- 到第 i 点，如果目前的血量有超过 100 的，先计算从第 i – 1 个点到第 i 个点后的血量，然后再进行讨论
- 对所有的 dp[n+m+1][x][y][k] 计算他们的奖励分，枚举 k 的分布情况，然后算出总血量

Problem B: Sister’s Noise

- 给出一个可选的字符表，每种字符 Ci 在表中出现了 Ti 次
- 问，使用这些字符的全部或部分，可以组成的长度大于等于 L 的所有字符串中第 M 大的字符串是什么

Problem B: Sister’s Noise

- 先不管长度大于等于 L 的条件
- 枚举第一个字符，按从大到小顺序
- 计算剩下的字符能构成的不同字符串总数 cnt
  - 如果 M > cnt，说明当前枚举的字符不够小，于是将 M -= cnt 后枚举更小的字符
  - 如果 M <= cnt，说明第一个字符就是这个字符。将这个字符固定下来，然后类似地枚举第二个、第三个字符即可
- 怎么计算 cnt ？

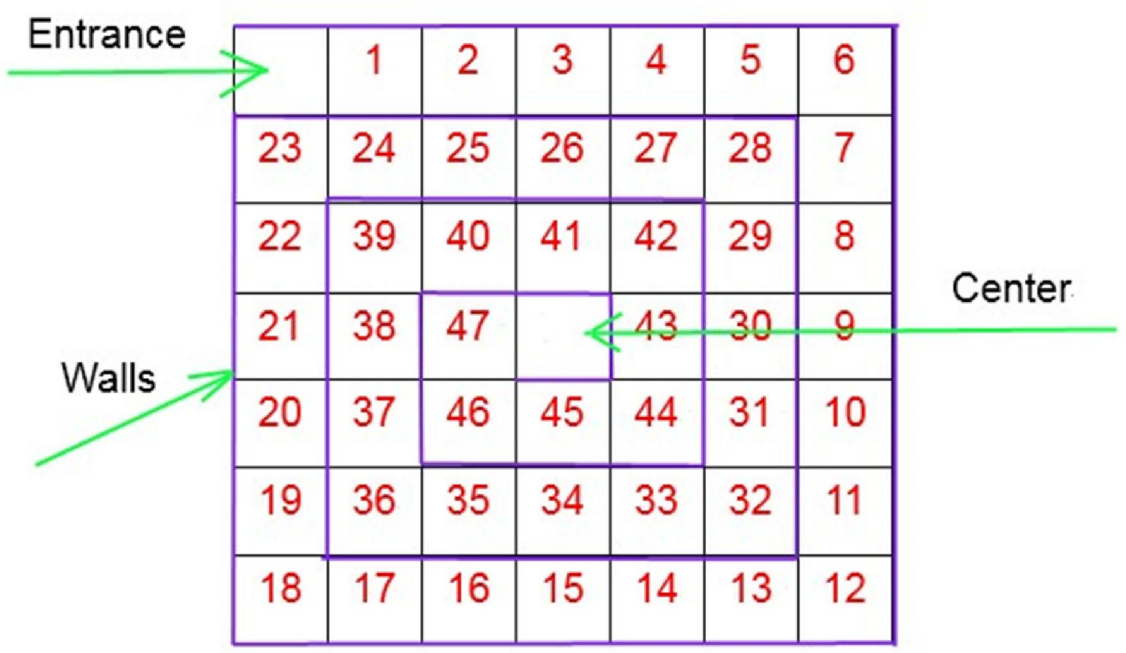
Problem B: Sister’s Noise

- dp[i][j] 表示处理完前 i 种字符，产生的字符串长度为 j 时的方案总数
- dp[i][j] 可以转移到 dp[i + 1][j + k]，其中 k = 0 ~ 当前字符的总个数
- dp[i + 1][j + k] += dp[i][j] * C(j + k, k)
- 放入 k 个字符后，长度将是 j + k。为了放这 k 个字符，必然需要在 j + k 的位置中为它们选择 k 个位置
- 最后，有了 dp[i][j]，处理长度 >= L 这个问题就很简单了

Problem C: Only My Railgun

- 给定一个7*7的格子图，里面有墙，格子内有炸弹。
- 炮姐的输出功率是会随时间变化，周期为 T。同时她可以蓄电最多 K 秒，来发射电磁炮，其威力 P 等于这 K 秒内功率之和。并且一发电磁炮能引爆一条直线上至多 P 个炸弹。
- 射出一发电磁炮后，炮姐输出功率变化重置。即回到周期最开始的功率。
- 炮姐可以原地等待一个最佳的时机发射电磁炮。
- 当一个区域内的炸弹被清空后，炮姐可以迅速通过该区域（移动不消耗时间）
- 问由起点到达终点的最短时间。

Problem C: Only My Railgun



Problem C: Only My Railgun

- 该题应该算是一道简单DP题。
- 因为所有数据都是小于1000，所以随便怎么搞都能过。
- 首先暴力 DFS 或直接 for 把格子图转换成 12 条直线上的炸弹数之和。
- 然后预处理，a[i] 表示用掉 i 秒能打出电磁炮的单次最高输出。
- dp[i] 表示用掉 i 秒能打出的最高总输出。
- 状态转移方程：dp[i] = max{dp[i - k] + a[k]} (1 <= k <= i)
- 之后再枚举清空每条直线所用的最短时间，其和即为答案。

Problem D: Eternal Reality

- 你的等级是 L，要参加一场比赛，每个环节需要 >= Ai 的等级才能得分
- 如果使用幻想御手 (Level Upper)，则可以在接下来连续 X 个环节中将等级增加到 min(L + 1, 5)，然后在之后的 Y 个环节中将等级降低为 0，最后恢复正常并可再次使用
- 没错，等级增加到 min(L + 1, 5)，不是直接 L + 1，注意读题即可
- 比赛心态很重要，别错了就乱交

Problem E: The Lambs

- 给出 N 个木桩，M 只羊，在木桩上绕绳子使得羊被围住且绳子长度最短
- 做法：
- 只对这样的两个点建有向边：
- 对于木桩 A -> B，所有羊都在 A -> B 的左侧
- 此时，从某一点出发 X，如果能走一段路径回到 X，则说明羊被围住了 (由于羊在所有线段的左侧，意味着羊一定只可能在多边形内)
- 题目就转化成了最小环问题，Floyd 求解即可

Problem F: Miko Miko Suika

- 给定一个平面，上面有N个既不覆盖原点也相互不相交的凸多边形，以及M个在多边形之外的收益点。现在可以：
  - 承担一定代价清除一些多边形
  - 若原点到某个收益点没被凸多边形遮挡，则可以获得一定收益
  - 若多边形上所有点到原点的线段都被其他凸多边形覆盖，则不能对它进行清除
- 问在这些条件下的最大收益 (连接点产生的收益 - 清除多边形承担的代价)。
- 输入的坐标都是实数，样例很良心地又提醒了一次。

Problem F: Miko Miko Suika

- 图上有正权点 (items)，又有负权点 (Kedamas)；要选择一个正权点，必须先选择它所依赖的一些负权点。
- 这正是最大权闭合图的模型。
- 源点与正权点连边，容量为 B_i
- 负权点与汇点连边，容量为 A_i
- 若某个正权点与一个负权点有依赖关系，则从正权点往负权点连一条容量为 ∞ 的边
- 最终答案为 $\sum Bi \ - \ Maxflow$

Problem F: Miko Miko Suika

- 要判断点与凸多边形是否存在依赖关系，只需判断该点到原点的线段L与某个凸多边形是否相交。
- 做法有很多，一种比较简单的做法就是把凸多边形的边一条条拆出来，判一下它是否与 L 相交。