声明：不是官方题解，有问题可以*CF*私信交流

A

做法很多，考虑离线。每个询问可以看作单点+1，然后对于每一个植物，区间查询它时间段内的和，如果不为0，则这个植物有1的贡献。离散化后可以用线段树或者树状数组等维护。

B

找出第一行的非'.'字符，如果是数字则直接返回，不是数字则递归左右两个区间。注意答案会爆int。

C

对数组a可以处理出将数字i映射到j带来的贡献。之后就变成一个二分图带权最大匹配问题，可以用KM解决。要求最小字典序的解，只需要对数组a从前到后的元素，枚举当前元素与1-m匹配时，剩下的元素能有的最大匹配。时间复杂度O($nm + m^5$)

D

本人的做法是离线，根据P{每个人出现的所有分数，每个人的编号}来离散化，用线段树维护每一个位置的人数。对于i获得分数p，i的硬币变化是i原始分数位置s到最新分数位置t的区间和。对于s到t这段的人，进行区间打标记。在某个人分数有变化时，进行单点修，把标记删除，记录标记对答案的贡献。

E

没看，不会，附上官方日文题解。

F

如果把'('看作1，')'看作-1，因此操作1就是区间*-1。考虑任意一个括号字符串，它是合法字符串，当且仅当：

1. 它的sum=0
2. 它的任意一个前缀和>=0

考虑一个不合法的字符串，要通过操作变成合法的，怎么操作能让操作次数最小。

1. 如果它的sum≠0，则需要根据sum的符号，在前添加')'或者在后添加'('；
2. 对于一个交换操作，其本质是让前缀和中的一个数字+2。

(	)	)	)	(	(
1	0	-1	-2	-1	0
(	)	)	(	)	(
1	0	-1	0	-1	0

所以，只考虑交换操作，答案是$(\sum_{i=1}^{n} pre[i] \times [pre[i] < 0] + \sum_{i=1}^{n} [pre[i]\%2 \neq 0]) \div 2$

对于这个问题，可以考虑分块。每个块大小是B，块内维护[-B,B]之间的信息，具体是：前缀和<=i的数字之和、前缀和<=i的奇数个数、前缀和<=i的偶数个数、块内sum、块内正负反转的标记。

取B=$\sqrt{n}$，这样，修改操作和询问操作都能在O($\sqrt{n}$)的时间复杂度内完成。注意最终答案会爆int。

G

考虑只有两个顾客1和2，先配送顾客1再配送顾客2的代价是：

$$X_{1->2} = a_1t_1 + a_2(2t_1 + t_2 + 1)$$

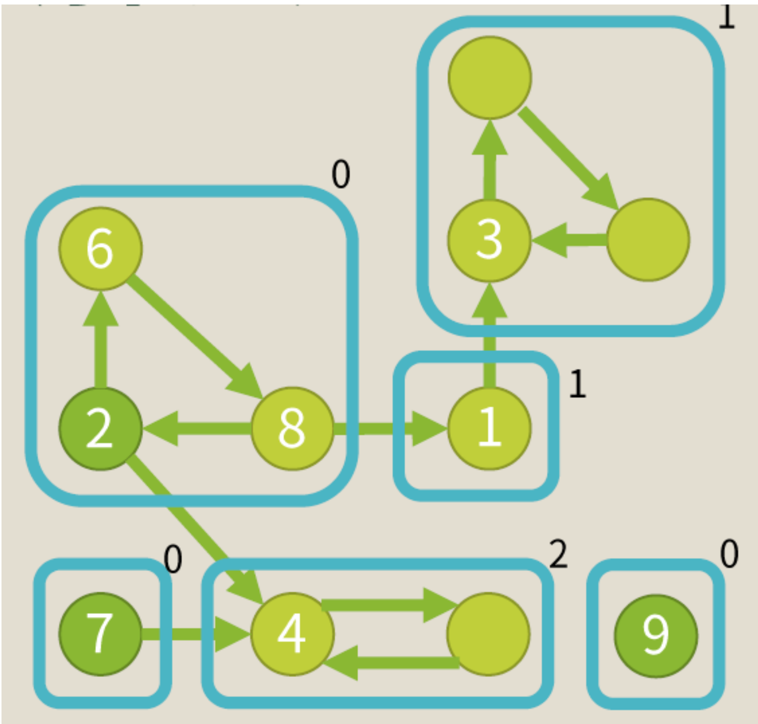
则先1后2和先2后1的代价差是：

$$\begin{aligned} X_{1->2} - X_{2->1} &= a_1t_1 + a_2(2t_1 + t_2 + 1) - a_2t_2 + a_1(2t_2 + t_1 + 1) \\ &= 2a_2t_1 + a_2 - 2a_1t_2 - a_1 \\ &= a_2(2t_1 + 1) - a_1(2t_2 + 1) \end{aligned}$$

所以，如果$\frac{a_1}{2t_1+1} > \frac{a_2}{2t_2+1}$，则先配送1代价更小。这个结论可以推广到顾客数为n的情况。按照$\frac{a_i}{2t_i+1}$从大到小排序，然后模拟做即可。

H

对于一个联通分量，对其缩点，计算出每一个联通分量的入度，则答案的个数一定是入度为0的连通分量个数。对于每一个联通分量，取最小编号的点即可。具体如下图所示：



一次tarjan即可完成，时间复杂度O(n+m)

I

既然是可以任意排列的，答案就是$\sum_{i=1}^{N} a_i - (max_{h_i} - min_{h_i})$

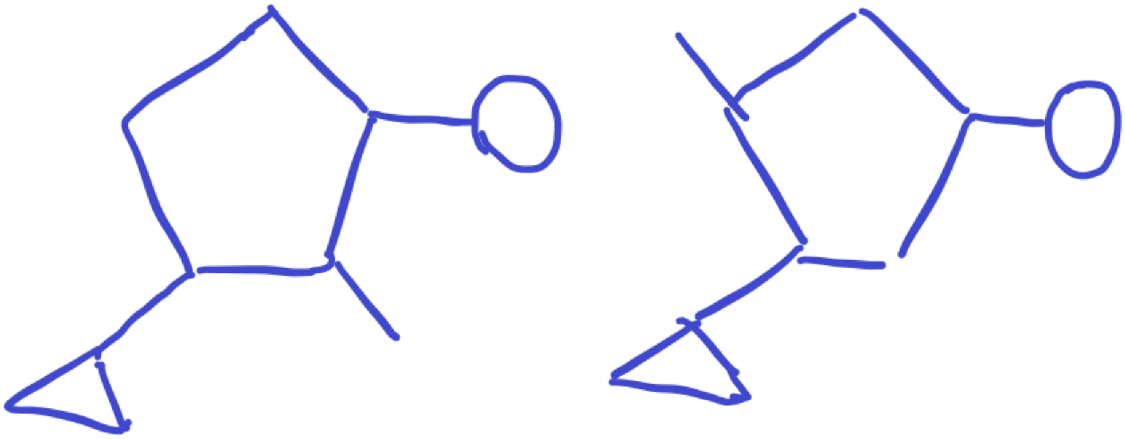
按照身高从小到大排序，考虑DP，记dp[i][j]表示前i个人，选了j个时，{a之和+最小身高}的最大值。转移方程为：

$$\begin{aligned} dp[i][1] &= max(dp[i-1][1], a[i] + h[i]) \\ dp[i][j] &= max(dp[i-1][j] + dp[i-1][j-1] + a[i]) \\ ans &= max(ans, dp[i-1][j-1] + a[i] - h[i]) \end{aligned}$$

注意dp初始化以及答案爆int。时间复杂度O(SN)

J

不能简单把环缩点变成树做hash。注意一下下图的情况。



因为只有n个点，n条边的连通图，所以一定是有一个环。以这个环里的每一个点为树根，对它的子树做hash，分别得到一个hash数组。然后对这两个图的hash数组按照各种姿势（移位、倒置）判断一下每一个元素是否相等即可。

K

用类似求最短路的方法，优先队列维护，每次选出一个最短时间t被感染的人，然后求出在t时间后，他感染其他人的时间。这里我的处理是使用一次三分，找出t时间后两人距离最近的时刻v，然后在t到v之间再用一次二分找出最早的距离<=D的时刻。不管是时间还是精度都比较卡，不断调整eps后最终还是卡过去了。

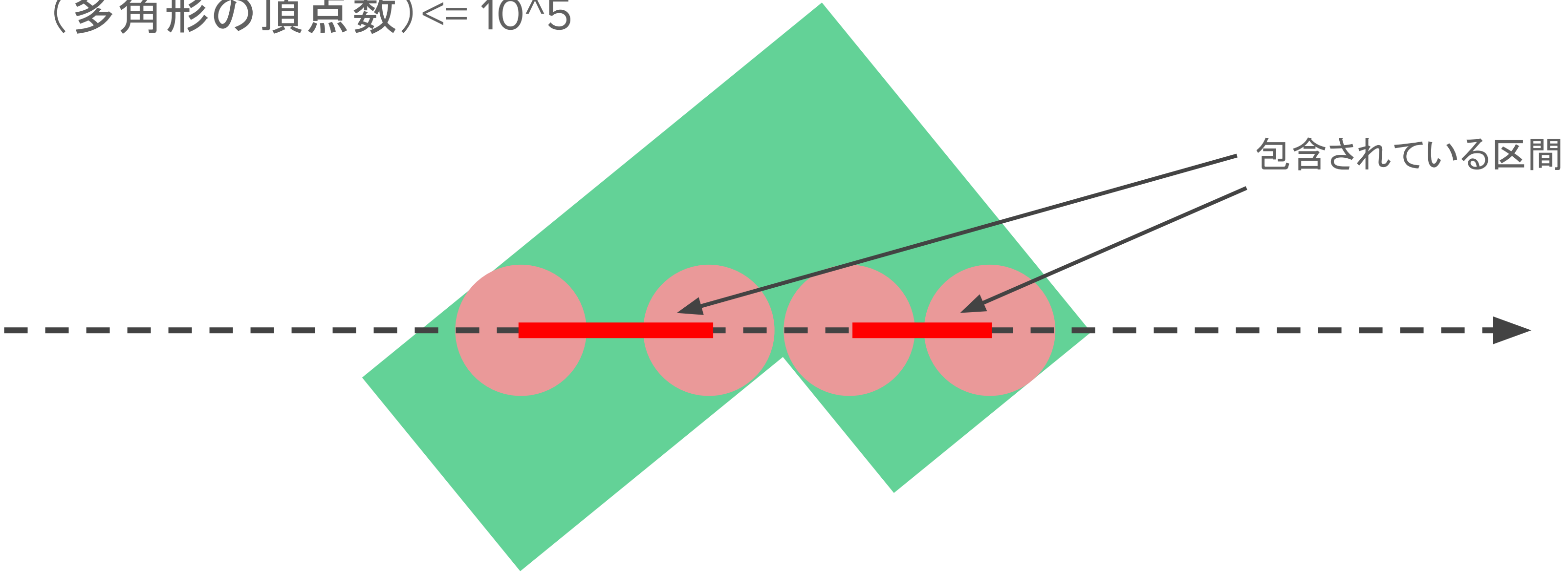
也有人用计几处理出两两之间距离在D内的时间段，然后建图用最短路的方法搞。

JAG ICPC模擬地区予選2021
E: Underground's SUNDAY

原案：shora_kujira16
問題文：riantkb
データセット：Darsein
解答：climpet, hos, riantkb
解説：riantkb

問題概要

- 二次元平面上に凸とは限らない多角形がある
    - サンプルには凸なものしか入ってなかったようです ...
- x 軸上を移動する半径 R の円が多角形に完全に包含される総時間を求めよ
- (多角形の頂点数)<= 10^5



考察

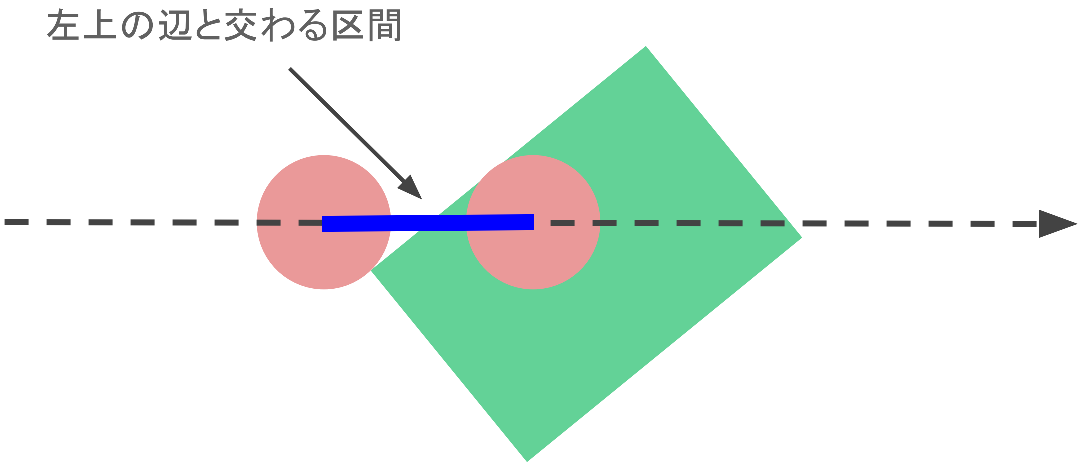
- 円が多角形に包含されているかどうかが入れ替わるタイミングでは、必ず多角形と円は（辺または頂点で）接している

- つまり、円と多角形が接するタイミング全てに対し以下の 2 つが判定できれば良い
  - 他の辺が円と交わっていない
  - 円の中心が多角形の内部にある

- しかし、どちらも愚直に判定すると多角形の頂点数を N として O(N) かかるため、全体で O(N^2) になってしまう

考察

- 多角形の全ての辺と円が交わっていないかを高速に判定したい
- これは、多角形の一辺(線分)と円の交差条件(いつからいつまで交差しているか)を求めておくことで、「いま何本の辺と交わっているか」を管理できるので判定できる



考察

- 円の中心が多角形の内部にあるかを高速に判定したい
- これは、x 軸と各辺がいつ交わるかを求めておくことで、多角形の辺と
  奇数回交わったならば多角形の内部であると言える
  - x 軸にピッタリ重なる辺や端点が x 軸上にある辺の処理が面倒になるため、実際には
    直線 $y = 0.5$ と各辺の交点を管理すると楽

- 別の方法として、各頂点を反時計回りにした上で、
  y 座標が小さい方へ進む辺を横切った場合内部に入った、
  y 座標が大きい方へ進む辺を横切った場合外部へ出た、とすることもできる

解法

- 各辺について、以下のタイミングを頑張って求めておく
  - 辺と円が交わり始めるタイミング
  - 辺と円が交わり終わるタイミング
  - 円の中心が辺上に乗るタイミング
- それらを早い順に見て、何本の辺と交わっているかと
  円の中心が多角形の内部にあるかどうかを管理する
- 1 本の辺とも交わってないかつ中心が多角形の内部にある場合に、
  次のタイミングまでの時間を答えに足す
- 計算量は O(N log N) となる

解法

- 実装によっては x 軸に平行な辺、y 軸に平行な辺、端点が x 軸上にある辺などで壊れるので注意が必要
    - 惜しい提出が WA になっているのはおそらく大体この辺のせい

ジャッジ解

- climpet (C++): 156 lines, 3.3 kB
- hos (C++): 156 lines, 5.0 kB
- riantkb (C++): 151 lines, 4.1 kB

統計情報

- Acceptances / Submissions
  - 4 / 30 (13.33 %)
- AC teams / Trying teams
  - 4 / 10 (40.00 %)
- First Acceptance
  - SPJ (220 min)