1. Competition_2017_4_9

1.1. Item1

1.1.1. 描述

There are N queens in an infinite chessboard. We say two queens may attack each other if they are in the same vertical line, horizontal line or diagonal line even if there are other queens sitting between them.

Now given the positions of the queens, find out how many pairs may attack each other?

1.1.2. 输入

The first line contains an integer N.

Then N lines follow. Each line contains 2 integers Ri and Ci indicating there is a queen in the Ri-th row and Ci-th column.

No two queens share the same position.

For 80% of the data, 1 <= N <= 1000

For 100% of the data, 1 <= N <= 100000, 0 <= Ri, Ci <= 1000000000

1.1.3. 输出

One integer, the number of pairs may attack each other.

1.1.4. 解析

1、逐步统计在各行，各列，各对角线上的 Queen 数量，更新数量的同时计算 attach 次数

2、注意点，对角线的公式可以设为 z=ax+by+c，然后带入三个点求出 abc 即可

1.2. Item2

1.2.1. 描述

There are N jobs to be finished. It takes a robot 1 hour to finish one job.

At the beginning you have only one robot. Luckily a robot may build more robots identical to itself. It takes a robot Q hours to build another robot.

So what is the minimum number of hours to finish N jobs?

Note two or more robots working on the same job or building the same robot won't accelerate the progress.

1.2.2. 输入

The first line contains 2 integers, N and Q.

For 70% of the data, 1 <= N <= 1000000

For 100% of the data, 1 <= N <= 1000000000000, 1 <= Q <= 1000

1.2.3. 输出

The minimum number of hours.

1.2.4. 解析

1、明确以下三点

1) 在完成所有 job 的时刻，不可能有 robot 正在复制
2) robot 要么全部复制要么全部工作，即不存在部分工作部分复制
3) 一定存在某个时间点 i，前 i 个时间一直在复制，之后一直工作

2、何时选择复制

- 假设现在有 1 个 robot，有 N 个 job，如果不复制那么需要的时间为 N
- 如果复制，那么 Q 时间后有 2 个 robot，可以做 job 的时间为 N -Q
- 必须满足(N-Q)*2>=N*1，即 N>=2Q

3、伪代码 1

```
for(int totalTime=0;;totalTime++)
    for(int copyNum=0;copyNum*Q<totalTime;copyNum++)
        workTime=totalTime-copyNum*Q
        finishedJob=workTime*2^copyNum
        if(finishedJob>=N) return totalTime
```

4、伪代码 2

```
gain=n
time=0
while (gain/2>q)
    robot*=2
    gain/=2
    time++
workTime=ceil(n/robot)
return time*Q+workTime
```

- 机器人复制一次时间代价是 q，复制前工作时间是 N，复制后工作时间是 N/2(注意只算工作时间)，有 N-N/2=N/2 的时间增益
- 如果 N/2>q 说明复制有增益，每次迭代保留当前所需的工作时间

1.3. Item3

1.3.1. 描述

Little Hi has a box which consists of 2xN cells as illustrated below.

```
+----+----+----+----+----+----+
| A1 | A2 | A3 | A4 | .. | AN |
+----+----+----+----+----+----+
| B1 | B2 | B3 | B4 | .. | BN |
+----+----+----+----+----+----+
```

There are some coins in each cell. For the first row the amounts of coins are A1, A2, ... AN and for the second row the amounts are B1, B2, ... BN.

Each second Little Hi can pick one coin from a cell and put it into an adjacent cell. (Two cells are adjacent if they share a common side. For example, A1 and A2 are adjacent; A1 and B1 are adjacent; A1 and B2 are not adjacent.)

Now Little Hi wants that each cell has equal number of coins by moving the coins. He wants to know the minimum number of seconds he needs to accomplish it.

1.3.2. 输入

The first line contains an integer, N. 2 <= N <= 100000

Then follows N lines. Each line contains 2 integers Ai and Bi. (0 <= Ai, Bi <= 2000000000)

It is guaranteed that the total amount of coins in the box is no more than 2000000000 and is a multiple of 2N.

1.3.3. 输出

The minimum number of seconds.

1.3.4. 分析

1、采用贪心算法，优先从上下位置拿取，再不够再从右边拿或者放

2、贪心过程中可能将右边位置的数置为负数

3、**以下求法是错误的**

A1	A2	A3	A4	..	AN
B1	B2	B3	B4	..	BN

- 假设 A1 不足，然后 A1 从 A2 拿，只拿 A2 有的，若仍然不足，向 B2 拿，若，若仍然不足向 A3 拿…以此类推
- 给一个反例

1	2	15
2	1	15

  - A1 从 A2 拿两个，从 B2 拿 1 个，从 A3 拿 2 个
  - B1 从 B3 拿 4 个
  - A2 从 A3 拿 6 个
  - B2 从 B3 拿 6 个
  - A3 从 B3 拿 1 个
  - 红色的两处会导致额外操作，原本 A1 只需要向 A2 借，但是 A1 从 B2 借了一个(多了一次垂直操作)，这会导致 B3 又要从 A3 那里拿一个(又一次垂直操作)，因此额外的操作数是 2

1.3.5. 类似题目

1、给出一列字符串，只包含 AB 两个字符，求把 AB 分开的最少交换次数

1.4. Item4

1.4.1. 描述

In a video game, Little Hi is going to assassinate the leader of an evil group, EL SUENO.

There are N high-value targets in the group, numbered from 1 to N. Each target has another target as his direct superior except for EL SUENO who has no superior. So the superior-subordinate hierarchy forms a tree-like structure. EL SUENO is at the root.

The cost of assassinating the i-th target is Ci. Before assassinating a target Little Hi has to obtain enough information about him. For the i-th target, Little Hi needs INi units of information. By assassinating a target Little Hi will obtain some information about the target's direct superior. More specifically, the i-th target has IPi units of information about his superior. So in order to assassinate EL SUENO, Little Hi needs to assassinate some of his direct subordinates so that the sum of information obtained is enough; assassinating the subordinates needs to assassinate their direct subordinates ... until it reaches some targets require zero information in advance. (Luckily if a target has no subordinate he always requires zero information.)

How Little Hi wants to know what is the minimum cost for successful assassinating EL SUENO.

1.4.2. 输入

The first line contains an integer N.

Then follow N lines. Each line describes a target with 4 integers, Fi, INi, IPi, Ci.

Fi is i-th target's superior. For EL SUENO his Fi is zero.

INi is the amount of information needed to assassinate the i-th target. For a target has no subordinates his INi is always zero.

IPi is the amount of information the i-th target has about his superior Fi.

Ci is the cost of assassinating the i-th target.

For 30% of the data, 1 <= N <= 10, 0 <= INi, IPi <= 100

For 60% of the data, 1 <= N <= 100, 0 <= INi, IPi <= 1000

For 100% of the data, 1 <= N <= 2000, 0 <= Fi <= N, 0 <= INi, IPi <= 20000, 1 <= Ci <= 1000.

It is guaranteed that the N targets form a tree structure.

1.4.3. 输出

The minimum cost. If Little Hi is not able to assassinate EL SUENO output a number

-1

1.4.4. 分析

1、树形 DP

2、刺杀 i 节点需要花费 Ci，而且必须获取 Fi 的信息量，这两部分是独立的。因此问题分解为刺杀根节点的花费(就是个常数)，以及为了刺杀根节点而得到足够信息的过程中的花费，记为刺杀 i 节点获取足够信息量的最少花费为 Cost(i)

3、Cost 的求解：

- 由于一个节点可能有若干个孩子，问题转化为，得到给定信息量的最少花费，该问题可用背包问题求解，dp[i][j]：代表前 i 个孩子，取得 j 个信息量的最少花费
- dp[i][j]=min(dp[i-1][j],Cost[i]+C[i]+dp[i-1][max(0,j-IP[i])]，其中 dp[i][0]=0
  - 即选择杀第 i 个孩子或不杀第 i 个孩子，杀了第 i 个孩子可以得到 IP[i]的信息量，于是在前 i-1 个节点中，只需要再获得 j-IP[i]个信息量即可(若 j-IP[i]<0，那么只需要获得 0 个信息量即可)

1.4.5. 再探 0-1 背包问题

1、给定 N 个物品，第 i 个物品重 weights[i]，价值 values[i]。给定容量 capacity，求最大收益

2、伪代码 1

```
dp[0...N][0...capacity]初始化为 0
for i=1 to N
    for v=1 to capacity
        if v<weights[i]
            dp[i][v]=dp[i-1][v]//当前容量装不下第 i 件物品
        else
            dp[i][v]=Math.max(dp[i-1][v],dp[i-1][v-weights[i]]+values[i]
return dp[N][capacity]
```

- 两层循环可以交换

3、伪代码 2

```
dp[0...capacity]初始化为 0
for i=1 to N
    for v=capacity downto 1
        if v<weights[i] break;
        dp[v]=max(dp[v],dp[v-weights[i]]+values[i]
```

- 第二层循环必须从大到小，因为对于第 i 件物品来说，不能重复拿两次，如果从小到大循环，可能 dp[v-weights[i]]是包含第 i 件物品的收益
- 里外两层循环不能交换，因为每件物品不能重复获取
- 当 i=1 时，如果最终结果包含获取了第一件物品，那么 dp[capacity-weights[1]]好像尚未计算，确实如此，但是后续的循环(i>1)会逐步将 dp[capacity-weights[1]]置为最优，最优时会反过来调用剩余部分(i=N 必然能保证最优)

1. Week145

1.1. 题目

1.1.1. 描述

小 Hi、小 Ho 还有被小 Hi 强拉来的小 Z，准备组队参加一个智力竞赛。竞赛采用过关制，共计 N 个关卡。在第 i 个关卡中，小 Hi 他们需要获得 Ai 点分数才能够进入下一关。每一关的分数都是独立计算的，即使在一关当中获得超过需要的分数，也不会对后面的关卡产生影响。

小 Hi 他们可以通过答题获得分数。答对一道题获得 S 点分数，答错一道题获得 T 点分数。在所有的 N 个关卡中，小 Hi 他们一共有 M 次答题机会。在每个关卡中，都可以在累计答题次数不超过 M 的情况下使用任意次的答题机会。

那么现在问题来了，对于给定的 N、M、S、T 和 A，小 Hi 他们至少需要答对多少道题目才能够完成所有的关卡呢？

1.1.2. 输入

每个输入文件包含多组测试数据，在每个输入文件的第一行为一个整数 Q，表示测试数据的组数。

每组测试数据的第一行为四个正整数 N、M、S 和 T，意义如前文所述。

第二行为 N 个正整数，分别表示 A1~AN。

对于 40%的数据，满足 1<=N,M<=100

对于 100%的数据，满足 1<=N,M<=1000,1<=T<S<=10,1<=Ai<=50

对于 100%的数据，满足 1<=Q<=100

1.1.3. 输出

对于每组测试数据，如果小 Hi 他们能够顺利完成关卡，则输出一个整数 Ans，表示小 Hi 他们至少需要答对的题目数量，否则输出 No

1.2. 解析

首先我们可以发现，由于每一关需要的分数 Ai 是固定的，所以如果确定第 i 关

答错了 x 题，那么最少需要答对的题目数就是 max{0, ⌈(Ai-xT)/S⌉}。

为了描述方便我们把这个值记为 c[i][x], 即第 i 关如果答错的题目数是 x，那么这一关最少答对的题目数是 c[i][x]。

于是这道题实际是让我们决策每一关答错多少题，才能使得在总答题次数不超过 M 的情况下，总答对的题目数最少。

这是一个比较典型的动态规划题目，比较容易想到可以令 f[i][j]表示完成前 i 关如果总答错题目数是 j 次，最少需要的总答对题目数。

按关卡划分阶段，每次转移就是枚举第 i 关答错的题目数 x，即

$$f[i][j] = \min\{f[i-1][j-x] + c[i][x], \mid x = 0 .. \lceil Ai/T \rceil\}$$

最后我们要在所有 f[n][j]，j=0..M 中找到最小的 j 满足 f[n][j] + j <= M。

这个算法总状态数是 O(NM)的，转移复杂度是 O(max{Ai})的。对于极限数据还是可能超时。

另一种动态规划算法需要我们换一种状态表示。我们可以用 g[i][j]表示答错 i 题，答对 j 题时，能达到的最好记录是什么。

这里记录用一个二元组(x, y)表示，其中 x 是关卡，y 是得分。也就是说 g[i][j]=(x, y)表示答错 i 题，答对 j 题时，最高能进行到第 x 关，并且得分是 y。

显然任何两个记录(x1, y1)和(x2, y2)都是可比较优劣的。同时为了描述方便，我们定义一个记录"加"得分的算子+，(x1, y1) + s = (x2, y2)表示：如果当前在第 x1 关 y1 分，那么再加 s 分之后，到达的是第 x2 关 y2 分。

我们可以按答题总数划分阶段，每次转移就是枚举最后一题是答对还是答错了：

$$g[i][j] = \max\{g[i-1][j] + T, g[i][j-1] + S\}$$

最后我们在所有 g[i][j]里找到通过第 n 关，并且 j 最小的。

这个算法总复杂度是 O(M^2)，转移是 O(1)的。