

#05-4. 다항식의 빠른 곱셈


나정휘

<https://justicehui.github.io/>

문제 소개

두 N차 다항식을 곱하는 문제

- 만약 한쪽의 차수가 작으면 계수가 0인 항을 추가해서 두 다항식의 차수를 맞출 수 있음
- 중학교에서 배운 곱셈 방식을 단순하게 구현하면 $O(N^2)$
- 이번 슬라이드의 목표는 문제를 $O(N^{\log_2 3})$ 시간에 해결하는 것



```
// f[i] : x^i의 계수
vector<int> Multiply(const vector<int> &a, const vector<int> &b){
    vector<int> res(a.size()+b.size()-1);
    for(int i=0; i<a.size(); i++){
        for(int j=0; j<b.size(); j++){
            res[i+j] += a[i] * b[j];
        }
    }
    return res;
}
```

분할 정복 알고리즘

$O(N^2)$ 알고리즘

- N 차 다항식은 $f(x) = f_0(x) \times x^{N/2} + f_1(x)$ 와 같이 $N/2$ 차 다항식 2개로 나타낼 수 있음
 - $f(x) \times g(x) = (f_0(x)x^{N/2} + f_1(x)) \times (g_0(x)x^{N/2} + g_1(x))$
 - $= f_0(x) \times g_0(x) \times x^N + \{f_0(x) \times g_1(x) + f_1(x) \times g_0(x)\} \times x^{N/2} + f_1(x) \times g_1(x)$
- 따라서 두 N 차 다항식의 곱셈은 $N/2$ 차 다항식의 곱셈 4번과 덧셈 2번으로 구할 수 있음
 - 덧셈과 x^N 을 곱하는 것은 $O(N)$ 이므로 $T(N) = 4T\left(\frac{N}{2}\right) + O(N) = O(N^2)$

분할 정복 알고리즘

$O(N^{\log_2 3})$ 알고리즘

- N 차 다항식은 $f(x) = f_0(x) \times x^{N/2} + f_1(x)$ 와 같이 $N/2$ 차 다항식 2개로 나타낼 수 있음
 - $f(x) \times g(x) = (f_0(x)x^{N/2} + f_1(x)) \times (g_0(x)x^{N/2} + g_1(x))$
 - $= f_0(x) \times g_0(x) \times x^N + \{f_0(x) \times g_1(x) + f_1(x) \times g_0(x)\} \times x^{N/2} + f_1(x) \times g_1(x)$
- 곱셈 횟수를 줄이는 아이디어
 - $\{f_0(x) + f_1(x)\} \times \{g_0(x) + g_1(x)\} = f_0(x) \times g_0(x) + \{f_0(x) \times g_1(x) + f_1(x) \times g_0(x)\} + f_1(x) \times g_1(x)$
 - 따라서 3번의 곱셈으로 빨간색, 하늘색, 보라색 부분을 구하면 2번의 곱셈으로 초록색 부분을 구할 수 있음
- $T(N) = 3T\left(\frac{N}{2}\right) + O(N) = O(N^{\log_2 3})$

분할 정복 알고리즘



```
vector<ll> Multiply(vector<ll> a, vector<ll> b){
    int n = a.size(), k = n / 2;
    vector<ll> res(n+n);
    if(n <= 32){
        for(int i=0; i<n; i++) for(int j=0; j<n; j++) res[i+j] += a[i] * b[j];
        return res;
    }

    // a = a0 * x^k + a1, b = b0 * x^k + b1
    vector<ll> a1(a.begin(), a.begin()+k), a0(a.begin()+k, a.end());
    vector<ll> b1(b.begin(), b.begin()+k), b0(b.begin()+k, b.end());

    // z0 = a0 * b0, z2 = a1 * b1
    auto z0 = Multiply(a0, b0), z2 = Multiply(a1, b1);
    // a0 <- a0 + a1, b0 <- b0 + b1
    for(int i=0; i<k; i++) a0[i] += a1[i], b0[i] += b1[i];
    // z1 = (a0 + a1) * (b0 + b1) - a0b0 - a1b1
    auto z1 = Multiply(a0, b0);
    for(int i=0; i<z0.size(); i++) z1[i] -= z0[i];
    for(int i=0; i<z2.size(); i++) z1[i] -= z2[i];

    // res = z0 * x^2k + z1 * x^k + z2
    for(int i=0; i<z0.size(); i++) res[i+k+k] += z0[i];
    for(int i=0; i<z1.size(); i++) res[i+k] += z1[i];
    for(int i=0; i<z2.size(); i++) res[i] += z2[i];
    return res;
}
```

질문?

더 공부할 거리

FFT를 이용한 다항식 곱셈

- $O(N \log N)$ 시간에 두 다항식의 곱을 구하는 방법
- 선수 지식: 오일러 공식($e^{i\theta} = \cos \theta + i \sin \theta, e^{i\pi} + 1 = 0$), 복소 평면

스트라센 알고리즘 (Strassen algorithm)

- $N \times N$ 크기의 행렬 곱셈을 $\frac{N}{2} \times \frac{N}{2}$ 크기의 행렬 곱셈 7번과 덧셈 18번으로 처리
- $T(N) = 7T\left(\frac{N}{2}\right) + O(N^2) = O(N^{\log_2 7})$ 시간에 두 정사각행렬의 곱을 구하는 알고리즘