**题目1：竞价**

时间限制: 1000ms 内存限制: 256MB

**描述**

Alice和Bob都要向同一个商人购买钻石。商人手中有 N 颗钻石，他会将它们一颗颗地卖给他们，Alice和Bob通过竞价的方式来决定钻石的归属。具体的过程如下：商人首先指定其中一个人开始报价，之后两人轮流报价，要求是一定要比对方报的价格更高。任何时候，如果一个人不愿出价或者出不起价钱时，可以宣布弃权，则对手以最后一次报的价格将钻石买下。当然，如果两人都没钱，商人是不会卖钻石的。首次报价至少为 1，并且只能报整数的价钱。

Alice和Bob特别爱攀比，因此他们都希望能比对方买到更多的钻石。Alice和Bob各自带了 CA 和 CB 的钱用于竞拍钻石。此外，Alice和商人有很不错的私人关系，因此商人总是会让Alice先报价。现在请问，在Alice和Bob都用最优策略的情况下，谁能买到更多钻石？假设双方都知道对方手中的现金数量，以及商人将要拍卖的钻石数量 N。

**输入**

输入文件包含多组测试数据。

第一行，给出一个整数T，为数据组数。接下来依次给出每组测试数据。

每组数据为三个用空格隔开的整数 N，CA，CB，表示钻石的数量，以及双方带的现金数量。

**输出**

对于每组测试数据，输出一行"Case #X: Y"，其中X表示测试数据编号，Y的取值为{-1, 0, 1}，-1表示Alice买到的钻石会比Bob少，0表示两人能买到一样多，1表示Alice能买到更多钻石。所有数据按读入顺序从1开始编号。

**数据范围**

1 ≤ T ≤ 1000

小数据：0 ≤ N ≤ 10; 0 < CA, CB ≤ 10

大数据：0 ≤ N ≤ 105; 0 < CA, CB ≤ 106

**样例输入**

2

4 3 5

7 4 7

**样例输出**

Case #1: 0

Case #2: 1

**解题思路**

题目描述中讲了大段竞价的规则，其实不必考虑得很复杂，双方的最优策略最终可以总结成一条：假设A要买下N枚钻石中的至少K枚，A的钱数为Ca，B的钱数为Cb，则A首先出一个自己能承受的最高价格P（P\*K<=Ca），然后检查B在P+1这个价格上能否买下超过N-K+1枚（(P+1)\*(N-k+1)>=Cb），如果可以，则A无法买下K枚。

于是，我们只要讨论一下N的奇偶性，然后判断A能否买下超过一半或恰好一半的钻石，最后输出结果。

**题目2：相似字符串**

时间限制: 4000ms 内存限制: 256MB

**描述**

对于两个长度相等的字符串，我们定义其距离为对应位置不同的字符数量，同时我们认为距离越近的字符串越相似。例如，“0123”和“0000”的距离为 3，“0123”和“0213”的距离则为 2，所以与“0000”相比，“0213”和“0123”最相似。

现在给定两个字符串 S1 和 S2，其中 S2 的长度不大于 S1。请在 S1 中寻找一个与 S2 长度相同的子串，使得距离最小。

**输入**

输入包括多组数据。第一行是整数 T，表示有多少组测试数据。每组测试数据恰好占两行，第一行为字符串 S1，第二行为 S2。所有字符串都只包括“0”到“9”的字符。

**输出**

对于每组测试数据，单独输出一行“Case #c: d”。其中，c 表示测试数据的编号（从 1 开始），d 表示找到的子串的最小距离。

**数据范围**

1 ≤ T ≤ 100

小数据：字符串长度不超过 1000

大数据：字符串长度不超过 50000

**样例输入**

3

0123456789

321

010203040506070809

404

20121221

211

**样例输出**

Case #1: 2

Case #2: 1

Case #3: 1

**解题思路**

这道题要求在S1中寻找与S2对应位置相同字符最多的子串。

对于小数据，通过枚举子串起点，再按照定义计算距离就可以通过。对于大数据，我们需要更高效的算法。

如果可以“批量”地计算出S1中所有长度为|S2|的子串与S2的相同字符数量，那么剩下的问题就仅仅是在这些数量值中寻找最大值了。对于S2和S1中的任意一个长度为|S2|的子串，显然这两个字符串相同字符的数量是'0', '1', ..., '9'各自相同字符数的总和，所以我们首先只考虑关于单个字符c的相同字符数量。于是可以把S1和S2看作成两个01序列：字符c对应于1，其他字符对应于0。具体地说，记S1和S2的长度分别为N1和N2，构造两个序列：

x[n] = x[n mod N1] = (S1[n mod N1] == c) ? 1 : 0

y[n] = (S2[N2 - n - 1] == c) ? 1 : 0

其中x是循环序列，y对S2头尾调转。聪明的你一定发现了，这两个序列卷积的第n项(x \* y)[n]就是S1[n + 1 ... n + N2]与S2相同的c字符的数量，而卷积可以利用FFT在O(NlogN)时间内计算(https://en.wikipedia.org/wiki/Fast\_Fourier\_transform)。

于是我们可以得到这样一个O(NlogN)的算法：对于每个字符c，构造序列x和y，使用FFT计算卷积，最后把结果相加，找最大值即可。需要注意的是，FFT中涉及的大量sin和cos的计算结果需要提前预处理，否则很容易超时。

因为计算一次卷积需要执行3次FFT，所以上面的算法总共需要执行30次FFT，常数很大。别忘了FFT可以计算复数域的卷积，利用这个特点可以一次处理两种字符c1和c2，只需稍稍修改下x和y的定义。在x中，c1对应于1，c2对应于i，其他字符对应于0；在y中恰好相反，c1对应于i，c2对应于1，其他字符对应于0。若记卷积第n项(x \* y)[n]为a + bi，那么b就是c1和c2的相同字符总数。经过这样的修改，计算量减少一半。

由于上述算法的常数较大，考虑到选手各种可能的实现，此题的时限设置比较宽松（改进版的三倍），但这也导致了少量比较紧凑的暴力算法通过了大数据。让人遗憾的是，在所有通过大数据的提交中，没有发现使用上述算法的代码。

**题目3：仙剑5前传之璇光殿**

时间限制: 4000ms 内存限制: 256MB

**描述**

仙剑是一款经典的RPG游戏，最近又推出了仙剑5前传。Alice身为忠实的仙剑粉丝，当然是在第一时间就开始玩了。迷宫以及各类小游戏是仙剑系列的一大传统，这次也不例外。而且还增加了称号系统，玩家如果在满足一定条件下通过迷宫或是完成小游戏，都可以获得相应的称号奖励。Alice虽然智商也不算太低，顺利的完成游戏还是没什么问题的，但是某些称号对于她来说好像就比较困难，所以她来找你帮忙。

在这个迷宫中，有很多魔法台需要关闭，也有很多宝箱可以捡。你最主要的目标就是关闭所有的魔法台，一旦所有魔法台都被关闭后，立刻通关。这个迷宫中有两个称号可以获得，一个要求拣到所有的宝箱，另一个要求在一定时间内完成。

为了让迷宫变得更加复杂，迷宫中还有两种特殊的法阵。第一种法阵是加速阵，可以瞬间使玩家的移动速度提高5m/s（初始速度为30m/s）。第二种是五灵阵，玩家必须按照五灵相克相生的关系来选择开启适当的阵法才可顺利通过。若玩家顺利通过五灵阵，则也可获得5m/s的速度提升，反之，用户则会被困在原地5秒钟。幸运的是，Alice在网上找到了攻略，所以在通过五灵阵时，她只需要查看一下即可保证顺利通过。不过介于她的思考速度，她需要3秒钟来查看攻略。

Alice想知道如果她想要拣到所有的宝箱，那么最快多长时间能够通过这个迷宫呢？

**输入**

输入数据的第一行包含一个整数 T，表示数据组数。

接下来有 T 组数据，每组数据中：

第一行包含两个整数 N, M，表示迷宫中节点个数以及边数。节点由 1 到 N 标号。

接下来的 M 行每行包含三个整数 a, b, len，表示节点 a 和节点 b 之间有一条长度为 len 米的路。

接下来的一行包含一个整数 NM，表示魔法台的个数。下一行 NM 个整数，表示魔法台所在的节点编号。

接下来的一行包含一个整数 NT，表示宝箱的个数。下一行 NT 个整数，表示宝箱所在的节点编号。

接下来的一行包含一个整数 NS，表示加速阵的个数。下一行 NS 个整数，表示加速阵所在的节点编号。

接下来的一行包含一个整数 NR，表示五灵阵的个数。下一行 NR 个整数，表示五灵阵所在的节点编号。

最后一行包含一个整数 S，表示玩家的出发节点编号。

**输出**

对于每组数据，输出一行"Case #X: Y",其中 X 为数据组数编号，从 1 开始，Y 为通过该迷宫的最短时间，四舍五入保留5位小数。

**数据范围**

1 ≤ T ≤ 10

0 ≤ len ≤ 1000

小数据：

1 ≤ N ≤ 30

0 ≤ M ≤ 200

1 ≤ NM ≤ 3

0 ≤ NT, NS, NR ≤ 3

大数据：

1 ≤ N ≤ 100

0 ≤ M ≤ 2000

1 ≤ NM ≤ 4

0 ≤ NT, NS, NR ≤ 4

**提示**

1. 每个节点上至多有一个特殊的物品。

2. 当你经过魔法台，加速阵以及宝箱时，可以选择不触发它。

3. 每个法阵都是一次性的，不会被多次触发。

**样例输入**

2

2 1

1 2 30

1

2

0

0

0

1

3 2

1 2 100

2 3 100

1

3

0

0

1

2

1

**样例输出**

Case #1: 1.00000

Case #2: 9.19048

**解题思路**

这道题其实就是一道最短路问题，只是增加了一些例如宝箱、魔法台、五灵阵和加速阵的特殊属性。由于每个特殊地点都只能被触发一次，所以在状态中必须记录当前有哪些地点是触发过的。所以最终的状态中应该包括：当前位置，特殊地点的触发状态以及当前的速度。其中，当前位置有用的其实也只有那些特殊地点，因为经过普通地点时其它状态都不会发生改变；速度之需要记录被提速了几次即可，因为每次提速都是固定的5m/s。所以总的状态大小应当为(Nm+Nt+Ns+Nr)\*2^(Nm+Nt+Ns+Nr)\*(Ns+Nr)。状态确定之后之需要用正常的最短路算法（spfa或dijkstra），状态转移时按照几个法阵不同的功能处理即可。需要注意的是，魔法阵是有可能经过不触发的，其它都是经过就直接触发。