FWT快速沃尔什变换

算法思想

或卷积

大概长这个样子 $C_{k} = \sum_{i|j=k} A_{i}×B_{j}$

满足交换律、结合律 $(A+B)|C=A|C+B|C$ 均为显然

几个重要的结论：

$FWT(A+B) = FWT(A) + FWT(B)$

$FWT(A) = (FWT(A_{0}),FWT(A_{0})+FWT(A_{1}))$ A_{0} 表示最高位为 0 的部分，也就是前 2^{n-1} 项； A_{1} 表示最高位为 1 的部分，也就是后 2^{n-1} 项。

对于 or 卷积来说 $FWT(A)[I] = \sum_{j|i=i}A[j]$

此时用数学归纳法有：

$FWT(A|B)$

$=FWT((A|B)_{0},(A|B)_{1})$

$=FWT(A_{0}|B_{0},A_{0}|B_{1}+A_{1}|B_{0}+A_{1}|B_{1})$

$=(FWT(A_{0}|B_{0}),FWT(A_{0}|B_{0}+A_{0}|B_{1}+A_{1}|B_{0}+A_{1}|B_{1}))$

$=(FWT(A_{0})×FWT(B_{0}),FWT(A_{0})×FWT(B_{0})+FWT(A_{0})×FWT(B_{1})+FWT(A_{1})×FWT(B_{0})+FWT(A_{1})×FWT(B_{1}))$

$=(FWT(A_{0})×FWT(B_{0}),(FWT(A_{0})+FWT(A_{1}))×(FWT(B_{0})+FWT(B_{1})))$

$=(FWT(A_{0},FWT(A_{0}+A_{1})))×(FWT(B_{0},FWT(B_{0}+B_{1})))$

$=FWT(A)×FWT(B)$

得证

所以类似于 FFT 也有一个叫做 $IFWT$ 的东西。

$IFWT(A)=(IFWT(A_{0}),IFWT(A_{1})-IFWT(A_{0}))$

和卷积

和或卷积类似

结论：

$FWT(A)=(FWT(A_{0}+A_{1}),FWT(A_{1}))$

$FWT(A+B)=FWT(A)+FWT(B)$

同样的数学归纳法可以证明和卷积成立

$IFWT(A)=(IFWT(A_{0})-IFWT(A_{1}),IFWT(A_{1}))$

异或卷积

结论：

$FWT(A)=(FWT(A_{0})+FWT(A_{1}),FWT(A_{0})-FWT(A_{1}))$

$FWT(A+B)=FWT(A)+FWT(B)$

$IFWT(A)=({\frac {IFWT(A_{0})-IFWT(A_{1})} 2},{\frac {IFWT(A_{0})-IFWT(A_{1})} 2})$

算法实现

实现的时候分治一下就好啦

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=1500000;

int MOD=998244353;
int inv2=(MOD+1)>>1,N;
ll a1[MAXN],b1[MAXN],a2[MAXN],b2[MAXN],a3[MAXN],b3[MAXN];

void or_FWT(ll *P,int opt) {
    for(int i=2; i<=N; i<<=1)
        for(int p=i>>1,j=0; j<N; j+=i)
            for(int k=j; k<j+p; ++k)
                P[k+p]+=P[k]*opt,P[k+p]=(P[k+p]+MOD)%MOD;
}

void and_FWT(ll *P,int opt) {
    for(int i=2; i<=N; i<<=1)
        for(int p=i>>1,j=0; j<N; j+=i)
            for(int k=j; k<j+p; ++k)
                P[k]+=P[k+p]*opt,P[k]=(P[k]+MOD)%MOD;
}

void xor_FWT(ll *P,int opt) {//如果不是在模意义下的话，把逆元变成直接除二就好了。
    for(int i=2; i<=N; i<<=1)
        for(int p=i>>1,j=0; j<N; j+=i)
```

```
            for(int k=j; k<j+p; ++k) {
                ll x=P[k],y=P[k+p];
                P[k]=(x+y)%MOD;
                P[k+p]=(x-y+MOD)%MOD;
if(opt==-1)P[k]=1ll*P[k]*inv2%MOD,P[k+p]=1ll*P[k+p]*inv2%MOD;
            }
}

int main() {
    scanf("%d",&N);
    N=1<<N;
    for(int i=0;i<N;i++) {
        scanf("%lld",&a1[i]);
        a2[i]=a3[i]=a1[i];
    }
    for(int i=0;i<N;i++) {
        scanf("%lld",&b1[i]);
        b2[i]=b3[i]=b1[i];
    }
    or_FWT(a1,1);or_FWT(b1,1);
    for(int i=0;i<N;i++) a1[i]=a1[i]*b1[i]%MOD;
    or_FWT(a1,-1);
    and_FWT(a2,1);and_FWT(b2,1);
    for(int i=0;i<N;i++) a2[i]=a2[i]*b2[i]%MOD;
    and_FWT(a2,-1);
    xor_FWT(a3,1);xor_FWT(b3,1);
    for(int i=0;i<N;i++) a3[i]=a3[i]*b3[i]%MOD;
    xor_FWT(a3,-1);
    for(int i=0;i<N;i++) {
        printf("%lld ",a1[i]);
    }
    putchar(10);
    for(int i=0;i<N;i++) {
        printf("%lld ",a2[i]);
    }
    putchar(10);
    for(int i=0;i<N;i++) {
        printf("%lld ",a3[i]);
    }
    putchar(10);
    return 0;
}
```