开火车(opentrain)题目选讲

ICPCCamp 2016

赵轩昂

2016 年 1 月 27 日

Tree generation[1]

Description

有这样一个生成随机数的函数wnext(a,b,type)。

```
int wnext(int a, int b, int type) {
    int result = random(a, b);
    for (int i = 0; i < type; i++)
        result = max(result, random(a, b));
    for (int i = 0; i < -type; i++)
        result = min(result, random(a, b));
    return result;
}
```

[1] AIM Fund Cup 2015, Problem K

Tree generation

Description

有这样一段生成随机树的代码。

```
int n = 10000;
int p[n];
for (int i = 0; i < n; ++i) {
    p[i] = i;
}
shuffle(p, p + n);
int type = random(-4, 4);
for (int i = 1; i < n; ++i) {
    add_edge(p[i], p[wnext(0, i - 1, type)]);
}
```

Tree generation

Description

输入数据会给出100棵上述代码生成的树，所有树的节点数都是10000。

要判定生成每棵树的参数type，只要对90个及以上即可通过。

Tree generation

```
int wnext(int a, int b, int type) {
    int result = random(a, b);
    for (int i = 0; i < type; i++)
        result = max(result, random(a, b));
    for (int i = 0; i < -type; i++)
        result = min(result, random(a, b));
    return result;
}
```

对于type=0，wrand期望返回的是a,b的中点，type越大，随机的期望就更倾向于b，type越小，期望更倾向于a。

Tree generation

```
int wnext(int a, int b, int type) {
    int result = random(a, b);
    for (int i = 0; i < type; i++)
        result = max(result, random(a, b));
    for (int i = 0; i < -type; i++)
        result = min(result, random(a, b));
    return result;
}
```

对于type=0，wrand期望返回的是a,b的中点。type越大，随机的期望就更倾向于b，type越小，期望更倾向于a。

Tree generation

考虑一下极端的情况：

- 当$type << 0$时，wrand(a，b，type) $=$ a，所有点i的父亲都是0，形成一朵花。
- 当$type >> 0$时，wrand(a，b，type) $=$ b，所有点i的父亲都是i-1，形成一条链。

所以我们需要依靠树的形态，来判定参数type的选择。

哪些特征可以作为衡量type的标准？

Tree generation

考虑一下极端的情况：

- 当$type << 0$时，wrand(a，b，type) = a，所有点i的父亲都是0，形成一朵花。
- 当$type >> 0$时，wrand(a，b，type) = b，所有点i的父亲都是i-1，形成一条链。

所以我们需要依靠树的形态，来判定参数type的选择。

哪些特征可以作为衡量type的标准？

Tree generation

考虑一下极端的情况：

- 当$type << 0$时，wrand(a，b，type) $=$ a，所有点i的父亲都是0，形成一朵花。
- 当$type >> 0$时，wrand(a，b，type) $=$ b，所有点i的父亲都是i-1，形成一条链。

所以我们需要依靠树的形态，来判定参数type的选择。

哪些特征可以作为衡量type的标准？

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Tree generation

特征可以是以下几个：

- 直径长度
- 度数的平方和
- 叶子节点数量
- ...

所以可以去按照题目给的代码自己生成一些树，看看在不同type下这些特征的期望是多少并记录下来。所以我们可以人工学习一个，对于输入的树也计算它的特征，找到特征最接近的type即可。

正确率可以达到95%左右。

Avengers, The[2]

Description

平面上有n个点，且前4个点构成一个边平行于坐标轴的矩形，所有点都在这个矩形内部，且没有三点共线。

现在有Q个询问，每个询问给出一个在矩形内部的点，要求选出平面上尽可能少的点(不少于3个)，使得询问的点在这些点构成的凸包内或边界上，并给出方案。

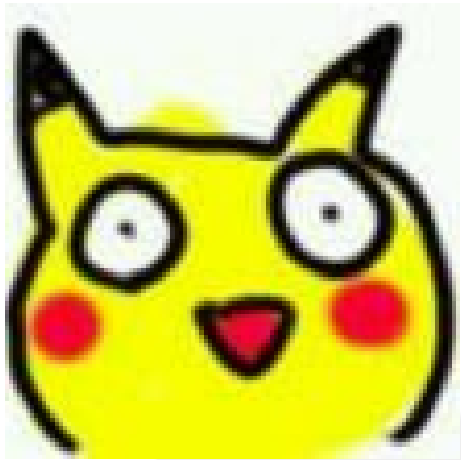
$n \leq 10^4, Q \leq 10^5$, 强制在线。

[2]XVI Open Cup named after E.V. Pankratiev, Grand Prix of Ekaterinburg, Problem A

Avengers, The

每个询问只要用3个点就足够了。

为什么呢？



Avengers, The

把矩形拆成两个直角三角形。对于一个大三角形，随机选一个内部的点，并以此点对三角形三角剖分，划分出3个小三角形，直到三角形内部没有点为止。

这样就可以得到一个树结构，对于每次询问，如果当前询问节点是叶子，就返回这个叶子节点对应的三角形，否则就看询问的点在自己儿子对应的哪个三角形中，递归下去。

单次询问$O(h) = O(\log n)$，h是树的深度。

Avengers, The

把矩形拆成两个直角三角形。对于一个大三角形，随机选一个内部的点，并以此点对三角形三角剖分，划分出3个小三角形，直到三角形内部没有点为止。

这样就可以得到一个树结构，对于每次询问，如果当前询问节点是叶子，就返回这个叶子节点对应的三角形，否则就看询问的点在自己儿子对应的哪个三角形中，递归下去。
单次询问$O(h) = O(\log n)$，h是树的深度。

Two Rectangles[3]

Description

已知两个矩形的面积之和为s，求它们的周长之和的最小值。

即有四个正整数a, b, c, d满足$a \cdot b + c \cdot d = s$，求$2(a + b + c + d)$的最小值。

$s \leq 10^{18}$

[3] XV Open Cup named after E.V. Pankratiev. GP of SPb, Problem J

Two Rectangles

$s \leq 10^9$怎么做？

如果没有要求正整数，那么显然当$a = b = \sqrt{s}, c = d = 0$时最小，所以可以猜想答案应该是由一个大矩形和一个小矩形组成的。

不妨设$a, b(a > b)$是大矩形的长和宽，从$\sqrt{s}$开始枚举a，那么可以认为$b = \lfloor \frac{s}{a} \rfloor, c \cdot d = s \mod a$。

Two Rectangles

$s \leq 10^9$怎么做？

如果没有要求正整数，那么显然当$a = b = \sqrt{s}, c = d = 0$时最小，所以可以猜想答案应该是由一个大矩形和一个小矩形组成的。

不妨设$a, b(a > b)$是大矩形的长和宽，从$\sqrt{s}$开始枚举a，那么可以认为$b = \lfloor \frac{s}{a} \rfloor, c \cdot d = s \mod a$。

Two Rectangles

$s \leq 10^9$怎么做？

如果没有要求正整数，那么显然当$a = b = \sqrt{s}, c = d = 0$时最小，所以可以猜想答案应该是由一个大矩形和一个小矩形组成的。

不妨设$a, b(a > b)$是大矩形的长和宽，从$\sqrt{s}$开始枚举a，那么可以认为$b = \lfloor \frac{s}{a} \rfloor, c \cdot d = s \mod a$。

Two Rectangles

- c, d的处理?
  如果$c \cdot d$过大肯定不优，所以可以预处理出每个$c \cdot d \leq 10^6$最小的$c + d$。
- a的枚举范围?
  $c + d$最大只有2×10^6，所以当$a + b$大到一定程度就可以跳出。
- 这样就做完啦。

Two Rectangles

- c, d的处理?
  如果$c \cdot d$过大肯定不优，所以可以预处理出每个$c \cdot d \leq 10^6$最小的$c + d$。
- a的枚举范围?
  $c + d$最大只有2×10^6，所以当$a + b$大到一定程度就可以跳出。
- 这样就做完啦。

Two Rectangles

- c, d的处理?
  如果$c \cdot d$过大肯定不优，所以可以预处理出每个$c \cdot d \leq 10^6$最小的$c + d$。
- a的枚举范围?
  $c + d$最大只有2×10^6，所以当$a + b$大到一定程度就可以跳出。
- 这样就做完啦。

Equilateral Polygon[4]

Description

构造一个边长都是1的每个内角大小都不相等的凸n边形。

两个内角大小相等指它们弧度相差不超过10^{-8}, $5 \leq n \leq 100$。

[4]XIV Open Cup named after E.V. Pankratiev. GP of Ukraine, Problem E

Equilateral Polygon

图样的做法：

先搞出一个正n边形，然后每次去找有没有相等的内角，有的话随机从中挑一个出来，把它的某条相邻的的边“挪”一个随机的角度，一直迭代到不存在相等的角为止。

但是"挪"的过程不是特别好写。

Equilateral Polygon

赛艇的做法：

先搞出有$\lfloor \frac{n+1}{3} \rfloor$个点的凸壳，且满足凸壳上每个外角都不相等。比如第一个外角是ε，第二个是2ε，第三个是3ε...。然后记录下来这个凸壳两端的距离。

对于剩下的点，一样构造一个凸壳，这时我们设定一个参数α，使得第一个外角是$\alpha+\varepsilon$，第二个是$\alpha+2\varepsilon$，第三个是$\alpha+3\varepsilon$...，这样也能得到凸壳的两端距离。

二分α让这两个凸壳能拼起来（距离相等）即可，ε不妨取10^{-4}。

Equilateral Polygon

赛艇的做法：

先搞出有$\lfloor \frac{n+1}{3} \rfloor$个点的凸壳，且满足凸壳上每个外角都不相等。比如第一个外角是ε，第二个是2ε，第三个是3ε...。然后记录下来这个凸壳两端的距离。
对于剩下的点，一样构造一个凸壳，这时我们设定一个参数α，使得第一个外角是$\alpha+\varepsilon$，第二个是$\alpha+2\varepsilon$，第三个是$\alpha+3\varepsilon$...，这样也能得到凸壳的两端距离。

二分α让这两个凸壳能拼起来（距离相等）即可，ε不妨取10^{-4}。

Rabbit Lunch[5]

Description

有n种萝卜，每种萝卜有a_i个，有m种猕猴桃，每种猕猴桃有b_i个。现在有一群兔子要吃东西，每只兔子必须要吃一个萝卜和一个猕猴桃(?)。每只兔子吃的萝卜和猕猴桃不能完全一样，问最多能满足多少只兔子？

$n, m \leq 2500000$，a, b由参数生成。

[5] Makoto Soejima Contest 3, Problem A

Rabbit Lunch

这不是**网络流嘛!

左边n个点，源往第i个点连流量为a_i的弧，右边m个点，第j个点往汇连流量为b_j的弧，中间是个流量全为1的完全二分图，最大流就是答案。

Rabbit Lunch

这不是**网络流嘛!

左边n个点，源往第i个点连流量为a_i的弧，右边m个点，第j个点往汇连流量为b_j的弧，中间是个流量全为1的完全二分图，最大流就是答案。

Rabbit Lunch

思考最小割一定是割一点a_i，割一点b_j，再割点中间的1。假设现在左边有x个a_i被割了，右边有y个b_j被割了，那么现在的割为：

$$cut = \sum_{k=0}^{x-1} a_{i_k} + \sum_{l=0}^{y-1} b_{j_l} + (n-x)\cdot(m-y)$$

为了让割达到最小割，显然要割a_i最小的x个，b_j最小的y个。所以将a, b基数排序，割就可以看作是关于x, y函数，目标是求这个函数的最小值。

$$cut(x,y) = \sum_{k=0}^{x-1} a_k + \sum_{l=0}^{y-1} b_l + (n-x)\cdot(m-y)$$

Rabbit Lunch

思考最小割一定是割一点a_i，割一点b_j，再割点中间的1。假设现在左边有x个a_i被割了，右边有y个b_j被割了，那么现在的割为：

$$cut = \sum_{k=0}^{x-1} a_{i_k} + \sum_{l=0}^{y-1} b_{j_l} + (n-x)\cdot(m-y)$$

为了让割达到最小割，显然要割a_i最小的x个，b_j最小的y个，所以将a, b基数排序，割就可以看作是关于x, y函数，目标是求这个函数的最小值。

$$cut(x,y) = \sum_{k=0}^{x-1} a_k + \sum_{l=0}^{y-1} b_l + (n-x)\cdot(m-y)$$

Rabbit Lunch

如果固定一个x，$cut(x, y+1) > cut(x, y)$等价于$b_{y+1} > n - x$，所以第一个满足$b_{y+1} > n - x$的y就是固定这个x时的最小值，可以依靠二分得到$O(n \log n)$的复杂度。

从小到大枚举x，因为$b_{y+1} > n - x$不等式右边严格下降，所以y的极值点非增，利用双指针即可得到$O(n)$的复杂度。

Rabbit Lunch

如果固定一个x，$cut(x, y+1) > cut(x, y)$等价于$b_{y+1} > n - x$，所以第一个满足$b_{y+1} > n - x$的y就是固定这个x时的最小值，可以依靠二分得到$O(n \log n)$的复杂度。

从小到大枚举x，因为$b_{y+1} > n - x$不等式右边严格下降，所以y的极值点非增，利用双指针即可得到$O(n)$的复杂度。

Hash It![6]

Description

有一个空串S，两种操作。

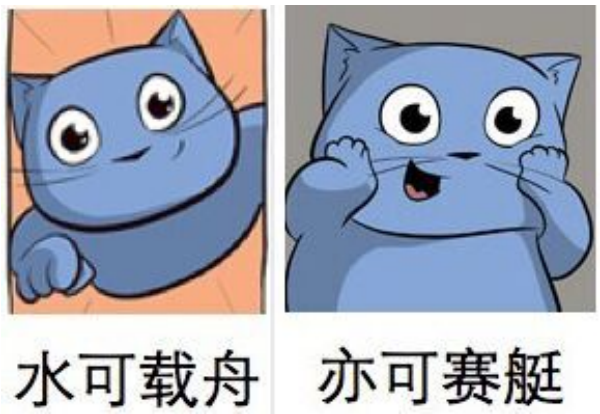
- 在S后面加一个字符
- 删去S最后一个字符

问每次操作结束后S的子串种数，操作数$\leq 10^5$。

[6] XIV Open Cup named after E.V. Pankratiev. GP of Ukraine, Problem H

Hash It!

看起来是可持久化后缀自动机？



Hash It!

回想一下怎么求子串数量？

将后缀排序后，设$sa[i]$为排名为i的后缀起始位置。

$$count = \sum_{i=0}^{n-1} \{n - sa[i] - LCP(sa[i-1], sa[i])\}$$

Hash It!

不妨把插入删除改到S的前面，那么每次插入删除影响的只有一个后缀。

所以问题变成了维护一个集合，可以插入一个后缀，删除一个后缀，顺带维护*count*。利用std::set即可。

怎么比较两个后缀的字典序？Hash it！这样就得到了$O(n \log^2 n)$的做法。

Hash It!

不妨把插入删除改到S的前面，那么每次插入删除影响的只有一个后缀。

所以问题变成了维护一个集合，可以插入一个后缀，删除一个后缀，顺带维护*count*。利用std::set即可。

怎么比较两个后缀的字典序？Hash it！这样就得到了$O(n \log^2 n)$的做法。

Hash It!

不妨把插入删除改到S的前面，那么每次插入删除影响的只有一个后缀。

所以问题变成了维护一个集合，可以插入一个后缀，删除一个后缀，顺带维护*count*。利用std::set即可。

怎么比较两个后缀的字典序？Hash it！这样就得到了$O(n \log^2 n)$的做法。

Hash It!

能不能不Hash啊？

插入和删除构成了一棵Trie，可以直接对Trie做类似后缀数组的倍增方法就可以求出所有节点到根的字符串的rank，后面在set里面直接比较rank即可。

这样就能得到$O(n \log n)$的做法了。

Hash It!

能不能不Hash啊？

插入和删除构成了一棵Trie，可以直接对Trie做类似后缀数组的倍增方法就可以求出所有节点到根的字符串的rank，后面在set里面直接比较rank即可。

这样就能得到$O(n \log n)$的做法了。

Hash It!

能不能不Hash啊？

插入和删除构成了一棵Trie，可以直接对Trie做类似后缀数组的倍增方法就可以求出所有节点到根的字符串的rank，后面在set里面直接比较rank即可。

这样就能得到$O(n \log n)$的做法了。

Board Game[7]

Description

平面上有n个点(x_i, y_i)，每个点还有一个颜色t_i。求：

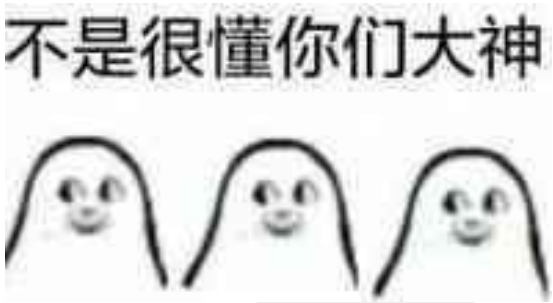
$$\max_{i,j}\{(x_i - x_j)^2 + (y_i - y_j)^2, t_i \neq t_j\}$$

$n \leq 250000, |x_i|, |y_i| \leq 10^9, t_i \leq n$。

[7] AMPPZ 2015, Problem G

Board Game

如果没有颜色不同的要求怎么做呢？



Board Game

没有颜色要求的话，最远点对一定在点集的凸包上，所以可以做出凸包，然后在凸包上卡(qia)一卡(qia)。

但现在这个做法就没有用了，还有什么能处理最远点查询呢？比如说kd-tree？

很遗憾kd-tree也没有办法在时限内给出解。

Board Game

没有颜色要求的话，最远点对一定在点集的凸包上，所以可以做出凸包，然后在凸包上卡(qia)一卡(qia)。

但现在这个做法就没有用了，还有什么能处理最远点查询呢？比如说kd-tree？

很遗憾kd-tree也没有办法在时限内给出解。

Board Game

没有颜色要求的话，最远点对一定在点集的凸包上，所以可以做出凸包，然后在凸包上卡(qia)一卡(qia)。

但现在这个做法就没有用了，还有什么能处理最远点查询呢？比如说kd-tree？

很遗憾kd-tree也没有办法在时限内给出解。

Board Game

如果只有两种颜色，就相当于要计算两个点集之间的最大距离，这要怎么计算呢？

当然这时候最远点对依然在两个点集的凸包上，这时候要计算凸包之间的最大距离。

这个问题可以卡壳吗？

Board Game

如果只有两种颜色，就相当于要计算两个点集之间的最大距离，这要怎么计算呢？

当然这时候最远点对依然在两个点集的凸包上，这时候要计算凸包之间的最大距离。

这个问题可以卡壳吗？

Board Game

如果只有两种颜色，就相当于要计算两个点集之间的最大距离，这要怎么计算呢？

当然这时候最远点对依然在两个点集的凸包上，这时候要计算凸包之间的最大距离。

这个问题可以卡壳吗？

Board Game

下定义

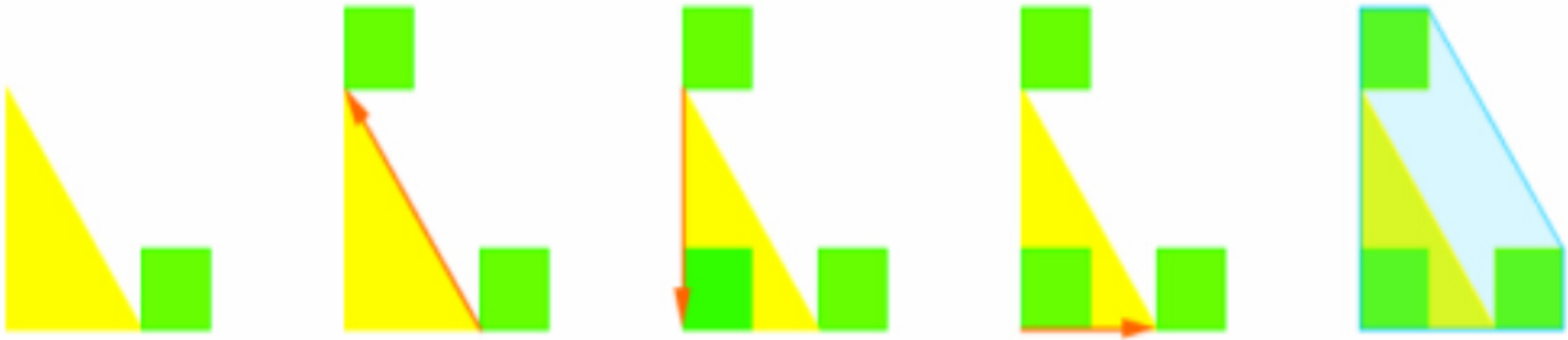
定义两个点集A, B的闵可夫斯基和为$\{a + b | a \in A, b \in B\}$。

摆事实

两个点集的闵可夫斯基和的凸集等于两个点集的凸集的闵可夫斯基和。

讲道理

两个凸集的闵可夫斯基和可以以$O(n + m)$的时间复杂度计算出来。



Board Game

下定义

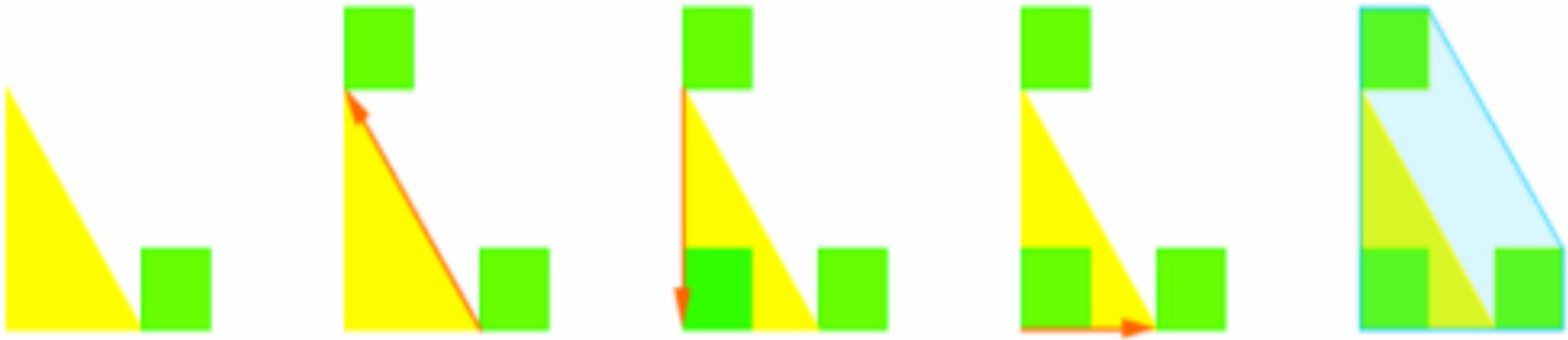
定义两个点集A, B的闵可夫斯基和为$\{a + b | a \in A, b \in B\}$。

摆事实

两个点集的闵可夫斯基和的凸集等于两个点集的凸集的闵可夫斯基和。

讲道理

两个凸集的闵可夫斯基和可以以$O(n + m)$的时间复杂度计算出来。



Board Game

下定义

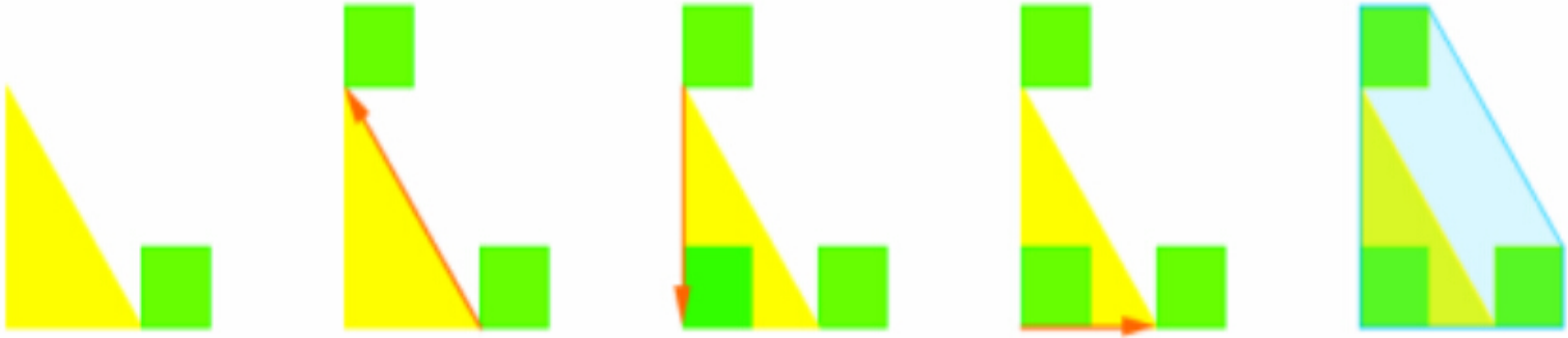
定义两个点集A, B的闵可夫斯基和为$\{a + b | a \in A, b \in B\}$。

摆事实

两个点集的闵可夫斯基和的凸集等于两个点集的凸集的闵可夫斯基和。

讲道理

两个凸集的闵可夫斯基和可以以$O(n + m)$的时间复杂度计算出来。



Board Game

所以只需把B集合中的每个点的横纵坐标取反，得到$-B$，再计算A的凸集与$-B$的凸集的闵可夫斯基和C，拿C中的每个点到原点的距离更新答案。

多种颜色怎么办呢？

分治，每次把颜色平分成两个集合，每个集合内部有着对应颜色的点。用这两个点集做上述的过程，再递归即可。时间复杂度$O(n \log n)$。

Board Game

所以只需把B集合中的每个点的横纵坐标取反，得到$-B$，再计算A的凸集与$-B$的凸集的闵可夫斯基和C，拿C中的每个点到原点的距离更新答案。

多种颜色怎么办呢？

分治，每次把颜色平分成两个集合，每个集合内部有着对应颜色的点。用这两个点集做上述的过程，再递归即可。时间复杂度$O(n \log n)$。

Board Game

所以只需把B集合中的每个点的横纵坐标取反，得到$-B$，再计算A的凸集与$-B$的凸集的闵可夫斯基和C，拿C中的每个点到原点的距离更新答案。

多种颜色怎么办呢？

分治，每次把颜色平分成两个集合，每个集合内部有着对应颜色的点。用这两个点集做上述的过程，再递归即可。时间复杂度$O(n \log n)$。

Sum of divisors[8]

Description

一个数是$K-expressible$的当且仅当这个数可以表示为K个它的约数的和，约数可以相同。有q组询问，每组给出L, R, K，询问$[L, R]$之间有多少个$K-expressible$的数字。

$q \leq 50000$，$1 \leq a \leq b \leq 10^{18}, 2 \leq K \leq 7$。

[8] AIM Fund Cup 2015, Problem F

Sum of divisors

什么数是满足条件的？

- $K = 2$时，所有偶数。
- $K = 4$时，$6 = 2 + 2 + 1 + 1...$
- $K = 6$时，$14 = 7 + 2 + 2 + 1 + 1 + 1...$
- $K = 7$时，$138 = 69 + 46 + 6 + 6 + 6 + 3 + 2...$

一个明显的性质是若某个数是合法的，那么它的所有倍数也是合法的。

Sum of divisors

什么数是满足条件的？

- $K = 2$时，所有偶数。
- $K = 4$时，$6 = 2 + 2 + 1 + 1...$
- $K = 6$时，$14 = 7 + 2 + 2 + 1 + 1 + 1...$
- $K = 7$时，$138 = 69 + 46 + 6 + 6 + 6 + 3 + 2...$

一个明显的性质是若某个数是合法的，那么它的所有倍数也是合法的。

Sum of divisors

对于一个合法的数n，n可以表示成：

$$n = \sum_{i=1}^{K} \frac{n}{a_i}, a_i | n$$

两边n可以同时去掉，等价于：

$$1 = \sum_{i=1}^{K} \frac{1}{a_i}, a_i | n$$

也就是每找到一组a使得它们倒数之和等于1，那么n可以取任何a_i的公倍数，也就是$LCM(a_1, a_2, ..., a_K)$的倍数。

Sum of divisors

对于一个合法的数n，n可以表示成：

$$n = \sum_{i=1}^{K} \frac{n}{a_i}, a_i | n$$

两边n可以同时去掉，等价于：

$$1 = \sum_{i=1}^{K} \frac{1}{a_i}, a_i | n$$

也就是每找到一组a使得它们倒数之和等于1，那么n可以取任何a_i的公倍数，也就是$LCM(a_1, a_2, ..., a_K)$的倍数。

Sum of divisors

因为$K \leq 7$，所以在一定范围内暴搜出所有的a，且它们的LCM要不大于10^{18}，而实际搜出来它们的LCM只有200左右。
对于每组询问实际上求的是有多少个数是这些LCM中至少一个的倍数，这个问题就可以用经典的容斥原理来解决。

可以先暴力处理掉LCM之间的整除关系，就可以发现$K = 7$的情况就只剩下15个数了。对于每个K可以先暴力处理出所有容斥的系数，对相同系数进行合并。这样单次询问极限情况只需要做大约400次运算。

Sum of divisors

因为$K \leq 7$，所以在一定范围内暴搜出所有的a，且它们的LCM要不大于10^{18}，而实际搜出来它们的LCM只有200左右。
对于每组询问实际上求的是有多少个数是这些LCM中至少一个的倍数，这个问题就可以用经典的容斥原理来解决。

可以先暴力处理掉LCM之间的整除关系，就可以发现$K = 7$的情况就只剩下15个数了。对于每个K可以先暴力处理出所有容斥的系数，对相同系数进行合并。这样单次询问极限情况只需要做大约400次运算。

- 感谢ICPCCamp给予我这次机会
- 祝大家后面几天Camping愉快!