2013多校

FFT快速求卷积

学会了FFT。这题就很容易了。

其实题目是给了n条线段。问随机取三个，可以组成三角形的概率。

其实就是要求n条线段，选3条组成三角形的选法有多少种。

首先题目给了a数组，

如样例一：

4

1 3 3 4

把这个数组转化成num数组，num[i]表示长度为i的有num[i]条。

样例一就是

num = {0 1 0 2 1}

代表长度0的有0根，长度为1的有1根，长度为2的有0根，长度为3的有两根，长度为4的有1根。

使用FFT解决的问题就是num数组和num数组卷积。

num数组和num数组卷积的解决，其实就是从{1 3 3 4}取一个数，从{1 3 3 4}再取一个数，他们的和每个值各有多少个

例如{0 1 0 2 1}\*{0 1 0 2 1} 卷积的结果应该是{0 0 1 0 4 2 4 4 1 }

长度为n的数组和长度为m的数组卷积，结果是长度为n+m-1的数组。

{0 1 0 2 1}\*{0 1 0 2 1} 卷积的结果应该是{0 0 1 0 4 2 4 4 1 }。

这个结果的意义如下：

从{1 3 3 4}取一个数，从{1 3 3 4}再取一个数

取两个数和为 2 的取法是一种：1+1

和为 4 的取法有四种：1+3， 1+3 ，3+1 ，3+1

和为 5 的取法有两种：1+4 ，4+1；

和为 6的取法有四种：3+3,3+3,3+3,3+3,3+3

和为 7 的取法有四种： 3+4,3+4,4+3,4+3

和为 8 的取法有 一种：4+4

利用FFT可以快速求取循环卷积，具体求解过程不解释了，就是DFT和FFT的基本理论了。

总之FFT就是快速求到了num和num卷积的结果。只要长度满足>=n+m+1.那么就可以用循环卷积得到线性卷积了。

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <algorithm>

#include <math.h>

using namespace std;

const double PI = acos(-1.0);

struct complex

{

double r,i;

complex(double \_r = 0,double \_i = 0)

{

r = \_r; i = \_i;

}

complex operator +(const complex &b)

{

return complex(r+b.r,i+b.i);

}

complex operator -(const complex &b)

{

return complex(r-b.r,i-b.i);

}

complex operator \*(const complex &b)

{

return complex(r\*b.r-i\*b.i,r\*b.i+i\*b.r);

}

};

void change(complex y[],int len)

{

int i,j,k;

for(i = 1, j = len/2;i < len-1;i++)

{

if(i < j)swap(y[i],y[j]);

k = len/2;

while( j >= k)

{

j -= k;

k /= 2;

}

if(j < k)j += k;

}

}

void fft(complex y[],int len,int on)

{

change(y,len);

for(int h = 2;h <= len;h <<= 1)

{

complex wn(cos(-on\*2\*PI/h),sin(-on\*2\*PI/h));

for(int j = 0;j < len;j += h)

{

complex w(1,0);

for(int k = j;k < j+h/2;k++)

{

complex u = y[k];

complex t = w\*y[k+h/2];

y[k] = u+t;

y[k+h/2] = u-t;

w = w\*wn;

}

}

}

if(on == -1)

for(int i = 0;i < len;i++)

y[i].r /= len;

}

const int MAXN = 400040;

complex x1[MAXN];

int a[MAXN/4];

long long num[MAXN];//100000\*100000会超int

long long sum[MAXN];

int main()

{

int T;

int n;

scanf("%d",&T);

while(T--)

{

scanf("%d",&n);

memset(num,0,sizeof(num));

for(int i = 0;i < n;i++)

{

scanf("%d",&a[i]);

num[a[i]]++;

}

sort(a,a+n);

int len1 = a[n-1]+1;

int len = 1;

while( len < 2\*len1 )

len <<= 1;

for(int i = 0;i < len1;i++)

x1[i] = complex(num[i],0);

for(int i = len1;i < len;i++)

x1[i] = complex(0,0);

//x1数组自己对自己卷积

fft(x1,len,1);

for(int i = 0;i < len;i++)

x1[i] = x1[i]\*x1[i];

fft(x1,len,-1);

for(int i = 0;i < len;i++)

num[i] = (long long)(x1[i].r+0.5);

len = 2\*a[n-1];

//减掉取两个相同的组合

for(int i = 0;i < n;i++)

num[a[i]+a[i]]--;

//选择的无序，除以2

for(int i = 1;i <= len;i++)

{

num[i]/=2;

}

sum[0] = 0;

for(int i = 1;i <= len;i++)

sum[i] = sum[i-1]+num[i];

long long cnt = 0;

for(int i = 0;i < n;i++)

{

cnt += sum[len]-sum[a[i]];

//减掉一个取大，一个取小的

cnt -= (long long)(n-1-i)\*i;

//减掉一个取本身，另外一个取其它

cnt -= (n-1);

//减掉大于它的取两个的组合

cnt -= (long long)(n-1-i)\*(n-i-2)/2;

}

//总数

long long tot = (long long)n\*(n-1)\*(n-2)/6;

printf("%.7f\n",(double)cnt/tot);

}

return 0;

}