CS300 Chapter #1

마틴 지글러 *

September 7th, 2021

1 Virtues of Computer Science

생략.

2 Calculating Fibonacci Numbers

$$F_0 = 0$$
$$F_1 = 1$$
$$F_n = F_{n-1} + F_{n-2}$$

2.1 Recursive Algorithm

```
FibRek(n)
    if n = 0 return 0; if n = 1 return 1;
    return FibRek(n-1) + FibRek(n-2)
```

이 같은 경우에는 덧셈의 횟수가 exponential 하게 증가한다.

2.2 Iterative Algorithm

계산한 피보나치 수를 저장해놓으면 위 방법보다 더 효율적이다.

* 동한뭉

```
FibIter(n)
    if n = 0 return 0;
    fib := 1; fibL := 0;
    while n > 1 do
        tmp := fibL;
        fibL := fib;
        fib := fibL + tmp;
        n := n - 1; end
    return fib;
```

addition 과 assignment 의 개수가 n 에 대해서 linear 하게 증가한다. $O(n)$.
피보나치 수는 exponential 하게 증가하므로, 이를 표현하기 위한 비트 수는 선형적으로 증가한다. 이런 요소까지 고려하였을 때, bit cost 는 n^2 에 비례하게 된다.

2.3 Using Vectors and Matrices (Most efficient)

이전까지의 알고리즘은 모두 덧셈만을 사용했었지만, 이 알고리즘은 정수의 곱셈도 사용한다.

$$\begin{bmatrix} F_n \\ F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} F_{n-1} \\ F_{n-2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^k \cdot \begin{bmatrix} F_{n-k} \\ F_{n-k-1} \end{bmatrix} \quad k := n-1$$

행렬의 거듭제곱을 연산하기 때문에 logarithmic 하다. 뭔가 자세한 내용이 있지만 교수님께서 넘어가신다.

2.4 Using Explicit Formula

$$F_n = \frac{\varphi^n - \left(-\frac{1}{\varphi}\right)^n}{\sqrt{5}} \quad \varphi := \frac{1+\sqrt{5}}{2}$$

이것도 n 거듭제곱을 포함한다.

2.5 Using Exponential and Logarithmic Function

$$\varphi^n = \exp(n \cdot \ln \varphi)$$

이렇게 거듭제곱을 계산하면 실제 계산은 상수번 이루어지긴 한다. 하지만 사용된 함수들이 너무 powerful 하다.

5가지 알고리즘을 보면서 얘기하고자 하는 것은 trade-off 이다. 많은 연산을 하지 않으려면 powerful 한 operation 이 필요하고, 너무 high level 의 연산을 사용하면 실제 컴퓨터 연산에서는 생각보다 많은 시간이 걸리거나 정확하지 않은 값이 나올 수도 있다.

3 Master Theorem

$f:\mathbb{N}\to\mathbb{R}$ 가 증가함수이고 1 이상의 어떤 정수 a, b 와 양수 d 에 대해 b 의 임의의 거듭제곱 n 이

$$f(n) = a \cdot f(n/b) + O(n^d)$$

를 만족하면

$$f(n) \quad \text{is} \quad \begin{cases} O\left(n^d\right) & \text{if} \quad a < b^d \\ O\left(n^d \log n\right) & \text{if} \quad a = b^d \\ O\left(n^{\log_b a}\right) & \text{if} \quad a > b^d \end{cases}$$

이다.

4 Polynomial Multiplication

Input : $N-1$ 차 다항식 $A(x)$ 와 $(< N)$ 차 다항색 $B(x)$ 의 계수.
Output : $C(x) := A(x) \cdot B(x)$ 의 계수. 이때 $C(x)$ 의 차수는 $2N-2$ 이하이다.
$T(N)$ 은 선형결합 같은 arithmetic operation 의 개수이다.

4.1 Long Multiplication

$A(x)$ 와 $B(x)$ 의 항들을 차수가 높은 절반과 낮은 절반으로 나누어서 아래와 같이 표현한다. (N 이 짝수라고 치자.)

$$A(x) = A_0(x) + A_1(x) \cdot x^{N/2}$$
$$B(x) = B_0(x) + B_1(x) \cdot x^{N/2}$$

그러면 $A_0(x), A_1(x), B_0(x), B_1(x)$ 는 각각 $N/2-1$ 차 다항식이다. 여기서 $A(x) \cdot B(x)$ 를 계산하면

$$
\begin{aligned}
A \cdot B = &(A_0 \cdot B_0) \\
&+ (A_1 \cdot B_0 + A_0 \cdot B_1) \cdot x^{N/2} \\
&+ (A_1 \cdot B_1) \cdot x^N
\end{aligned}
$$

$$
\begin{aligned}
C_0(x) &:= A_0 \cdot B_0 \\
C_1(x) &:= A_1 \cdot B_0 + A_0 \cdot B_1 \\
C_2(x) &:= A_1 \cdot B_1
\end{aligned}
$$

이제는 처음 문제에서 절반 크기가 된 4개의 문제

$$(A_0 \cdot B_0), (A_1 \cdot B_0), (A_0 \cdot B_1), (A_1 \cdot B_1)$$

를 풀고($4 \cdot T(N/2)$), 4개의 답을 더하기만($O(N)$) 하면 된다.

$$\therefore T(N) = 4 \cdot T(N/2) + O(N)$$

4.2 Karatsuba Multiplication

Long multiplication 에서의 C_1 은 아래와 같이 계산될 수도 있다.

$$C_1 = (A_0 + A_1) \cdot (B_0 + B_1) - C_0 - C_2$$

이렇게 C_1 을 계산하면 덧셈은 좀 늘지만, 다항식의 곱셈 연산이 한번 줄어든다.

$$\therefore T(N) = 3 \cdot T(N/2) + O(N)$$

4.3 Toom Multiplication

지금까지는 주어진 다항색을 2등분했는데, 이제는 3등분한다. 이번에도 N 이 3의 배수라 치자.

$$A(x) = A_0(x) + A_1(x) \cdot x^{N/3} + A_2(x) \cdot x^{2N/3}$$
$$B(x) = B_0(x) + B_1(x) \cdot x^{N/3} + B_2(x) \cdot x^{2N/3}$$

그러면 $C(x)$ 는 다섯개로 나뉘어진다.

$$A(x) \cdot B(x) = C_0(x) + C_1(x) \cdot x^{N/3} + C_2(x) \cdot x^{2N/3} + C_3(x) \cdot x^{3N/3} + C_4(x) \cdot x^{4N/3}$$

Karatsuba multiplication 이 long multiplication 을 최적화한것처럼, 선형 결합을 이용해서 곱셈의 개수를 이리저리 줄이다 보면

$$T(N) = 5 \cdot T(N/3) + O(N)$$

이 된다. yay.

4.4 Toom–Cook Multiplication

Toom multiplication 을 일반화한 뇌절 버전이다. 강의 내용으로는 이해가 잘 되지 않아 구글링을 통해 이해했다.
Input : $(< N)$ 차 다항식 $A(x)$ 와 $(< M)$ 차 다항식 $B(x)$ 의 계수.
Output : $C(x) := A(x) \cdot B(x)$ 의 계수. $C(x)$ 의 차수는 $N + M - 2$ 이하이다.
$A(x),\ B(x)$ 를 몇 개로 나누는지를 먼저 결정하지 않고, 다항식을 나눌 때 공통적인 기준이 되는 base 차수 (Toom multiplication 에서는 $x^{N/3}$ 이었다.) 를 정한다. 이에 의해서 $A(x),\ B(x)$ 가 몇 개로 나누어지는지가 각각 결정된다. 나누어 떨어지지 않으면 나머지 계수들을 0으로 채운다.
base 차수를 어떤 정수 i 로 결정한 후에는 $A(x)$ 와 $B(x)$ 가 각각 n, m 조각으로 쪼개진다고 생각할 수 있다.

$$y := x^{N/n} = x^{M/m} = x^i$$

따라서 $A(x)$ 와 $B(x)$ 의 형태는

$$
\begin{aligned}
A(x) &= A_0(x) + A_1(x) \cdot y + \cdots + A_{n-1}(x) \cdot y^{n-1} \\
B(x) &= B_0(x) + B_1(x) \cdot y + \cdots + B_{m-1}(x) \cdot y^{m-1}
\end{aligned}
$$

x를 상수처럼 생각했을 때, A와 B를 y 에 대한 다항식이라고도 생각할 수 있다.

$$
\begin{aligned}
A'_x(y) &= A_0 + A_1 \cdot y + \cdots + A_{n-1} \cdot y^{n-1} \\
B'_x(y) &= B_0 + B_1 \cdot y + \cdots + B_{m-1} \cdot y^{m-1}
\end{aligned}
$$

그러면 $C(x)$ 또한 y 에 대한 다항식이라고 생각할 수 있다.

$$C'_x(y) = A'_x(y) \cdot B'_x(y) = C_0 + C_1 \cdot y + \cdots + C_{n+m-2} \cdot y^{n+m-2}$$

$C_0, C_1, \ldots$ 등의 계수들은 x 에 대한 식일 것이기 때문에, 이 친구들을 모두 구하고 $y = x^i$ 를 대입하면 선형 결합을 통해 최종 $C(x)$ 를 구할 수 있게 된다. 이 과정은 $N + M - 2$ 차 식들을 $n + m - 2$ 번 더하므로 $O((N + M) \cdot (n + m))$ 이다.
$C_0, C_1, \ldots$ 를 구하기 위해서, 아래와 같은 행렬 연산이 필요하다. $k := n + m - 2$ 라고 하면,

$$A'_x(y) \cdot B'_x(y) = \begin{bmatrix} 1 & y & \ldots & y^k \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_k \end{bmatrix}$$

이고, 서로 다른 y에 대해서 계속 쌓으면

$$\begin{bmatrix} A'_x(y_0) \cdot B'_x(y_0) \\ A'_x(y_1) \cdot B'_x(y_1) \\ \vdots \\ A'_x(y_k) \cdot B'_x(y_k) \end{bmatrix} = \begin{bmatrix} 1 & y_0 & \ldots & y_0^k \\ 1 & y_1 & \ldots & y_1^k \\ \vdots & \vdots & & \vdots \\ 1 & y_k & \ldots & y_k^k \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_k \end{bmatrix}$$

위 식은 $y_0, \ \ldots, \ y_k$ 에 대한 항등식이므로 아무 상수나 대입해도 성립한다. 이때 대입하는 상수가 모두 distinct 하면 우변의 정사각행렬은 역행렬이 존재한다고 한다. (Vandermonde Matrix)

적당한 숫자(혹은 약속된 숫자)를 잘 대입했다 치고 역행렬을 곱하면

$$\begin{bmatrix} 1 & y_0 & \dots & y_0^k \\ 1 & y_1 & \dots & y_1^k \\ \vdots & \vdots & & \vdots \\ 1 & y_k & \dots & y_k^k \end{bmatrix}^{-1} \begin{bmatrix} A'_x(y_0) \cdot B'_x(y_0) \\ A'_x(y_1) \cdot B'_x(y_1) \\ \vdots \\ A'_x(y_k) \cdot B'_x(y_k) \end{bmatrix} = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_k \end{bmatrix}$$

이다. 역행렬은 k의 크기별로 특정한 상수값들에 대해 미리 다 계산해놓고, 그 결과를 사용하면 된다.
좌변의 열벡터를 보면, y 의 자리에는 모두 상수가 대입되었기 때문에 모든 원소가 x 에 대한 식이다. $A'_x(y_0)$ 은 x 에 대한 N/n 차 식이고 $B'_x(y_0)$ 은 y 에 대한 M/m 차 식이다. 따라서 열벡터의 모든 원소들을 전개하는데 드는 비용은 $(k+1) \cdot T(N/n,\ M/m)$ 이다. 전개 후에는 단순한 행렬곱(선형 결합)을 통해서 $C_0,\ C_1,\ \dots,\ C_k$ 를 알아낼 수 있다.

$$\therefore T(N,\ M) = (n+m-1) \cdot T(N/n,\ M/m) + O((N+M) \cdot (n+m))$$

앞서 다룬 long multiplication, Karatsuba multiplication, Toom multiplication 도 이 식을 잘 따르고 있다.

5 Matrix Multiplication

input: entries of $n \times n$ matrices A , B
output: entries of $C := A \times B$ 우리가 학교 수학시간에 배운대로 naive하게 계산하면 $O(n^3)$ 이지만, 각각의 행렬을 4등분해서 생각하면

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

이것도 이리저리 선형결합해서 곱셈을 줄였을 때,

$$T(n) = 7 \cdot T(n/2) + 18 \cdot (n/2)^2 = O\left(n^{\log_2(7)}\right)$$