stoorz 的博客

所有类型

未分类

题解

input search text

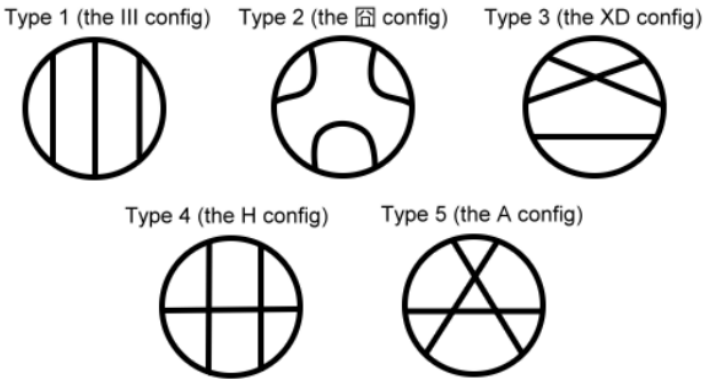
Theme by @cleverdango
Site Map GitHub 洛谷

题解 CF297E 【Mystic Carvings】

←文章列表

2020-02-13 10:55:01

不难发现，在一个圆中任意三条弦的位置关系只有如下五种情况：



~~图2的不是弦但是意思是一样的。~~

那么如果要满足任意一条弦所连接的两个点之间的点数相同，只有图2和图5满足条件，我们就是要计算这两种情况的方案数。

直接计算不是很好计算，我们考虑用总方案数减去不合法（即图1、3、4）的方案数。

总方案数很简单，就是从 n条弦中选出 3 条的方案数 C_n^3。

对于图1，我们可以枚举每一条弦，把它当做中间的那一条，那么我们只需要求出这条弦左、右分别有多少条与它不相交的弦，记作 $l[i]$ 和 $r[i]$，那么这一条弦在中间的方案数就是 $l[i] \times r[i]$。所以图1的总方案数就是 $\sum_{i=1}^{n} l[i] \times r[i]$。

对于图3和图4，我们把它们放在一起计算。容易发现这两张图的共同点是“三条弦中有两条满足另外的两条弦一条与其相交，一条与其相离”。仔细看一下这句话，应该能看懂的 :) 。

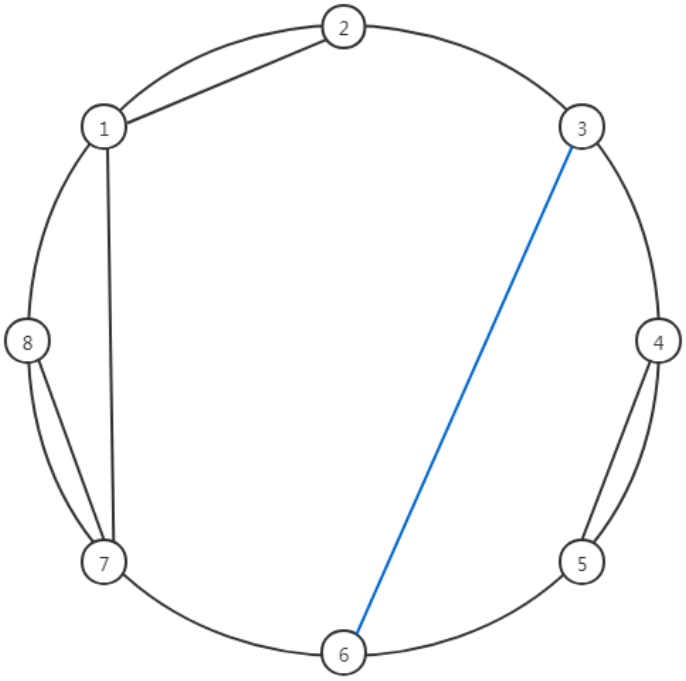
所以我们依然枚举每一条弦，对于任意一条弦，它的方案数就是“与它相交的弦的个数” $\times$ “与它不相交的弦的个数”。而我们知道与它不相交的弦的个数是 $l[i]+r[i]$，那么与它相交的弦的个数就是 $n-l[i]-r[i]-1$了。

考虑到每张图有两条弦满足上述“共同点”，所以这样每张图都会被算两遍，最终除以 2 即为图4和图5的答案。也就是

$\frac{\sum_{i=1}^{n}(l[i]+r[i])(n-l[i]-r[i]-1)}{2}$。

计算答案部分我们已经在 $O(n)$ 的时间复杂度内完成了，现在如果我们可以计算出 $l[i], r[i]$，这道题就解决了。

例如下图，我们需要求蓝色弦的左右分别与它不相交的弦的条数，黑色弦都不与它相交，其中仅有弦 $4-5$ 在它右边。



观察在蓝色弦左边的弦，设蓝色弦的两个端点为 x, y，其他弦的端点为 x', y'，（$x < y, x' < y'$）发现需要满足一下三个条件之一：

1. $x' < x$ 且 $y' < x$
2. $x' < x$ 且 $y' > y$
3. $x' > y$ 且 $y' > y$

可以发现，这与二维偏序十分像，用二维偏序即可解决。

同理，右边的弦只需要满足一个条件：$x' > x$ 且 $y' < y$，同样可以用二维偏序解决。

至此，这道题的两个部分都已经结束。时间复杂度 $O(n \log n)$。

```
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long ll;

const int N=300010;
int n,l[N],r[N];
ll ans1,ans2;
```

```
struct node
{
    int x,y,id;
}a[N];

struct BIT
{
    int c[N];

    void add(int x,int val)
    {
        for (;x<=n*2;x+=x&-x)
            c[x]+=val;
    }

    int ask(int x)
    {
        int sum=0;
        for (;x;x-=x&-x)
            sum+=c[x];
        return sum;
    }
}bit;

bool cmp1(node x,node y)
{
    return x.x<y.x;
}

bool cmp2(node x,node y)
{
    return x.y>y.y;
}

int main()
{
    scanf("%d",&n);
    for (int i=1;i<=n;i++)
    {
        scanf("%d%d",&a[i].x,&a[i].y);
        if (a[i].x>a[i].y) swap(a[i].x,a[i].y);
        a[i].id=i;
    }
    sort(a+1,a+1+n,cmp1);
    for (int i=1;i<=n;i++)  //二维偏序求每一条弦的
    {
        l[a[i].id]+=bit.ask(a[i].x)+bit.ask(n*2)
        bit.add(a[i].y,1);
```

```
    }
    memset(bit.c,0,sizeof(bit.c));
    for (int i=n;i>=1;i--)
    {
        r[a[i].id]+=bit.ask(a[i].y)-bit.ask(a[i]
        bit.add(a[i].y,1);
    }
    memset(bit.c,0,sizeof(bit.c));
    sort(a+1,a+1+n,cmp2);
    for (int i=1;i<=n;i++)
    {
        l[a[i].id]+=bit.ask(n*2)-bit.ask(a[i].y)
        bit.add(a[i].x,1);
    }
    ans1=1LL*n*(n-1)*(n-2)/6;  //C(n,3)
    for (int i=1;i<=n;i++)
    {
        ans1-=1LL*l[i]*r[i];
        ans2+=1LL*(l[i]+r[i])*(n-l[i]-r[i]-1);
    }
    printf("%I64d",ans1-ans2/2);
    return 0;
}
```

2

评论：

发布

skiy_gyx 2020-03-02 11:42:05

二维偏序妙啊