杭电多校5题解

1001 Tetrahedron(A)

算法1：海伦公式

- 设斜面三条边长度为x,y,z，面积为S，四面体体积为V
- 则$V=abc/6=Sh/3$，两边同时平方得到：

$$\frac{1}{36}a^2b^2c^2=\frac{h^2S^2}{9}$$

- 海伦公式

$$S=\sqrt{p(p-x)(p-y)(p-z)},\quad p=(x+y+z)/2$$

- 代入得：

$$16a^2b^2c^2=4\,h^2 2p(2p-2x)(2p-2y)(2p-2z)$$
$$16a^2b^2c^2=4h^2(x+y+z)(-x+y+z)(x-y+z)(x+y-z)$$

- 直接化简右边计算量较大，由于其是轮换对称的，考虑含x的项：x^4,x^2y^2的系数
  - 首先x的奇数次幂系数为0，因为$-(x+y+z)(-x+y+z)(-x+y-z)(x+y-z)$，注意到前两项$x$加的内容是相同的，但是$x$本身符号不同，所以前两项中的$x$要么同时取，要么同时不取，否则就会被对方抵消，后两项也一样
  - x^2yz系数为0，若取前两项的x，则是$-2yz$；取后两项的x，则是$2yz$，故是0
  - x^4系数为-1
  - x^2y^2系数为2，同样讨论两个x取自前两项还是后两项
  - 故

$$(x+y+z)(-x+y+z)(x-y+z)(x+y-z)=-\sum_{cyc}x^4+2\sum_{cyc}x^2y^2=\sum_{cyc}x^2(2y^2-x^2)$$

- 勾股定理：

$$x^2=a^2+b^2$$
$$y^2=a^2+c^2$$

- 代入得：

$$\sum_{cyc}x^2(2y^2-x^2)=\sum_{cyc}(a^2+b^2)(a^2-b^2+2c^2)$$
$$\sum_{cyc}(a^2+b^2)(a^2-b^2)+\sum_{cyc}2(a^2+b^2)c^2$$
$$=2\sum_{cyc}(c^2a^2+b^2c^2)=4\sum_{cyc}a^2b^2$$
$$16a^2b^2c^2=16h^2\sum_{cyc}a^2b^2$$

- 可得到$1/h^2=1/a^2+1/b^2+1/c^2$。

- $E(1/h^2)=3E(1/a^2)$。
- 先线性预处理逆元的平方，维护前缀和即可。
- 时间复杂度$O(n)$。

算法2：向量叉积

考虑在题面的图中，我们令小写字母a,b,c表示三直角边长度，而字母$\vec{a},\vec{b},\vec{c}$表示从上向下方向的三维向量，例如$\vec{a}=\overrightarrow{DA}$

- 考虑底面积为$|\overrightarrow{AB}\times\overrightarrow{AC}|$
- $$\overrightarrow{AB}\times\overrightarrow{AC}=(\vec{b}-\vec{a})\times(\vec{c}-\vec{a})=\vec{b}\times\vec{c}-\vec{a}\times\vec{c}-\vec{b}\times\vec{a}$$
- 考虑这三个叉积得到的向量方向是两两垂直的，故根据**勾股定理**：
- $$|\overrightarrow{AB}\times\overrightarrow{AC}|^2=|\vec{b}\times\vec{c}-\vec{a}\times\vec{c}-\vec{b}\times\vec{a}|^2=|\vec{b}\times\vec{c}|^2+|\vec{a}\times\vec{c}|^2+|\vec{b}\times\vec{a}|^2$$
- 考虑其含义：底面三角形面积^2=侧面积1^2+侧面积2^2+侧面积3^2，即：
$$4S^2=a^2b^2+a^2c^2+b^2c^2$$
- 考虑等积法：
$$a^2b^2c^2=4S^2h^2$$
- 同样得到：
$$\frac{1}{h^2}=\frac{1}{a^2}+\frac{1}{b^2}+\frac{1}{c^2}$$
- 后面算法是一样的

1002 Funny String(B)

- 注意到询问是独立的，每次处理询问时，可以在原后缀排名的基础上，考虑修改后对询问后缀的影响
- 因此先处理出s的后缀数组
- 分类讨论修改造成的影响:
- 对于操作类型一，产生一个新后缀$s[1,n+1]$
  - 若询问新后缀$s[1,n+1]$的排名，可在原sa数组上二分其排名;
  - 否则，借助$rank$数组的信息，直接比较询问后缀$s[i,n+1]$和新后缀的大小关系即可
- 对于操作类型二，产生了一个新后缀$s[n+1,n+1]$，且每个原后缀末尾新增字符c
  - 若询问新后缀$s[n+1,n+1]$的排名，统计有多少后缀以小于c的字符为首(即s中小于c的字符数)即可
  - 否则，对于询问后缀$s[i,n+1]$，考虑有多少原先排名在其后的后缀排到了它前面，有多少原先排名在其前的后缀排到了它后面，统计两者数量以修正排名
  - 对于前者，应统计这样的后缀$s[j,n]$，满足:
    - $s[i,n]$是$s[j,n]$的前缀
    - 设len_i为$[i,n]$的长度，$c>s[j+len_i]$
  - 观察可以发现，这样的$s[j,n]$在sa数组中的位置一定是排在$s[i,n]$后连续的一段，因此二分最后一个满足条件的$s[j,n]$的位置，即可统计数量，简要证明：

        - 对于三个不同的后缀A,B,C，满足在sa中$A<B<C$，我们证明如果$C+c<A+c$，即C飞跃了A，那么也有$B+c<A+c$
        - $C+c<A+c$等价于A是C的前缀，且C中A前缀后紧跟的字符$p<c$
        - 而$B>A$同样也满足$lcp(A,B)=|A|$，否则$B>C$矛盾
        - 故B中A前缀后紧跟的字符$q\le p<c$，即$B+c<A+c$，单调性得证
    - 对于后者，应统计这样的后缀$s[j,n]$，满足:
        - $s[j,n]$是$s[i,n]$的前缀
        - 设len_j是$s[j,n]$的长度，$c>s[i+len_j]$
    - 由于要寻找后缀$s[i,n]$的所有合法前缀，因此想到对s的反串进行kmp预处理，这样对于任意后缀跑$next$指针，可以找到所有是其前缀的$s[j,n]$，从而统计满足$c>s[i+len_j]$的数量
    - 但每次询问都暴力遍历所有的$s[j,n]$的复杂度并不优秀
    - 我们想象这样一棵树:
        - 树上每个节点p对应s的一个后缀$s[p,n]$
        - 对于每个后缀$s[p,n]$，记其$next$指针指向的后缀为$s[next_p,n]$，则$next_p$是p的父节点
        - 记$s[next_p,n]$的长度为L，则节点p的点权为字符$s[p+L]$
    - 对于$s[i,n]$我们要统计后缀$s[j,n]$的数量，可以转化为统计节点i走向根节点的路径中，点权严格小于c的节点数量
    - 下面说明这样转化为何正确
        - 设$s[i,n]$的$next$指向后缀$s[next1_i,n]$，记其长度为len_1
        - 设$s[next1_i,n]$的$next$指向后缀$s[next2_i,n]$，记其长度为len_2
        - 直观上看，我们应统计$s[i+len_1]$，与$s[i+len_2]$的字符是否严格小于c
        - 但其实，由于公共前后缀的性质，$s[i+len_2]$与$s[next1_i+len_2]$的字符是相同的，所以$s[i+len_2]$已经在处理$s[next1_i,n]$时贡献过了
        - 所以每个后缀$s[i,n]$贡献的点权设置为字符$s[i+len_1]$，其余的已在其祖先节点贡献过了
        - 因此，从$s[i,n]$跑到根节点，沿途统计点权小于c节点数量，即可统计所有满足条件的$s[j,n]$
    - 此时，问题转化为询问某一节点到根节点的路径上，有多少点权小于c的节点
    - 由于强制在线，我们可以在树上建立可持久化值域线段树来加速查询
- 综上所述，若用倍增处理后缀数组，则总复杂度为$O(n\log n+q\log n)$

1003 Boring Game(C)

- 想象n张纸展开的情况，每次截取上半部分，倒置(左转90度)后将其放置于剩余部分的左侧，模拟k次即可
- 下面给出题面样例的展开过程

$$\begin{bmatrix}1\\2\\3\\4\\5\\6\\7\\8\\9\\10\\11\\12\\13\\14\\15\\16\end{bmatrix} \rightarrow \begin{bmatrix}8 & 9\\7 & 10\\6 & 11\\5 & 12\\4 & 13\\3 & 14\\2 & 15\\1 & 16\end{bmatrix} \rightarrow \begin{bmatrix}12 & 5 & 4 & 13\\11 & 6 & 3 & 14\\10 & 7 & 2 & 15\\9 & 8 & 1 & 16\end{bmatrix}$$

- 复杂度$O(n \times k \times 2^k)$

1004 Expression(D)

复合函数求偏导公式：

$$u = \phi(t)$$
$$v = \psi(t)$$
$$z = f(u, v)$$
$$\frac{dz}{dt} = \frac{\partial z}{\partial u}\frac{du}{dt} + \frac{\partial z}{\partial v}\frac{dv}{dt}$$

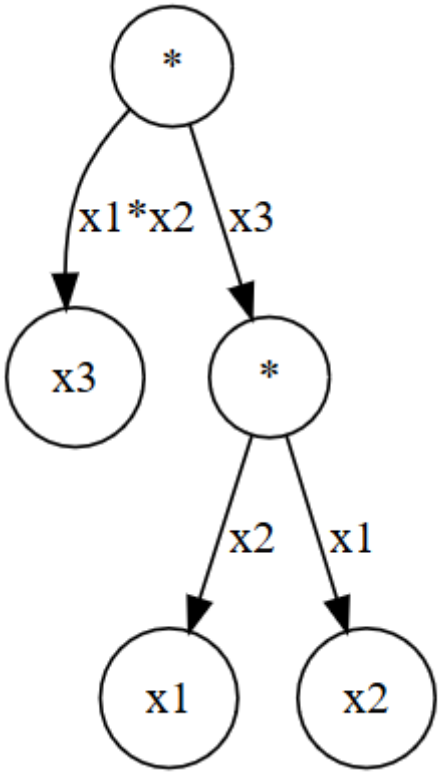
考虑对于给定的表达式，建一棵表达式树，表达式的值即为$g(0)$，因为每个变量只出现一次，n个节点则对应表达式树上的n个叶子节点；表达式树上每个节点对应的也是一个表达式f_i， 对于节点u，对应的符号为op，如果他的左右儿子节点分别为x, y， 那么$f_u = f_x op f_y$ 。

问题1

由复合函数求偏导的链式法则可知，对于一个非叶子节点，我们只需要计算出该节点的表达式对左右子节点的表达式的偏导即可，将求得的偏导表达式，当作边权。那么原表达式对于变量x_i 求偏导，即等于根到对应的叶子节点路径上的边权的表达式的乘积。

$$op = *, \frac{\partial fu}{\partial fx} = fy, \frac{\partial fu}{\partial fy} = fx$$
$$op = +, \frac{\partial fu}{\partial fx} = 1, \frac{\partial fu}{\partial fy} = 1$$
$$op = -, \frac{\partial fu}{\partial fx} = 1, \frac{\partial fu}{\partial fy} = -1$$

对于表达式$x_1 * x_2 * x_3$，建立的表达式树以及对应边权如下



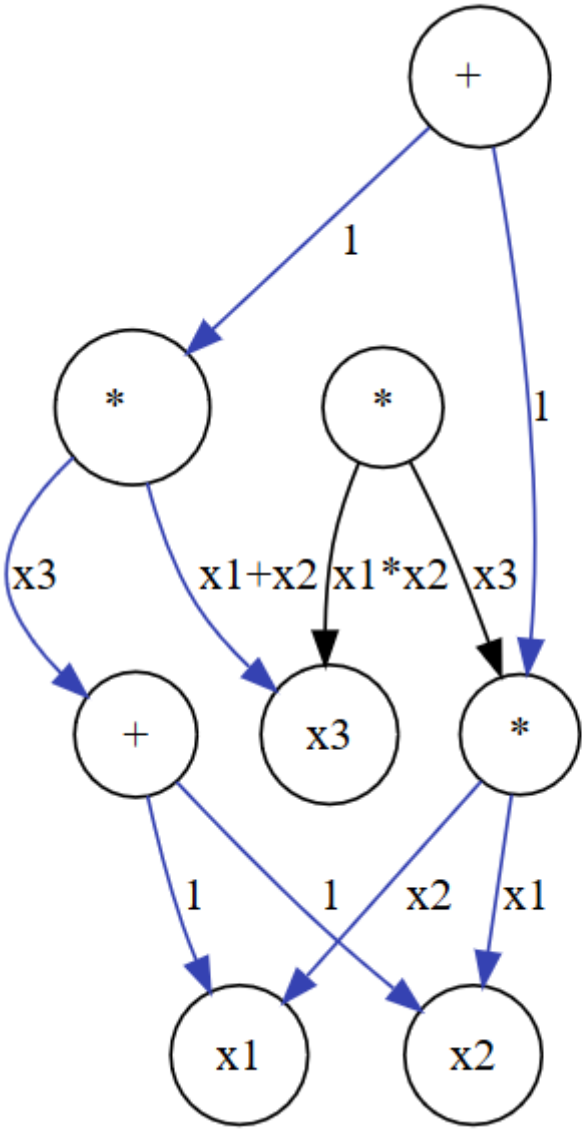
现在考虑，对所有变量求偏导的表达式的和$g(1)$，即所有根到叶子节点的路径上边权的表达式的乘积和；

- 令$dp[u]$ 表示以u为根对应的子树，所有u到叶子节点路径上边权的表达式的乘积和
- 转移为$dp[u] = w_{v1}dp[v1] + w_{v2}dp[v2]$ ，最后根节点的dp式子即为$g(1)$的表达式
- 因此通过dp转移，可以在$g(0)$的表达式树上建立$g(1)$的表达式树，对该表达式树求值则为$g(1)$的值
- 那么对$g(1)$的表达式树，进行类似的求偏导过程，即可得到$g(2)$，如法炮制依次算出$g(3), g(4), g(5)$。

对于表达式

$$f(x_1, x_2, x_3) = x_1 * x_2 * x_3$$

$$g(1) = (x2 + x1) * x3 + x1 * x2$$



- 因为边权只可能是$-1,1,0,f_i$，可以可持久化建树
- 边权不用显示的算出具体的值，可以用对应的表达式树上的节点号来表示
- 最后再dfs一遍计算表达式的值得到答案。由$g(i)$构造出$g(i+1)$的表达式树，对表达式树求值即可。
- 时间复杂度$0(3^5n)$

问题2

对$x_{i1},x_{i2}....x_{it}$ 求t阶偏导

- 由于每个变量只出现一次，如果$x_{i1},x_{i2}....x_{it}$中出现重复变量，那么结果一定为0
- 对于$x_{i1},x_{i2}....x_{it}$都不同的情况，考虑依次对求$x_{i1},x_{i2}....x_{it}$偏导，设$G(x_i)$为x_i到根的路径上边权的表达式的乘积
  - $t=1$，由上可知，对于一个变量求一阶导数的表达式为根到对应的叶子节点路上的边权的表达式的乘积，表示为

$$\frac{\partial f}{\partial x_i}=f_i*f_j....f_p*f_k*f_q....*f_s=G(x_i)$$

- $t=2$， 对x_i,x_j 求偏导
  - 如果表达式树上$LCA(x_i,x_j)$ 节点处的符号为$+$，$-$，那么偏导值必定为0
  - 如果表达式树上$LCA(x_i,x_j)$ 节点处的符号为$*$，考虑先对x_i 求偏导，得到的一阶偏导的表达式由若干个边权的表达式相乘得到

- 再对一阶偏导的表达式求对x_j求偏导的结果，一阶偏导的表达式为中唯一包含x_j 的项为$LCA(x_i,x_j)$节点处，连向x_i所在子树的边权的表达式f_k，则

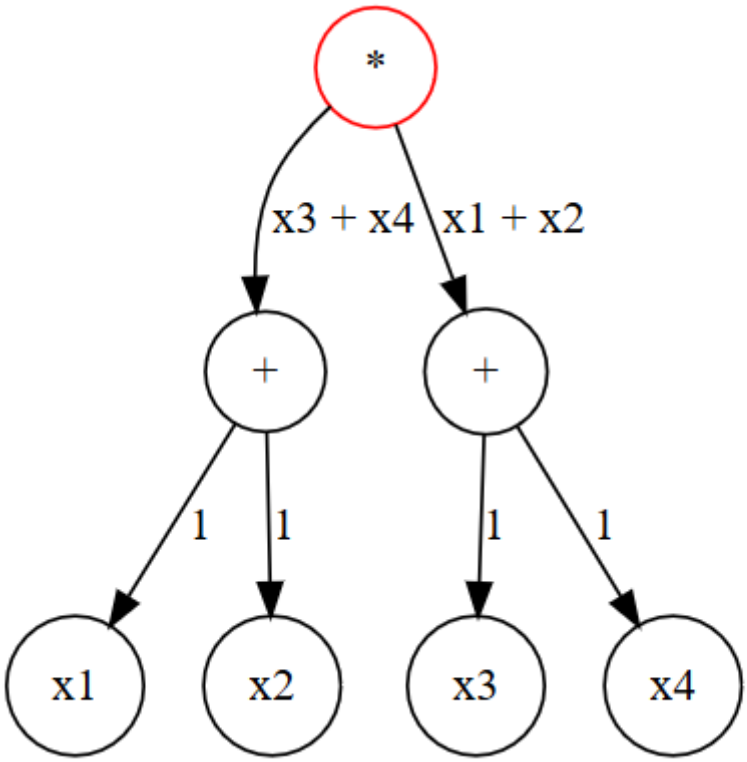
$$\frac{\frac{\partial f}{\partial x_i}}{\partial x_j}=f_i*f_j....*f_p*f_q*....*f_s*\frac{\partial f_k}{\partial x_j}$$

- 由上可得，$\frac{\partial f_k}{\partial x_j}$ 则为f_k在表达式树上对应的节点到x_j对应的节点的路径上边权的表达式的乘积
- 设$LCA(x_i,x_j)$ 连向x_i,x_j节点所在的左右子树边权为f_k,f_h ，则

$$\frac{\partial f}{\partial x_i\partial x_j}=\frac{G(x_i)*G(x_j)}{G(LCA(x_i,x_j))*f_k*f_h}$$

例：

$$f(x_1,x_2,x_3,x_4)=(x_1+x_2)*(x_3+x_4)$$
$$G(x_2)=(x_3+x_4)*1,G(x_3)=(x_1+x_2)*1$$
$$\frac{\partial f}{\partial x_1\partial x_2}=\frac{\{(x3+x4)*1\}*\{(x1+x2)*1\}}{(x_3+x_4)*(x_1+x_2)}=1$$



- $t>2$时，本质类似于**树链乘积并问题**
  - 先对变量按照表达式树上的dfs序进行排序，得到的序列为$x_i,x_j,x_h,...x_s$

- 如果对徐序列中任意两个相邻的变量，$LCA(x_j,x_h)$处的符号不为$*$，那么偏导值一定为0
  - 否则依次对一个个变量求偏导，由上可知，对前两个变量求偏导，结果为

$$\frac{\partial f}{\partial x_i\partial x_j}=\frac{G(x_i)*G(x_j)}{G(LCA(x_i,x_j))*f_k*f_h}$$

  - 这个结果还是为若干个边权的表达式的乘积，考虑这个表达式中包含x_h的项，那么有且仅有$LCA(x_j,x_h)$节点指向x_j和x_h所在子树的边权的表达式分别为f_u,f_t，那么只有f_u中会包含x_h项，f_u对x_h求偏导为f_u在表达式树上对应的节点到x_h路径上的边权乘积。那么

$$\frac{\partial f}{\partial x_i\partial x_j\partial x_h}=\frac{G(x_i)*G(x_j)*G(x_h)}{G(LCA(x_i,x_j))*f_k*f_h*G(LCA(x_j,x_h))*f_u*f_t}$$

    - 如上步骤，按dfs序依次对变量求导，即可算出最后的值，函数$G(x_i)$可通过预处理算出，因此只需要求LCA即可，因为边权可能为0，所以需要重载一下除法
        - 记录一下当前式子中乘上的0的数量，以及非零项的乘积
    - 除0时，将0的数量减1
        - 最后答案中0的数量不为0，则值为0，否则为非零项的乘积
- 时间复杂度为$O(n\log n+\sum t)$

1005 Array Repairing(E)

- 记随机生成的序列为$a[1..n]$，我们先考虑给定一个a的最小花费
- 首先，$k=0$答案一定为0，下面只考虑$k\geq 1$
- 考虑任意一种最优的操作序列方案，总是可以通过调整操作顺序，使得先做完所有第二种操作(注意下标可能随之变化)
- 令cnt为a中未出现的数值种数，容易证明第二种操作次数一定为cnt
- 因此我们需要计算$E(cnt)$，第二种操作期望代价为$k\times E(cnt)$
    - 令01随机变量$x_i,i\in[1,n]$，表示i这种数值有没有出现，没有出现令$x_i=1$
    - $P(x_i=1)=\left(1-\frac{1}{n}\right)^n$
    - $cnt=\sum_{i=1}^n x_i$
    - $E(cnt)=\sum_{i=1}^n E(x_i)=\sum_{i=1}^n P(x_i=1)=n\left(1-\frac{1}{n}\right)^n$
- 我们还需要决策这cnt次第二种操作如何进行，不同的方案，可能会影响后面第一种操作的次数
- 为了搞清楚这个问题，我们不妨假设第二种操作已经做完，即现在a是一个排列，考虑仅用第一种操作的最小次数
    - 考虑连边$i\to a[i]$，其结构是若干个圈
    - 对于一个圈长为L的圈，其最小交换次数为$L-1$
    - 故答案为：$\sum_{i=1}^k(L_i-1)=n-num$，这里$num$是圈个数
- 因此，我们需要执行cnt次第二种操作，使得k最大
- 类比经典做法，我们也连边$i\to a[i]$，这时结构是基环内向森林，且每个联通块都有一个环，包括自环，一共有cnt个叶子
- 我们一共有cnt次改边操作，显然最多新增cnt个圈
- 且可以构造一个操作方案，使得取到这个值
    - 每次考虑选择叶子节点x，沿着有向边走到第一个这样的点y，它的父亲p，有至少2个入边
    - 将$y\to p$改为$y\to x$，新建一个圈，且减少一个叶子
    - 最终，没有叶子(入度为0)，n个点n条有向边只能是若干个圈的结构，回到经典模型
- 令a中本身有X个联通块，即有X个圈，则第一种操作最小总代价为$n-cnt-X$
- 故$k\geq 1$，$Cost_k(a)=n+(k-1)cnt-X$
- 现在计算$E(X)$，即n个点的带标号随机基环内向森林(每个点连向n个点中的一个，包括自己)，所包含联通块的期望个数$E(X)=h(n)/n^n$：
    - 先考虑计算n个点的仅有一个联通块的方案数$f(n)$
        - 如果联通块数量可以任意，则显然有$g(n)=n^n$种
        - 则$g(n)$的指数型生成函数$G(x)=\sum_{n\geq 0}n^n\frac{x^n}{n!}$，注意$0$个点看成是$1$种
        - 令$f(n)$的指数型生成函数为$F(x)=\sum_{n\geq 0}f(n)\frac{x^n}{n!}$，注意$0$个点看成是$0$种
        - $G(x)=\exp(F(x))$，故$F(x)=\log(G(x))$
        - 证明推导考虑exp函数的泰勒展开之后用组合意义解释

    - 容易得到$H(x)=F(x)G(x)$，这里仅介绍一种计算贡献的方法，$H(x)$是$h(n)$的指数型生成函数
        - 考虑一个联通块ω自身，它可以在多少种图中出现，则它的贡献即为多少
        - 先枚举ω的点数$k=|\omega|\in[1,n]$，然后用组合数选择方案$\binom{n}{k}$，这些点构成一个联通块的基环树有$f(k)$种
        - 至于剩下的$n-k$个点，它们自己自由连接，显然有$g(n-k)=(n-k)^{n-k}$种
        - 故$h(n)=\sum_{k=1}^{n}\binom{n}{k}f(k)g(n-k)$
        - $h=f\circ g$，即$G(x)=F(x)G(x)$
    - $h(n)$是总和，$E(X)=h(n)/n^n$即为期望
- 最终答案：$E(Cost_k(a))=n+(k-1)E(cnt)-E(X)$，$k\geq 1$
- 时间复杂度：$O(N\log N)$

1006 Alice and Bob(F)

- 设共有 k个桶，第i个桶中的一个数x表示，取出这个数x接到序列A最后，会形成的最长的以这个数x为结尾的<u>最长上升子序列</u>长度为恰好为i。
- 游戏初始时，所有数都在编号为1的桶里。对于每次操作等价于，**<u>取出某个桶里的某个数删除并将该桶中比删除数更大的数移动到下一个桶</u>**（当然最后一个桶例外，因为无法移动元素，只能取最大的数，如果最后一个桶不为空，详细需要结合游戏的版本）。
- 对于已经进行的q次操作，可以使用树状数组逐个元素x考虑，计算出此时每个桶里的数的数量$num[i]$。

对于版本一

- 如果$num[k]>0$，取出第k个桶里的数即可，方案数即$num[k]$。
- 如果$num[k]=0$，不能将第$k-1$里的数后移（否则对方直接获胜），即只能删除$k-1$桶里最大的数，所以双方都不能凑出长度为k的子序列，最终会因为数字被取完而游戏结束。
    - 从整体上看，每次操作总的个数减1（即$\sum num[i]-=1$），**所以如果剩下的个数为奇数，$Alice$ 获胜，否则失败。**
    - 方案个数，前$k-2$个桶中的数是任意取的，**而第$k-1$个桶如果有多个数，也只能去最大的那个**，所以方案数为$\sum_{i=1}^{k-2}num[i]+(num[k-1]>0)$

对于版本二

- 先对每个桶可行操作进行说明：
    - **第k个桶的数是不能取的，桶里的数相当于被删除了。**
    - 对于第$k-1$个桶，每次操作会先删除一个数，再将比该数大的数移动到第k个桶。所以对$k-1$个桶的操作相当于可以删除这个桶中最大的若干个数（正整数个数）。
    - 对于前$k-2$个桶，对这些桶的操作和版本一的情况相同。每次操作后，剩余可选数的总数量的奇偶会变化。
- **如果$num[k-1]\geq 2$ 先手必胜**，且只有一种操作方案。该操作为，删除第$k-1$个桶的若干个数，使得第$k-1$个桶剩余个数小于2（非0即1）且前$k-1$个桶内个数的和为偶数，其他的操作都是必败的。

- 后续Bob无论如何操作，在他操作后，前$k-1$个桶的数的个数和必然变为奇数，或者他根本无法操作，因为如果他选择第$k-1$个桶，该桶也只有严格1个元素。
- **如果$num[k-1]<2$，若前$k-1$个桶数的个数和为奇数则$Alice$必胜，否则必败。**
  - 考虑必胜的先手第一次操作：**先手不能使得$num[k-1]\geq 2$。**
  - 必胜情况下，先手对于前$k-3$个桶内的数可以任取，即有$\sum_{i=1}^{k-3} num[i]$种方案。
  - 如果$num[k-1]=1$时，只能取$k-1$桶里的数或者$k-2$桶里最大的数（如果有），即$1+(num[k-2]>0)$。
  - 如果$num[k-1]=0$时，只能取$k-2$桶里的最大的两个数（如果有），即要么选择次大的数，移动一个数(最大数)，要么选择最大的数，不移动数，否则会导致$num[k-1]\geq 2$，即最多两种方案：$(num[k-2]>0)+(num[k-2]>1)$。
  - 方案数即前三个点讨论的和。
- 选手可以用**数学归纳法**对上述结论综合严密地论证，对$\sum_{i=1}^{k} num[i]$进行归纳。

1007 Tree(G)

- 首先$k=0$时答案显然为0
- 对于最终所求的连通块，要求不超过一个点的度大于k，其余点的度均小于等于k的树，采用树形dp。
- 首先将原树看成一颗有根树，根节点任意：
  - 我们假设$dp[x][0]$表示x为根节点，所有节点儿子数量均小于等于$k-1$的点的子连通图最大边权和。
  - $dp[x][1]$表示x点为根节点，有不超过1个节点儿子数量大于$k-1$，其余节点儿子个数等小于等于$k-1$的子连通图最大边权和。
  - 只要保证点儿子数量均不超过$k-1$，那么显然该点度不会超过k。
- 转移方程：
  - 假设 x 有c个儿子，表示为$s_1,s_2....s_c$，我们首先将儿子节点按 $dp[s_i][0]$ 由大到小排序。
  - 可以得到$dp[x][0]=\sum_{i=1}^{k-1} dp[s_i][0]+w_{xs_i}$，其中$w_{xs_i}$表示$x$到$s_i$点的边的边权。
  - 而对于$dp[x][1]$，由于度大于k的不超过一个，那最多只能有1个点儿子数量大于$k-1$
    - 若该点为x点，则有
    $$dp[x][1]=\sum_{i=1}^{c} dp[s_i][0]+w_{xs_i}$$
    - 而若该点为x点的子孙节点，则选取x的$k-2$个儿子的$dp[0]$以及1个儿子的$dp[1]$，可以先贪心的将最大的$k-2$个$dp[0]$取下，由于$dp[1]$只需要取一个，那么有两种取法：
      - 第一种为取前$k-1$个点中的某点的$dp[1]$，那么$dp[0]$就少了一个，需要将$dp[s_{k-1}][0]$一并选取，即有
      $$dp[x][1]=\sum_{i=1}^{k-1}(dp[s_i][0]+w_{xs_i})+max(dp[s_j][1]-dp[s_j][0]),j\in[1,k-1]$$
      - 第二种为取$k-2$个节点后的某个节点，有
      $$dp[x][1]=\sum_{i=1}^{k-2}(dp[s_i][0]+w_{xs_i})+max(dp[s_j][1],j\in[k-1,c])$$

- 最终方案显然是可以选取**某个点的**$dp[x][1]$**，但是存在一种情况，即选取了某一个节点和其**k**个子树，而没有将联通块和该节点的父亲节点相连**，这时候答案为该节点的$k-1$个儿子节点s_i的$dp[s_i][0]$以及一个儿子节点的$dp[s_i][1]$组成，做法和求该节点$dp[x][1]$相同，只是多选了一个儿子节点的$dp[0]$而已，在dfs的时候顺便求了即可。
- 时间复杂度为$O(n\log n)$

1008 Set2(H)

- 问题：给定一个$1-n$的集合S，每次删除当前集合中最小的元素，再考虑**顺序的**随机删掉k个元素，直到$|S|\le k$，求每个元素最后被留下来的概率。
- $r=n\%(k+1)$，如果$r=0$，所有元素留下来的概率都是0
- 如果$n<k+1$，所有元素留下来的概率都是1
- 观察可知，如果最后留下的元素中，最小元素为x，那么$1,2,3...x-1$，一定全被删除了
- 将操作看成两种
  - 第一种，每次删掉一个最小的元素
  - 第二种，每次随机删除一个元素
- 设$dp[i][j]$ 为前i个元素全被删除后第二种操作还需要做j的方案数，转移如下
  - 做第一种操作，如果$j+k\le n-i-1$：
$$dp[i+1][j+k]=dp[i+1][j+k]+dp[i][j]$$
  - 做第二种操作，如果$j>0$：
$$dp[i+1][j-1]=dp[i+1][j-1]+dp[i][j]\times j$$
- 考虑最后留下来的元素组成的集合T，从小到大为$x_1,x_2,\dots x_r$，将所有方案按照集合T中最小元素x_1进行分类
  - 令$f[i]$ 表示集合T中最小元素为i的方案个数
$$j=n-i-1-r$$
$$f[i]=dp[i-1][j]\times j!\times\binom{n-i}{j}$$
  - 令$cnt[i]$ 表示集合T中最小元素为i时，元素$x>i$ 被留下来的方案数
$$j=n-i-1-r$$
$$cnt[i]=dp[i-1][j]\times j!\times\binom{n-i-1}{j}$$
  - 令前缀和$sum[i]=\sum_{j=1}^{i}cnt[i]$
- 那么首先总方案为$SUM=\sum_{i=1}^{n}f[i]$
- 元素i被留下来的方案数$ans[i]=f[i]+sum[i-1]$，而概率$p[i]=\frac{ans[i]}{SUM}$
- 时间复杂度$O(n^2)$

1009 Paperfolding(I)

- 模拟一下可以看出水平对折和垂直对折的答案相对独立

- 对于x 次水平对折和y次垂直对折
- 答案是$(2^x+1)(2^y+1)$
- 因为通过反向逐操作还原，可以看到刀的痕迹的数量变化是每次在某一维倍增的
- 因此，相当于一张纸，水平和竖直分别切了$2^x,2^y$刀
- 所以数学期望为$E(x)=\frac{1}{2^n}\sum_{i=0}^{n}C_n^i(2^i+1)(2^{n-i}+1)=1+2^n+2\times 3^n/2^n$

1010 Function(J)

- 这里仅给出一个思路，本题考虑方式有很多。
- 我们首先可以用**高斯消元(线性基)**求出由$\{a_1,a_2,\cdots,a_n\}$生成的空间S的基底$base(S)$。
- 我们记由$f(x)=0$生成的解空间为M。
- 设$b_1,b_2,\ldots,b_m$为M的基$base(M)$，$b_{m+1},b_{m+2},\cdots,b_{m+k}$为$S-M$的基$base(S-M)$。
- 这里的减号表示M和$S-M$之间的和是**直和**。
- 则对于任意的b_i，$i\in\{1,2,..,m\}$，至少存在一个j，$j\in\{m+1,m+2,\ldots,m+k\}$使得$f(b_j)=b_i$，所以$k\geq m$
- 类似地我们可以推出$m\geq k$。
- 由此得到：**当且仅当$m=k$时存在某种合法构造方式。**
- 考虑如何构造：
  - 我们只需令$f(b_i)=0,f(b_{i+m})=b_i,i\in[1,m]$即可。
- 考虑如何检验构造的解是否正确，下面仅讨论有解时候的情况：
  - 我们先剔除$x\notin S$的情况
  - 我们假设由所有的x，$f(x)$生成的空间为S'
  - 对于$x\in S$，有$x=b_{p_1}\oplus b_{p_2}\oplus\ldots\oplus b_{p_{m'}}$，其中$p_i\in[1,m+k]$且$p_i\neq p_j\ if\ i\neq j$。
  - 我们可以据此将x分解，首先将$f(x)=0$提取出来，并用这些x生成M'。
  - 若存在$x\in M$但$f(x)\neq 0$，显然构造错误。
  - 注意到一个细节：对于$f(x)=y\neq 0$时，我们可以得到$f(y)=0$，我们同样需要$y\in M'$。
  - 这时候我们考虑$f(x)\neq 0$的情况，这时候我们可以先将x含有M'的基底处理掉，这是因为$x'\in M'$ 那么$f(x\oplus x')=f(x)$。
  - 此时可以得到$S'-M'$，再将所有$(x,f(x))$放在**增广矩阵**中进行**高斯消元**。
  - 如果在消元之后，得到了$f(0)\neq 0$的情况，那么构造必然是错误的。
  - 最后检查$S'-M'$和M'的空间是否分别包含于$S-M$和M即可
    - 即$base(S'-M')$的元素个数小于$base(S-M)$，$base(M')$的元素个数小于$base(M)$。
- 时间复杂度$O((n+m)\log 2^{30})$

1011 Exam(K)

- 题面中通过考试i必须通过两段时间中的其中一场，我们设$[a,a+at]$选择与否为变量i，$[b,b+bt]$选择与否为变量i'，很简单的就能联想到$2-sat$问题的选i与选i'。

- 但是题目要求输出完成所有考试的最小时间，而$2-sat$只能解决有解和无解的问题。考虑二分完成的所有考试最小的时间，如果在现在二分的值下有解，则答案至多为现在二分的这个值。
  - $\rightarrow$表示连边，假设当前二分的值为mid，如果$a+at>mid$，则说明考试i'是必选的，$i\rightarrow i'$，如果$b+bt>mid$，则说明考试i是必选的，$i'\rightarrow i$。
  - 剩下只需要将有冲突的考试进行连边。
  - 如果$[a_i,a_i+at_i]$与$[a_j,a_j+at_j]$有交，则说明是冲突的，$i\rightarrow j',j\rightarrow i'$。
  - 如果$[a_i,a_i+at_i]$与$[b_j,b_j+bt_j]$有交，则说明是冲突的，$i\rightarrow j,j'\rightarrow i'$。
  - 如果$[b_i,b_i+bt_i]$与$[a_j,a_j+at_j]$有交，则说明是冲突的，$i'\rightarrow j',j\rightarrow i$。
  - 如果$[b_i,b_i+bt_i]$与$[b_j,b_j+bt_j]$有交，则说明是冲突的，$i'\rightarrow j,j'\rightarrow i$。
  - 显然直接暴力建边的话时间和空间复杂度都是n^2的。
- 对于每一场考试i，我们发现只需要考虑其他的考试j，$a_i\leqslant a_j\leqslant a_i+at_i$，就可以包括所有情况
- 如果将所有考试**按开始时间排序**，注意到与每一场考试i有冲突考试j都是一段连续的区间
- 可以通过**线段树优化建图**在$n\log n$的时间内完成建图
- 时间复杂度$O(n\log^2 n)$
- 空间复杂度$O(n\log n)$

1012 Set1(L)

- 问题：给定一个$1-n$的集合S，每次删除当前集合中最小的元素，再随机删掉1个元素，直到$|S|=1$，求每个元素最后被留下来的概率。
- 考虑元素i被留下来的方案数，前面有$i-1$个元素，后面有$n-i$个元素。
- 当前仅当$n-i\leq i-1$的时候，i才可能被留下。
- 每次删除最小的元素为操作1，随机删掉一个元素为操作2
- 大于元素i的$n-i$个元素一定是被操作2给删除的
- i被留下来的情况，一定是后面所有的元素($n-i$个)，全部被前面$i-1$个元素中的某个1对1的选择(使用操作2)
  - 那么对后$n-i$个元素，它们被选择的方案数是$\binom{i-1}{n-i}\times(n-1)!$
- 然后剩下的$(i-1)-(n-i)=(2\times i-n-1)$个元素两两删除。
- 那么i被留下来的方案数
  - $$\begin{aligned}cnt[i]&=\binom{i-1}{n-i}\times(n-i)!\times\binom{2\times i-n-1}{2,2,2,2,\ldots 2,2,2}\\&=\binom{i-1}{n-i}\times(n-i)!\times\frac{(2\times i-n-1)!}{(2!)^{(2\times i-n-1)/2}}\end{aligned}$$
    注意这里：$2\times i-n-1$是一个偶数
- 总方案数$sum=\sum_{i=1}^{n}cnt[i]$
- 第i个人被留下来的概率$p[i]=\frac{cnt[i]}{sum}$。
- 时间复杂度$O(n)$。

1013 An Easy Matrix Problem(M)

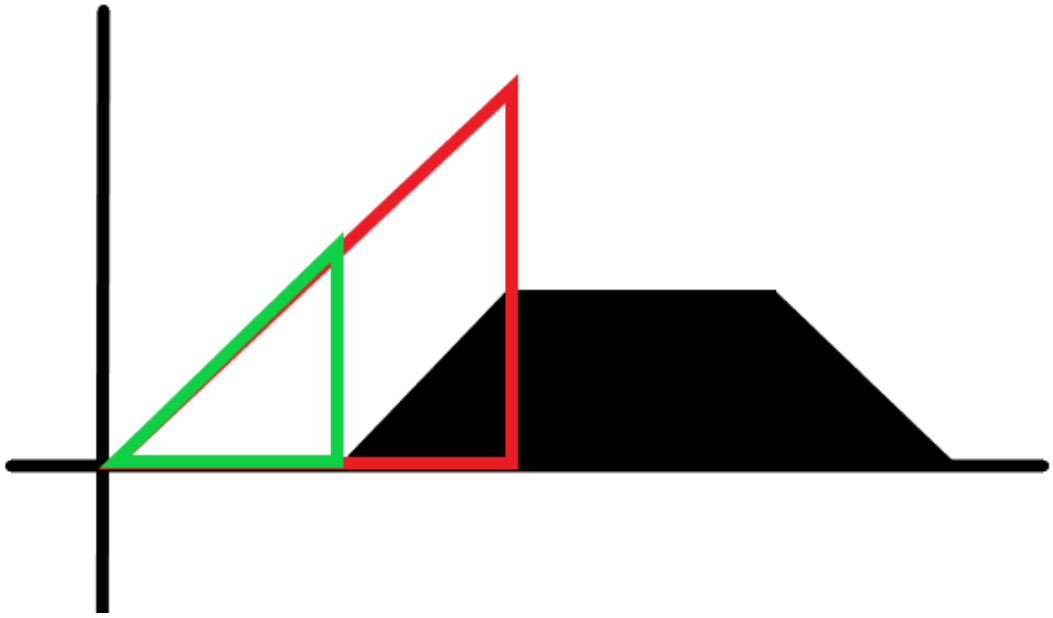
- 对于操作0和操作3，根据容斥原理，可以很容易转成4个包含点$(0,0)$的矩阵的加减(**左上角二维前缀和**)。
- 对于操作0，注意到t是一个定值，预处理出0到$n-1$的t次方，可以预处理加权前缀和直接算出答案。
  - 不妨假设$t=2,n=5$，取模后的，5×5矩阵(C中所有元素平方)如下：
  - $$\begin{bmatrix} 0 & 2 & 4 & 1 & 3 \\ 3 & 0 & 2 & 4 & 1 \\ 1 & 3 & 0 & 2 & 4 \\ 4 & 1 & 3 & 0 & 2 \\ 2 & 4 & 1 & 3 & 0 \end{bmatrix}$$
  - 设当前询问矩阵的左上角为是$(0,0)$，右下角是$(3,2)$，即左上角4×3的子矩形
  - $$\begin{bmatrix} 0 & 2 & 4 \\ 3 & 0 & 2 \\ 1 & 3 & 0 \\ 4 & 1 & 3 \end{bmatrix} \rightarrow \begin{bmatrix} & & 0 & 2 & 4 \\ & 3 & 0 & 2 & \\ 1 & 3 & 0 & & \\ 4 & 1 & 3 & & \end{bmatrix} \rightarrow \begin{bmatrix} & & & & \\ & 3 & 0 & & \\ 1 & 3 & 0 & 2 & \\ 4 & 1 & 3 & 0 & 2 & 4 \end{bmatrix}$$
  - 从下往上4出现了1次，1出现了2次，3出现了3次，0出现了3次，2出现了2次，4出现了1次
  - 如果将每个数次的出现顺序作为横坐标，数字出现次数作为纵坐标，**可以得到图形是一个梯形**
    - 中间的部分我们可以通过预处理前缀和直接得到，中间的梯形是前缀和的倍数
    - 剩下是一个斜率为1和一个斜率为-1的三角形
  - 我们将左下角的元素作为第一个出现的元素，右上角的元素作为最后一个出现的元素，处理一个元素出现顺序乘元素值的前缀和，这个前缀和是一个斜率为1三角形。
    - PS：这是上面提到前缀和的具体例子。定义b数组=$2,4,1,3,0,2,4,1,3$，下标从1开始，这个b数组是把t次方的A矩阵**左下角第一个元素开始取，取到左上角第一个元素，再取到右上角第一个元素。**

$$\begin{bmatrix} \mathbf{0} & \mathbf{2} & \mathbf{4} & \mathbf{1} & \mathbf{3} \\ \mathbf{3} & 0 & 2 & 4 & 1 \\ \mathbf{1} & 3 & 0 & 2 & 4 \\ \mathbf{4} & 1 & 3 & 0 & 2 \\ \mathbf{2} & 4 & 1 & 3 & 0 \end{bmatrix}$$

$$\begin{bmatrix} & & & & \\ & & 3 & 0 & \\ & 1 & 3 & 0 & 2 \\ 4 & 1 & 3 & 0 & 2 & 4 \end{bmatrix}$$

$$\begin{array}{rl} b: & 2\ \ 4\ \ 1\ \ 3\ \ 0\ \ 2\ \ 4\ \ 1\ \ 3 \\ s: & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ sum: & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{array}$$

- 前缀和$s_i=s_{i-1}+b_i$，加权前缀和$sum_i=sum_{i-1}+b_i\times i$
-



- 用红色的三角形减去绿色的三角形再用前缀和减去剩下的矩形，就可以得到左边三角形(黑色)。同理可以处理一个逆向的前缀和得到另外一边的三角形(黑色)。
  - 左边黑色小三角形和的公式类似于：$sum_p - sum_q - (s_p - s_q) \times h$，$h = q$
- 时间复杂度$O(n \log t)$
- 通过调整A，B矩阵的值，给某些位置$+n$或者$-n$，可以使得C矩阵是一个相邻项相差为定值的一个矩阵。
- 设$C_{0,0}$的值为x。
- $C_{i,j}$相当于把a数组循环左移$j-i$乘上B的第一列。相当于将a数组的每一位都加上$j-i$然后乘上B矩阵的第一列。
- 设相邻项相差的定值为d。矩阵C如下
  - $$\begin{bmatrix} x & x+d & x+2\times d & x+3\times d & x+4\times d \\ x-d & x & x+d & x+2\times d & x+3\times d \\ x-2\times d & x-d & x & x+d & x+2\times d \\ x-3\times d & x-2\times d & x-d & x & x+d \\ x-4\times d & x-3\times d & x-2\times d & x-d & x \end{bmatrix}$$
    - 这里的左上角x可以快速计算出，而d实际上就是读入生成矩阵B的那个**数组的元素和**
  - 先不考虑修改，假设现在询问的矩阵左上角是$(0,0)$，右下角是(x_i-1, y_i-1)，注意到每一行都是一个等差数列
  - 那就是求前x_i个等差数列的前y_i项和，等差数列与等差数列相加，还是一个等差数列
    - 新的等差数列的首项为旧的等差数列的首项相加
    - 公差为旧的等差数列的公差相加
    - 可以通过树状数组可以在$\log(n)$的时间内求出前x个等差数列的首项和以及公差和。
    - 如果$x_i = 3, y_i = 2$，就是求首项为$3\times x - 3\times d$，公差为$3\times d$的等差数列的前y_i项和。
  - 现在考虑带修改，我们发现添加的$shift(a,i)$能看成首项为$-i$，公差为1的等差数列，开两个树状数组分别维护行和列的等差数列即可。
- 时间复杂度$O(n \log n + q \log n)$