

# FWT快速沃尔什变换

## 算法思想

### 或卷积

大概长这个样子  $C_k = \sum_{i+j=k} A_i \times B_j$

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：

$$\text{FWT}(A+B) = \text{FWT}(A) + \text{FWT}(B)$$

$\text{FWT}(A) = (\text{FWT}(A_0), \text{FWT}(A_0) + \text{FWT}(A_1))$   $A_0$  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  $\text{FWT}(A)[i] = \sum_{j|i} A[j]$

此时用数学归纳法有：

$$\text{FWT}(A|B)$$

$$= \text{FWT}((A|B)_0, (A|B)_1)$$

$$= \text{FWT}(A_0|B_0, A_0|B_1 + A_1|B_0 + A_1|B_1)$$

$$= (\text{FWT}(A_0|B_0), \text{FWT}(A_0|B_0 + A_0|B_1 + A_1|B_0 + A_1|B_1))$$

$$= (\text{FWT}(A_0) \times \text{FWT}(B_0), \text{FWT}(A_0) \times \text{FWT}(B_0) + \text{FWT}(A_0) \times \text{FWT}(B_1) + \text{FWT}(A_1) \times \text{FWT}(B_0) + \text{FWT}(A_1) \times \text{FWT}(B_1))$$

$$= (\text{FWT}(A_0) \times \text{FWT}(B_0), (\text{FWT}(A_0) + \text{FWT}(A_1)) \times (\text{FWT}(B_0) + \text{FWT}(B_1)))$$

$$= (\text{FWT}(A_0), \text{FWT}(A_0 + A_1)) \times (\text{FWT}(B_0), \text{FWT}(B_0 + B_1))$$

$$= \text{FWT}(A) \times \text{FWT}(B)$$

得证

所以类似于 FFT 也有一个叫做 IFWT 的东西。

$$\text{IFWT}(A) = (\text{IFWT}(A_0), \text{IFWT}(A_1) - \text{IFWT}(A_0))$$

### 和卷积

和或卷积类似

结论：

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

$$\text{FWT}(A+B) = \text{FWT}(A) + \text{FWT}(B)$$

同样的数学归纳法可以证明和卷积成立

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

## 异或卷积

结论：

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

$$\text{FWT}(A+B) = \text{FWT}(A) + \text{FWT}(B)$$

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

## 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;
}

```

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

Permanent link:  
[https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E7%8E%8B%E6%99%BA%E5%BD%AA:fwf&rev=1626778308](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%8E%8B%E6%99%BA%E5%BD%AA:fwf&rev=1626778308)

Last update: 2021/07/20 18:51