Notes on FFT Problems Part 2

md.mottakin.chowdhury

August 2018

1 Online FFT - Does it really exist?

Let $G[1...N]$ be sequence of length N that is known to us. We wish to compute all N terms of sequence F defined as follows:

$$F[n] = \sum_{i=1}^{n-1} F[i] \times G[n-i]$$

.
So, value of $F[n]$ depends on previous $n-1$ values. How can we compute it efficiently?

It can be done by a trick known or not-known as *Online FFT*. The idea is to divide the $G[1...N]$ into blocks. First a block of size 1, then 1, then 2, then 4, then 8 etc. So we divide $G[1...N]$ in blocks of power of 2 where first two blocks are of size 1. So it looks like:

$$G[1] \mid G[2] \mid G[3]\ G[4] \mid G[5]\ G[6]\ G[7]\ G[8] \mid ...$$

Now, we do following for each $F[i]$:

- When we get an $F[i]$, first we convolve it with first two blocks. And add those to the appropriate entries.
- Now suppose i is non-zero multiple of 2^k for some k. Then we convolve $F[i-2^k]$ to $F[i-1]$ with the block in G of the same size for each k.

1.1 Problem 1

A problem from Codechef. Basically the problem reduces to this recursive relation:

$$f(n) = \sum_{i=0}^{n-1} A[n-1-i] \times f[i]$$

The curical part of the code:

```
void solve() {
    f[0]=1; // base case
    for(int i=0; i<=MAX; i++) {
        // Doing the part 1
        f[i+1]=(f[i+1]+f[i]*A[0])%mod;
        f[i+2]=(f[i+2]+f[i]*A[1])%mod;
        if(!i) continue;
        // part 2
        int limit=(i&-i);

        for(int p=2; p<=limit; p*=2)
        {
            convolve(i-p,i-1,p,min(2*p-1,MAX));
        }
```

```
    }
}

void convolve(int l1, int r1, int l2, int r2)
{
    int n=max(r1-l1+1,r2-l2+1);
    int t=1;
    while(t<n) t<<=1;
    n=t;

    vector<ll> a(n), b(n);
    for(int i=l1; i<=r1; i++) a[i-l1]=f[i];
    for(int i=l2; i<=r2; i++) b[i-l2]=A[i];

    vector<ll> ret=fft::multiply(a,b);

    for(int i=0; i<ret.size(); i++)
    {
        int idx=i+l1+l2+1;
        if(idx>MAX) break;
        // adding to the appropriate entry
        f[idx]+=ret[i];
        f[idx]%=mod;
    }
}
```

By the way, the problem needs NTT.

1.2 Problem 2

So you have n objects with m different colors. You take a group of k objects. How many different possible groups are there? Note that two groups are different only when number of occurrence of one or more colors are different. The order or object position does not matter. For instance, for input $1, 2, 3, 2$, there are 4 such groups: $(1, 2), (1, 3), (2, 3), (2, 2)$.

Solution: For each color l, we build a polynomial,

$$p_l(x) = \sum_{i=0}^{cnt[l]} x^i$$

Then we multiply all these polynomials. The answer is the coefficient of x^k. To do the multiplication, we can do divide and conquer.

```
vi go(int l, int r)
{
    vi curr;
    if(l==r)
    {
        FOR(j,0,cnt[l]+1)
        curr.pb(1);
        // cout<<"curr: "; VecPrnt(curr);
        return curr;
    }
```

```
    int mid=(l+r)/2;

    vi ls=go(l,mid);
    vi rs=go(mid+1,r);
    vi ret=mult_mod(ls,rs);

    return ret;
}
```