A. 最小公倍树

- 题目大意：对 $V = \{L, L+1, \cdots, R\}$ 的完全图建立最小生成树，其中 (u, v) 边的边权是 $\mathrm{lcm}(u, v)$
- $1 \leq L \leq R \leq 10^6$，$R - L \leq 10^5$

A. 最小公倍树

- 直接 Kruskal，边数是 $O\left((R-L)^2\right)$ 的，显然不能通过
- 由于边权是最小公倍数，考虑分因数优化建图
- 记 $m_d = d\left\lceil\frac{L}{d}\right\rceil$，假设有 $L \le m_d < u < v \le R$，其中 u, v 是 d 的倍数，$\gcd(u,v) = d$
  - $\gcd(u,v) > d$ 的情况会在别的因数时处理
- 若 (u,v) 在最小生成树中，将 (u,v) 断开后，m_d 一定恰与 u, v 中一个点连通
- $\operatorname{lcm}(m_d,u) \le \frac{m_d u}{d} < \frac{uv}{d} = \operatorname{lcm}(u,v)$，同理 $\operatorname{lcm}(m_d,v) < \operatorname{lcm}(u,v)$，故通过 m_d 将树重新连通，得到的生成树比原来的树权值和小
- 故对每个因数 d 而言，kd 只需要向 m_d 连边

A. 最小公倍树

- 由于边权是最小公倍数，考虑分因数优化建图
- 对每个因数 d 而言，kd 只需要向 m_d 连边
- 因数个数和是 $O\big((R-L)\ln(R-L)\big)$ 级别的，边数与因数个数和同级别，故 Kruskal 总时间复杂度是 $O\big((R-L)\log^2(R-L)\big)$
- 当运算复杂度看作 $O(1)$ 时复杂度和 L, R 大小无关，限制 $L \le R \le 10^6$ 只是保证不用写高精度
- 就算没发现这个性质也可以做，以下介绍使用 Prim 和 Borůvka 的做法

A. 最小公倍树

- 考虑 Prim，每次查询已经连通的集合与没加入集合的点之间连边的最小值
- 在 Prim 中，我们对每个点维护其到连通集合的距离；为了减小每次更新的复杂度，可以考虑分因数维护
  - 对每个因数 d 维护当前在集合中的点 u 和不在集合中的点 v 的 $\mathrm{lcm}(u,v)$ 的最小值，其中 u,v 都是 d 的倍数
- 由于因数个数是 $O\big((R-L)\ln(R-L)\big)$ 的，维护开销大大减小

A. 最小公倍树

- 更新一个因数 d 的边权时，需要知道已经在集合中的点的最小值 u_d 以及不在集合中的点的最小值 v_d（u_d, v_d 是 d 的倍数）
  - 可以（均摊）$O(1)$ 维护
- 更新完每个因数 d 对应的最小边权后，需要查询全局最小值
- 用单次修改 $O(\log(R-L))$，全局查询 $O(1)$ 的数据结构维护，总复杂度就是 $O\left((R-L)\log^2(R-L)\right)$

A. 最小公倍树

- 用 Borůvka 算法，也可以获得一个相同复杂度的做法
  - 一开始每个点属于一个集合
  - 每轮迭代，对每个集合 S 查询 S 与其它集合连边中最小的那条 e_S，并将所有的 e_S 加入最小生成树
  - 如果边权互不相同，保证加入的边不会连成环
- 查询时枚举每个点 x，枚举 x 的因数 d，此时 x 向其它集合连边最小值要么是全局 d 最小值 m_d，要么是与 m_d 属于不同集合的 d 的倍数中最小的 m'_d（此时 x 与 m_d 属于同一集合）
- 只需维护 m'_d，同样可以均摊 $O(1)$
- 做完这个题可以研究一下样例中的彩蛋

B. 骰子旅行

- 需要用到概率和期望的性质
- 推推式子
- 记 X_i 表示第 i 次说话记录价值（没有废话时 $X_i = 0$）

B. 骰子旅行

$$
\begin{aligned}
E\left(\sum_{i=1}^{T} X_i\right) &= \sum_{i=1}^{T} E\left(X_i\right) \\
&= \sum_{i=1}^{T}\sum_{x=1}^{N} xP\left(X_i = x\right) \\
&= \sum_{i=1}^{T}\sum_{x=1}^{N} x \sum_{j=0}^{i-1}\sum_{y=1}^{N} P\left(s_j = s_i = y, s_{j+1} = x, \forall j < k < i, s_k \neq y\right) \\
&= \sum_{i=1}^{T}\sum_{x=1}^{N} x \left(P\left(s_{i-1} = s_i = x\right) + \sum_{j=0}^{i-1} \sum_{\substack{(y,x)\in E \\ y\neq x}} P\left(s_j = s_i = y, s_{j+1} = x, \forall j < k < i, s_k \neq y\right) \right) \\
&= \sum_{i=1}^{T}\sum_{x=1}^{N} xP\left(s_{i-1} = s_i = x\right) \\
&\quad + \sum_{i=1}^{T}\sum_{x=1}^{N}\sum_{j=0}^{i-1} \sum_{\substack{(y,x)\in E \\ y\neq x}} xP\left(s_j = s_i = y, s_{j+1} = x, \forall j < k < i, s_k \neq y\right)
\end{aligned}
$$

↓第 i 次废话说的是 x 的充要条件为：存在某个 y，之前某次经过 (y,x) 这条边，且恰好在第 i 次返回 y

↑分是否为自环讨论

B. 骰子旅行

- $O(TM)$ 预处理 $P(s_i = x)$
- 对于自环的情况可以 $O(TN)$ 计算

$$\sum_{i=1}^{T}\sum_{x=1}^{N} xP\left(s_{i-1}=s_i=x\right)$$

$$=\sum_{x=1}^{N} x\sum_{i=1}^{T} P\left(s_{i-1}=x\right)P\left(s_i=x\,\middle|\,s_{i-1}=x\right)$$

$$=\sum_{(x,x)\in E} x\sum_{i=1}^{T} P\left(s_{i-1}=x\right)\frac{1}{m_x}$$

$$=\sum_{(x,x)\in E} \frac{x}{m_x}\sum_{i=0}^{T-1} P\left(s_i=x\right)$$

$$\sum_{i=1}^{T}\sum_{x=1}^{N}\sum_{j=0}^{i-1}\sum_{\substack{(y,x)\in E\\ y\neq x}} xP\left(s_j=s_i=y,s_{j+1}=x,\forall j<k<i,s_k\neq y\right)$$

↓用条件概率拆分

$$=\sum_{i=1}^{T}\sum_{x=1}^{N}\sum_{j=0}^{i-1}\sum_{\substack{(y,x)\in E\\ y\neq x}} xP\left(s_j=y\right)P\left(s_{j+1}=x\middle|\, s_j=y\right)P\left(s_i=y,\forall j<k<i,s_k\neq y\middle|\, s_j=y,s_{j+1}=x\right)$$

↖枚举点 y

$$=\sum_{i=1}^{T}\sum_{j=0}^{i-1}\sum_{\substack{(y,x)\in E\\ y\neq x}} xP\left(s_j=y\right)P\left(s_{j+1}=x\middle|\, s_j=y\right)P\left(s_i=y,\forall j<k<i,s_k\neq y\middle|\, s_j=y,s_{j+1}=x\right)$$

↖枚举边

$$=\sum_{i=1}^{T}\sum_{j=0}^{i-1}\sum_{\substack{(y,x)\in E\\ y\neq x}} xP\left(s_j=y\right)\frac{1}{m_y}P\left(s_i=y,\forall j<k<i,s_k\neq y\middle|\, s_j=y,s_{j+1}=x\right)$$

$$=\sum_{\substack{(y,x)\in E\\ y\neq x}}\frac{x}{m_y}\sum_{j=0}^{T-1}P\left(s_j=y\right)\sum_{d=2}^{T-j}P\left(s_d=y,\forall 0<k<d,s_k\neq y\middle|\, s_0=y,s_1=x\right)$$

$$=\sum_{\substack{(y,x)\in E\\ y\neq x}}\frac{x}{m_y}\sum_{j=0}^{T-1}P\left(s_j=y\right)\sum_{d=2}^{T-j}P\left(s_d=y,\forall 0<k<d,s_k\neq y\middle|\, s_1=x\right)$$

↑预处理　↑这部分概率只和经过的时间有关，与起始时刻无关

B. 骰子旅行

$$\sum_{\substack{(y,x)\in E\\ y\neq x}} \frac{x}{m_y} \sum_{j=0}^{T-1} P(s_j = y) \sum_{d=2}^{T-j} P(s_d = y, \forall 0 < k < d, s_k \neq y | s_1 = x)$$

- 框出来这部分概率倒着算的比较好算
- 考虑全概率公式，用 $P(s_d = y, \forall 0 < k < d, s_k \neq y | s_1 = x)$ 转移给 $P(s_{d+1} = y, \forall 0 < k < d+1, s_k \neq y | s_1 = z)$：

$$\begin{aligned}
&P(s_{d+1} = y, \forall 0 < k < d+1, s_k \neq y | s_1 = z) \\
&= \sum_{(z,x)\in E} P(s_{d+1} = y, \forall 1 < k < d+1, s_k \neq y | s_2 = x) P(s_2 = x | s_1 = z) \\
&= \sum_{(z,x)\in E} P(s_d = y, \forall 0 < k < d, s_k \neq y | s_1 = x) \frac{1}{m_z}
\end{aligned}$$

- 这样计算概率是 $O(TNM)$ 的，总复杂度 $O(T^2M + TNM)$
  - 另外也有 $O(TNM + T^2N^2)$ 之类的做法，常数正常就能过

THUPC 2022 初赛
C 赛程制定 solution

出题：交叉信息研究院 程泽瑞
验题：计算机科学与技术系 许哲楠

2022年3月12日

General info

- 中期题
- 数据结构模板题
- 套路题
- 推式子
- 细节题

关于题面

- 如果有F1车迷，可以发现人名来自梅奔和红牛车队
- 希望梅奔车迷对此不要有太大意见（doge）
- （出题人其实也是梅奔车迷，所以黑了一手自己的主队233）
- 插句题外话，F1 2022下周战火重燃，欢迎入坑~

Brief Statement

- 有1，2，3，4四个不同的人，1,2是A类，3,4是B类
- 共有n天，每天需要选出2个人
- 对于第i天，A+A收益为ai，A+B收益为bi，B+B收益为ci，满足ai<bi<ci
- 对于A+B的情况，A类中出场的一定是2
- 对于3，最后一次和第一次出场间隔至多为p
- 对于4，最后一次和第一次出场间隔至多为q

- 对于所有任意修改一天中一个选手都无法获得更好收益的方案，求收益中位数
- 数据范围：2*10^5

什么是“lewis的最优方案”？

首先，将t_m, t_c与n取min后，可以发现，所谓“Lewis的最优方案”中，Max一定出场连续的t_m天，Checo一定出场连续的t_c天。

可以反证法证明：不失一般性，假设Max的出场场次不连续，可以把任意一场中间未出场比赛中的Lewis或Valtteri换成Max，从而在不违反规则的情况下增加收益，因此一定连续；假设Max的出场场次不足t_m场，直接向前或者向后延长一场比赛即可在不违反规则的情况下增加收益，对于Checo同理——因此，确定了p_m和p_c，即可唯一确定一种可能的属于“Lewis的最优方案”的分配方式。

另外，我们可以证明，如果 $1 \le p_m \le n - t_m + 1$ 且 $1 \le p_c \le n - t_c + 1$，那么此方案一定属于“Lewis的最优方案”。

对于任意符合以上条件的p_m, p_c组合，注意到仅仅通过改变序列的一个位置增加总收入，等价于将一场比赛中的Lewis或Valtteri换成Max或Checo，但由因为Max和Checo的出场数已经都达到了最大可能值，因此无法实现，所以我们得到了“Lewis的最优方案”对应充要条件。

因此，我们知道，总方案数量为$(n - t_m + 1)(n - t_c + 1)$，而找中位数，等价于找到收入第$\lfloor \frac{(n-t_m+1)(n-t_c+1)+1}{2} \rfloor$和$\lceil \frac{(n-t_m+1)(n-t_c+1)+1}{2} \rceil$的方案收入的平均值（无论对总方案数是奇数还是偶数的情况均成立），考虑化为求收入第k多的方案收入，然后分别令$k = \lfloor \frac{(n-t_m+1)(n-t_c+1)+1}{2} \rfloor$以及 $k = \lceil \frac{(n-t_m+1)(n-t_c+1)+1}{2} \rceil$即可求出最终答案。

怎么求第k大的方案？

考虑二分答案，转化为对于给定的val, 判断有多少种方案的收入小于等于val。

首先，将a, b, c分别作前缀和（此后用a, b, c指代前缀和数组），对于给定的p_m和p_c，不妨设$t_m \le t_c$，最终的收入为

1. 若 $1 \le p_m \le p_m + t_m - 1 < p_c \le p_c + t_c - 1 \le n$，则 （不相交，短的在前）
   $sum_{p_m,p_c} = a_{p_m-1} + (b_{p_m+t_m-1} - b_{p_m-1}) + (a_{p_c-1} - a_{p_m+t_m-1}) + (b_{p_c+t_c-1} - b_{p_c-1}) + (a_n - a_{p_c+t_c-1})$
2. 若 $1 \le p_m < p_c \le p_m + t_m - 1 < p_c + t_c - 1 \le n$，则 （相交，短的在前）
   $sum_{p_m,p_c} = a_{p_m-1} + (b_{p_c-1} - b_{p_m-1}) + (c_{p_m+t_m-1} - c_{p_c-1}) + (b_{p_c+t_c-1} - b_{p_m+t_m-1}) + (a_n - a_{p_c+t_c-1})$
3. 若 $1 \le p_c \le p_m \le p_m + t_m - 1 \le p_c + t_c - 1 \le n$，则（长的覆盖短的）
   $sum_{p_m,p_c} = a_{p_c-1} + (b_{p_m-1} - b_{p_c-1}) + (c_{p_m+t_m-1} - c_{p_m-1}) + (b_{p_c+t_c-1} - b_{p_m+t_m-1}) + (a_n - a_{p_c+t_c-1})$
4. 若$1 \le p_c < p_m \le p_c + t_c - 1 < p_m + t_m - 1 \le n$，则（相交，长的在前）
   $sum_{p_m,p_c} = a_{p_c-1} + (b_{p_m-1} - b_{p_c-1}) + (c_{p_c+t_c-1} - c_{p_m-1}) + (b_{p_m+t_m-1} - b_{p_c+t_c-1}) + (a_n - a_{p_m+t_m-1})$
5. 若$1 \le p_c \le p_c + t_c - 1 < p_m \le p_m + t_m - 1 \le n$，则（不相交，长的在前）
   $sum_{p_m,p_c} = a_{p_c-1} + (b_{p_c+t_c-1} - b_{p_c-1}) + (a_{p_m-1} - a_{p_c+t_c-1}) + (b_{p_m+t_m-1} - b_{p_m-1}) + (a_n - a_{p_m+t_m-1})$

怎么求第k大的方案？

接下来，对于这些情况，可以将p_m, p_c分离变量如下（以第一种情况为例）：

- $sum_{p_m,p_c} = (b_{p_m+t_m-1} - a_{p_m+t_m-1} + a_{p_m-1} - b_{p_m-1}) + (a_{p_c-1} - b_{p_c-1} + b_{p_c+t_c-1} - a_{p_c+t_c-1}) + a_n \sim x_{p_m} + y_{p_c} + a_n$

  枚举p_m，化为：求满足 y_{p_c} 不超过给定阈值$val - a_n - x_{p_m}$ 且 $p_c \in [p_m + t_m, n - t_c + 1]$ 的 p_c数量

- 相似的，可以得到——

  对于第2种情况，

- $sum_{p_m,p_c} = (c_{p_m+t_m-1} - b_{p_m+t_m-1} + a_{p_m-1} - b_{p_m-1}) + (b_{p_c-1} - c_{p_c-1} + b_{p_c+t_c-1} - a_{p_c+t_c-1}) + a_n \sim x_{p_m} + y_{p_c} + a_n$

  枚举p_m，化为：求满足 y_{p_c} 不超过给定阈值$val - a_n - x_{p_m}$ 且 $p_c \in [p_m + 1, min(p_m + t_m - 1, n - t_c + 1)]$ 的 p_c数量

  对于第3种情况，

- $sum_{p_m,p_c} = (c_{p_m+t_m-1} - b_{p_m+t_m-1} + b_{p_m-1} - c_{p_m-1}) + (a_{p_c-1} - b_{p_c-1} + b_{p_c+t_c-1} - a_{p_c+t_c-1}) + a_n \sim x_{p_m} + y_{p_c} + a_n$

  枚举p_m，化为：求满足 y_{p_c} 不超过给定阈值$val - a_n - x_{p_m}$ 且 $p_c \in [max(1, p_m + t_m - t_c), min(p_m, n - t_c + 1)]$ 的 p_c数量

- 对于第4种情况，

  $sum_{p_m,p_c} = (b_{p_m+t_m-1} - a_{p_m+t_m-1} + b_{p_m-1} - c_{p_m-1}) + (a_{p_c-1} - b_{p_c-1} + c_{p_c+t_c-1} - b_{p_c+t_c-1}) + a_n \sim x_{p_m} + y_{p_c} + a_n$

- 枚举p_m，化为：求满足 y_{p_c} 不超过给定阈值$val - a_n - x_{p_m}$ 且 $p_c \in [max(0, p_m - t_c) + 1, p_m + t_m - t_c - 1]$ 的 p_c数量

  对于第5种情况，

- $sum_{p_m,p_c} = (b_{p_m+t_m-1} - a_{p_m+t_m-1} + a_{p_m-1} - b_{p_m-1}) + (a_{p_c-1} - b_{p_c-1} + b_{p_c+t_c-1} - a_{p_c+t_c-1}) + a_n \sim x_{p_m} + y_{p_c} + a_n$

  枚举p_m，化为：求满足 y_{p_c} 不超过给定阈值$val - a_n - x_{p_m}$ 且 $p_c \in [1, p_m - t_c]$ 的 p_c数量

Ending

- 发现5种情况对应的都是二维数点问题
- 离散化+树状数组即可解决
- 加上二分的复杂度，总体时间复杂度为O(nlog^2n)

- （推式子+数据结构模板题）

感谢各位对THUPC的支持！

D

liuzhangfeiabc

题目大意

- 给定排列p[1…n]
- 你要在后面添加m个数p[i]=i（i=n+1~n+m），构成一个n+m的排列
- 然后每次可以交换两个数，要求每一对位置最多只能交换一次，并且每次交换至少有一个位置需要是后m个位置
- 最后要让排列变得有序
- 你要最小化m

- 简单题，不解释

sol

- 除非排列本来就有序，否则m=2
- 构造：对于每个循环p[i1]=i2，p[i2]=i3，…，p[ik]=i1，进行操作：
- swap(n+1,i1)
- swap(n+1,i2)
- …
- swap(n+1,i_{k-1})
- swap(n+2,ik)
- swap(n+1,ik)
- swap(n+2,i1)
- 现在整个循环归位了，n+1和n+2是反的
- 依次做完所有循环之后，如果需要就交换一下n+1和n+2就行

E. 搬砖

- 题目大意：请问蓝龙亵渎打几？

- 每一摞砖有两个属性：数量和使小E下降的精力
- 每个小时小E会在每摞砖搬走等于自己的精力的块数
- 如果有砖摞搬完，那么小E的精力会下降与这摞砖属性相同的数值，并且继续搬运
- 直到某个小时之后没有新的砖摞被搬完或者精力下降到0或以下
- 询问之间互相独立，有添加操作，问小E可以搬几个小时

E. 搬砖

- 考虑维护一个变量now表示每摞砖被移走了几个
- 对于精力比较高的情况，可以暴力查找区间内有没有新的砖摞被移除，以及这些砖摞的属性之和，这是一个区间求和
- 否则可以维护一下信息
- 考虑维护砖摞数量上的gap
- 对于一个gap来说，枚举精力值，对于now模当前精力值的余数分类，会使一部分的询问停下来
- 这里是一个三维偏序（时间，余数区间，now的值）
- 用一个两个log的树套树维护

E. 搬砖

- 然后就是看gap先出现还是下一摞属性不等于0的砖先出现
- 这样只需要进行O(精力)次的树套树查询
- 区间求和可以用O(根号)修改O(1)查询的数据结构
- 这样两边均摊一下一边一个log
- 时间复杂度为$O(T\sqrt{n}\log n)$
- 验题人写了一个$\tilde{O}\left(Tn^{2/3}\right)$的，跑得比我快，大概是把余数区间拆开了

pF. 喵喵花园

出题人：Yen-Jen Wang

题目叙述

喵喵是一只非常富有的猫咪，他在海淀区拥有一个大花园。

这个大花园是由一些旧栅栏为边界所形成的凸N-gon（即具有 N边的凸多边形）。

由于圣诞节快到了，喵喵想用K棵圣诞树来装饰一下花园。 同时，喵喵坚信找到一些好的位置来种树会给他带来好运。

作为一只好猫咪，他决定寻找最佳位置如下：

- 所有的树都应该在花园的边界上。
- 这K棵圣诞树应该平均划分花园的周长。
- 由树木形成的新凸面K-gon 的面积应尽可能小。

虽然喵喵比你有钱，但他没有你那么聪明。 因此，他给了你一些钱，让你帮他找出凸K-gon 的最小面积。

一句话题意

给你一个N的点的凸多边形，从该多边形的边上选出N个周长等分点，使得这K个点形成的凸多边形面积最小。

- $3 \leq N, K \leq 1000$
- $-10^5 \leq x_i, y_i \leq 10^5$

一些小观察

一些小观察

- 选出第一个点以后，整个K-gon就确定了。

一些小观察

- 选出第一个点以后，整个K-gon就确定了。
- 假设p为最优解中K-gon上的其中一个点。

一些小观察

- 选出第一个点以后，整个K-gon就确定了。
- 假设p为最优解中K-gon上的其中一个点。
- 考慮某一條邊，点p點在上面，选出的第一个点离p点越接近，形成的K-gon越小，反之则越大。（详细证明略）

一些小观察

- 选出第一个点以后，整个K-gon就确定了。
- 假设p为最优解中K-gon上的其中一个点。
- 考慮某一條邊，点p點在上面，选出的第一个点离p点越接近，形成的K-gon越小，反之则越大。（详细证明略）

三分搜！

一些小观察

- 选出第一个点以后，整个K-gon就确定了。
- 假设p为最优解中K-gon上的其中一个点。
- 考慮某一條邊，点p點在上面，选出的第一个点离p点越接近，形成的K-gon越小，反之则越大。（详细证明略）

三分搜！ $O(NK\log(C))$

一些小细节

- 浮点精度问题。
- N-gon的同一条边上可能会选出多的点。
- …

Thank you!

G. 挑战

- 记 $f(i, 0/1, S)$ 表示当前在第 i 个格子，由 Alice/Bob 操作，当前操作状态对应的最优概率（Alice 为最大化，Bob 为最小化）
- 可能的状态有三种：一种是正常操作；一种是挑战成功的额外操作；一种是对手赠送的额外操作
- 分情况讨论一下，注意预处理每种 $(p_k, q_k + q_{k+x})$ 情况对应的挑战成功、打平、失败的概率
- 总复杂度 $O(Np + pq)$

TEST_96 题解

——nzhtl1477

有向邻域的不删除莫队。

使用树分块，将树分为若干个簇，其中有二度簇（存在两个端点，且为祖先关系）和一度簇（只有一个端点），端点是多个簇共用的点或原树的根。这个树分块需要保证每个簇不超过 B 个点，簇的个数为 $O(\frac{n}{B})$。

枚举每个二度簇，处理 x 在两个端点之间路径上的询问，设有 m_1 个。

询问可以表示为满足深度 $<((x\text{的深度})+y)$，且在 x 子树内的点构成的集合。按深度升序插入当前簇的下方端点的子树内的点，遇到询问时插入询问集合与块的交然后撤销，一个块处理完后撤销回空集。这部分代价为 $O(\frac{n^2}{B}+m_1B)$，操作数为 $5m_1$。

对于余下的询问（设有 $m_2=m-m_1$ 个），可能在一度簇中，或者在二度簇中但不在端点间路径上，此时询问邻域的点数不超过 B，可以直接从空集出发一步到达后撤销，这部分代价为 $O(m_2B)$，操作数为 $3m_2$。

总代价 $O(\frac{n^2}{B}+mB)=O(n\sqrt{m}+m)$，总操作数不超过 $5m$。

给出莫队方案的时间复杂度可以做到 $O(n+m)$。

可以用更精细的分析得到较好的常数上界。

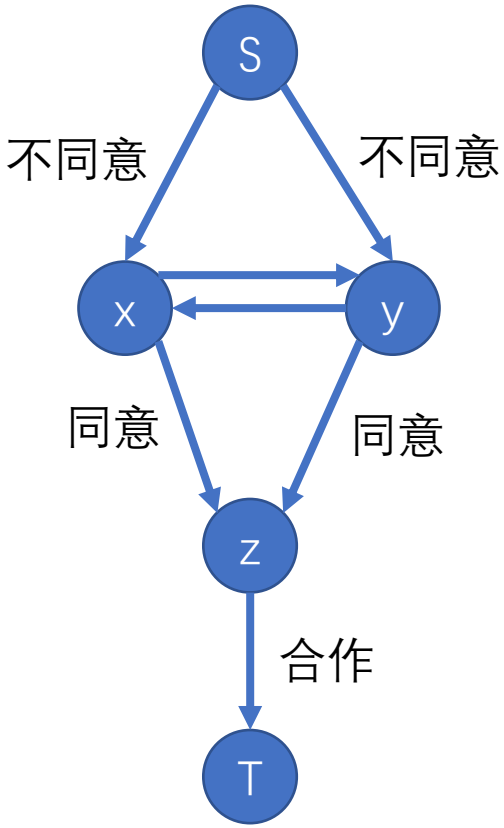
I. 分组作业

- 题目大意：有n个组，每组2人
- 每个人先进行一次表决，同意或不同意
- 若组里两个人都同意，那么可以选择合作或不合作

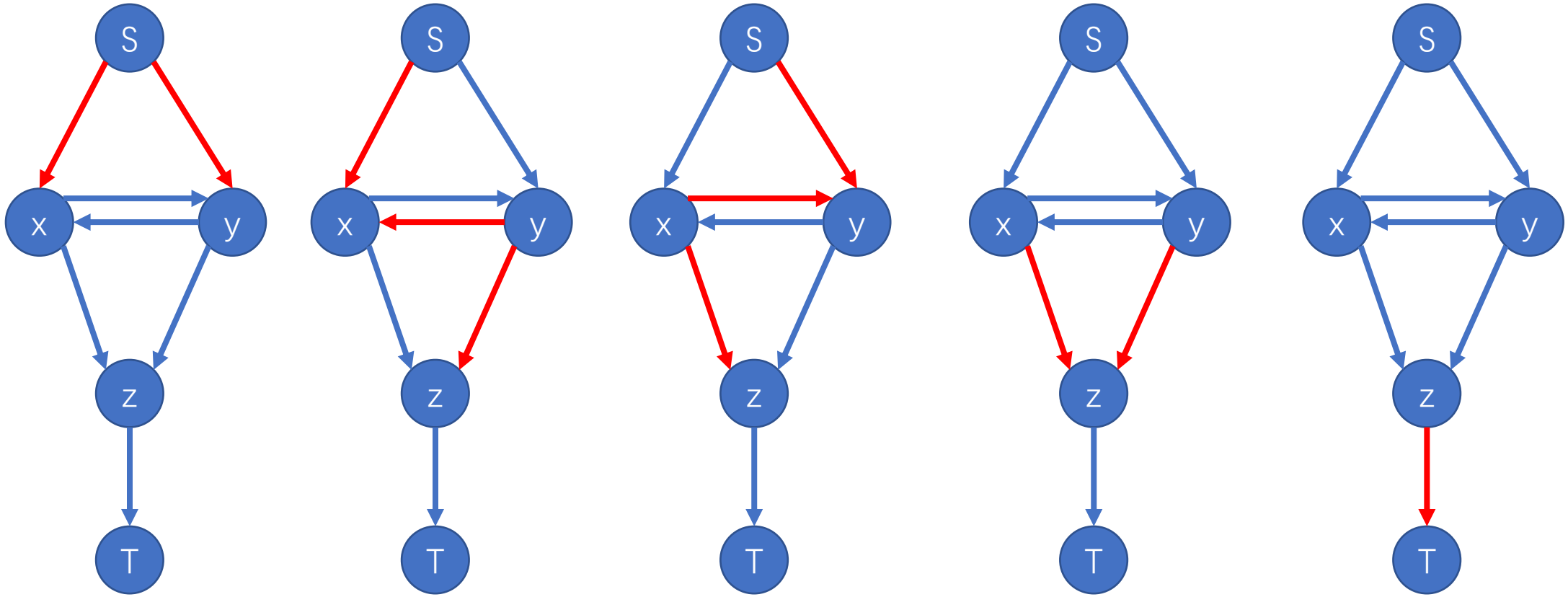
- 每个人选择同意有一个代价，选择不同意有一个代价
- 组里两个人选择不相同有一个代价
- 还有一些在不同组之间的人的关系，这里面还有两种代价
- 求所有可能性下的最小代价

I. 分组作业

- 一眼最小割题
- 给每一组建图
- 有5种情况



I. 分组作业



I. 分组作业

- 注意加权值为inf的边

J

liuzhangfeiabc

题目大意

- 给你一堆横向和纵向的线段，问你它们是否能拼成“THUPC”字样

- ~~按照题意模拟即可~~
- ~~难点：注意到这道题是可做题~~

sol

- 排序
- 合并线段
- 判断线段条数是否为15
- 暴力判断相交
- 每个连通块chk一下是哪个字母

K. 数正方体

- 题目大意：给一张字符画，问字符画上面有多少个正方体。

K. 数正方体

- 本场比赛的签到题
- 我们都知道，数正方体有一种方法叫做头顶标数法
- 比如说这个样例
- 三层标上3
- 二层标上2
- 一层标上1
- 全部加起来，结果等于14
- 但是我们还有比模拟这个更好的方法

K. 数正方体

- 我们甚至不需要直到每一列有多少个正方体，即A矩阵长什么样
- 我们有用的第一个信息是，A矩阵全正
- 也就是说，我们可以数最后一行的字符'.'的个数来确定A矩阵的高
- 最后一行前面不是点的字符除以4就是A矩阵的宽
- 由于A矩阵有一个单调的性质，所以所有顶面都露出来
- 于是我们可以通过字符串"/ /"来定位所有的顶面
- 用底面的纵坐标之和减去顶面的纵坐标之和再除以高度
- 底面的纵坐标之和可以简单计算

K. 数正方体

- 代码只有不到1K

```
char map[555][444];
int main()
{
    int r=io::F(),c=io::F();
    for(int i=1;i<=r;++i,getchar())
        for(int j=1;j<=c;++j)
            map[i][j]=getchar();
    int p=strstr(map[r]+1,".")-map[r];
    int m=p-2>>2,n=c-p+1>>1;
    int ans=m*n*(r-n);
    for(int i=2;i<=r;++i)
    {
        int now=1;
        char* tmp;
        while(tmp=strstr(map[i]+now,"/   /"))
            ans-=i,now=tmp-map[i]+4;
    }
    printf("%d\n",ans/3);
    return 0;
}
```