

快速傅里叶变换(FFT)

原理

[OI Wiki-快速傅里叶变换](#)

例题

例 1

题意

[P3803 【模板】多项式乘法 FFT](#)

给定一个 n 次多项式 $F(x)$ 和一个 m 次多项式 $G(x)$ 求出 $F(x)$ 和 $G(x)$ 的卷积。

题解

通过 DFT 和 IDFT 两个过程，实现多项式由系数表示法 $\rightarrow$ 点值表示法 $\rightarrow$ 系数表示法。利用单位根的性质实现奇偶分治过程，时间复杂度 $O(n \log n)$ 其中递归的分治过程时空消耗较大，通过蝴蝶变换找到各项系数（点值）分治后的位置，直接往上迭代，实现优化。

评价

FFT 模板题

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
typedef complex<db> Comp;
const db PI=acos(-1.0);
const int N=1e7+5;
int rev[N]; // i 二进制翻转后的数
Comp a[N], b[N]; // 系数 (点值)
void FFT(Comp *y, int len, int on){
    for(int i=0; i<len; i++){
        if(i<rev[i])
            swap(y[i], y[rev[i]]); // 变到分治后的位置，准备往上迭代
        for(int mid=1; mid<len; mid<=<1){ // mid 是区间长度的一半
            Comp wn(cos(PI/mid), sin(on*PI/mid)); // 单位根
```

```

        for(int j=0,R=mid<<1;j<len;j+=R){// j 是区间左端点 R 是区间长度
            Comp w(1,0);
            for(int k=0;k<mid;k++,w=w*wn){// k 是区间内的位置, 枚举左半区间
                Comp u=y[j+k];
                Comp t=w*y[j+k+mid];
                y[j+k]=u+t;
                y[j+k+mid]=u-t;
            }
        }
    }
}

int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=0;i<=n;i++){
        db x;
        scanf("%lf",&x);
        a[i]={x,0};
    }
    for(int i=0;i<=m;i++){
        db x;
        scanf("%lf",&x);
        b[i]={x,0};
    }
    int len=1,l=0;
    while(len<=n+m) len<<=1,l++;// 区间总长度必须是2的幂次
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));// 蝴蝶变换
    FFT(a,len,1);// 系数->点值
    FFT(b,len,1);// 系数->点值
    for(int i=0;i<=len;i++) a[i]=a[i]*b[i];// 点值相乘
    FFT(a,len,-1);// 点值->系数
    for(int i=0;i<=n+m;i++)
        printf("%d ",(int)(real(a[i])/len+0.5));// 四舍五入输出整数值
    return 0;
}

```

例 2

题意

P1919 模板 A*B Problem升级版(FFT快速傅里叶)

给定正整数 a, b 求 $a \times b$ $1 \leq a, b \leq 10^6$

题解

任一 n 位十进制正整数 k 可以用多项式形式表示 $k = \overline{a_{n-1}a_{n-2}\cdots a_2a_1a_0} = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$ ($x=10$) 这样一来, 两数相乘即可转化为多

项式相乘的问题，套用 FFT 模板即可，时间复杂度 $O(n \log n)$ （ n 为位数）。

评价

FFT 模板题

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef double db;
typedef complex<db> Comp;
const db PI=acos(-1.0);
const int N=1e7+5;
int rev[N];
Comp a[N],b[N];
int c[N];
char s1[N],s2[N];
void FFT(Comp *y,int len,int on){// FFT 模板
    for(int i=0;i<len;i++)
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    for(int mid=1;mid<len;mid<=1){
        Comp wn(cos(PI/mid),sin(on*PI/mid));
        for(int j=0,R=mid<<1;j<len;j+=R){
            Comp w(1,0);
            for(int k=0;k<mid;k++,w=w*wn){
                Comp u=y[j+k];
                Comp t=w*y[j+k+mid];
                y[j+k]=u+t;
                y[j+k+mid]=u-t;
            }
        }
    }
}
int main(){
    scanf("%s%s",s1,s2);
    int n=strlen(s1),m=strlen(s2);
    n--,m--;
    for(int i=0;i<=n;i++) a[i]={(double)(s1[n-i]-'0'),0};
    for(int i=0;i<=m;i++) b[i]={(double)(s2[m-i]-'0'),0};
    int len=1,l=0;
    while(len<=n+m) len<=1,l++;
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));
    FFT(a,len,1);
    FFT(b,len,1);
    for(int i=0;i<=len;i++) a[i]=a[i]*b[i];
    FFT(a,len,-1);
```

```
for(int i=0;i<=n+m;i++){// 需要进位
    int x=(int)(real(a[i])/len+0.5);
    c[i]+=x;
    c[i+1]+=c[i]/10;
    c[i]=c[i]%10;
}
if(c[n+m+1]!=0) n++;
for(int i=n+m;i>=0;i--)
    printf("%d",c[i]);
return 0;
}
```

例 3

题意

P3338 [ZJOI2014]力

给出 n 个数 $q_1, q_2, \dots, q_n$ 定义 $F_j = \sum_{i=1}^{j-1} \frac{q_i \times q_j}{(i-j)^2} - \sum_{i=j+1}^n \frac{q_i \times q_j}{(i-j)^2}$ $E_i = \frac{F_i}{q_i}$ 对 $1 \leq i \leq n$ 求 E_i 的值。

题解

对题目式子进行化简，可得 $E_j = \sum_{i=1}^{j-1} \frac{q_i}{(i-j)^2} - \sum_{i=j+1}^n \frac{q_i}{(i-j)^2}$ 令 $a_i = q_i, b_i = \frac{1}{i^2}, a_0 = b_0 = 0$ $E_j = \sum_{i=0}^{j-1} a[i]b[j-i] - \sum_{i=j}^n a[i]b[i-j]$ 注意到左边已化成卷积形式，现对右边进行转化 $\sum_{i=j}^n a[i]b[i-j] = \sum_{i=0}^{n-j} a[i+j]b[i]$ 令 $c[i] = a[n-i]$ (翻转变换)，得 $\sum_{i=0}^{n-j} c[n-i-j]b[i]$ 令 $t = n-j$ 得 $\sum_{i=0}^t c[t-i]b[i]$ 至此，右边也化为了卷积形式。

令 $f(x) = \sum_{i=0}^n a_{ix}^i, g(x) = \sum_{i=0}^n b_{ix}^i, h(x) = \sum_{i=0}^n c_{ix}^i$ 答案即为 $E_j = [x^j](f(x)g(x)) - [x^{n-j}](g(x)h(x))$

用 FFT 做多项式乘法即可，时间复杂度 $O(n \log n)$

评价

主要考察推式子能力，将式子转化为卷积形式，再套用 FFT 加速多项式乘法。

代码

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e5+5;
int rev[N<<2];
typedef complex<double> Comp;
```

```

const double PI=acos(-1.0);
Comp a[N<<2],b[N<<2],c[N<<2];
void FFT(Comp *y,int len,int on){
    for(int i=0;i<len;i++){
        if(i<rev[i])
            swap(y[i],y[rev[i]]);
    }
    for(int mid=1;mid<len;mid<=<1){
        Comp wn(cos(PI/mid),sin(on*PI/mid));
        for(int j=0,R=mid<<1;j<len;j+=R){
            Comp w(1,0);
            for(int k=0;k<mid;k++,w=w*wn){
                Comp u=y[j+k];
                Comp t=w*y[j+k+mid];
                y[j+k]=u+t;
                y[j+k+mid]=u-t;
            }
        }
    }
    if(on==<1)
        for(int i=0;i<len;i++)
            y[i]={real(y[i])/len,imag(y[i])};
}

int main(){
    int n;
    scanf("%d",&n);
    a[0]=b[0]=c[n]={0,0};
    for(int i=1;i<=n;i++){
        double q;
        scanf("%lf",&q);
        c[n-i]=a[i]={q,0};
        b[i]={(double)1.0/i/i,0};
    }
    int len=1,l=0;
    while(len<=n+n) len<=<1,l++;
    for(int i=0;i<len;i++) rev[i]=(rev[i>>1]>>1)|((i&1)<<(l-1));
    FFT(a,len,1);
    FFT(b,len,1);
    FFT(c,len,1);
    for(int i=0;i<=len;i++){
        a[i]=a[i]*b[i];
        c[i]=c[i]*b[i];
    }
    FFT(a,len,<1);
    FFT(c,len,<1);
    for(int j=1;j<=n;j++){
        printf("%.3f\n",real(a[j])-real(c[n-j]));
    }
    return 0;
}

```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:2021%E3%89%B4%E5%BA%A6%E8%AE%AD%E7%BB%83%E8%AE%A1%E5%88%92%E5%8F%8A%E8%AE%AD%E7%BB%83%E8%AE%B0%E5%BD%95:lgwza:%E5%8F%A8%E9%80%9F%E5%82%85%E9%87%B8%E5%8F%B6%E5%8F%98%E6%8D%A2_ft

Last update: 2021/02/15 11:27