CNT@SSHS Lec001 - 1

우리의 레이팅을 올려주는 테크닉들

TAMREF Yun

tamref.yun@snu.ac.kr

April 2, 2019

Contents

- 이 프린트는 CNT@SSHS 강의의 보강 자료입니다. 이론적으로는 강의를 듣지 않고도 수업 내용을 이해할 수 있도록 내용을 충실하게 작성하였습니다. 물론 강의를 듣는 게 훨씬 이해하기에 수월합니다!
- 연습문제 중 **Exercise** 는 손으로 풀 수 있는 수학 문제입니다. 대부분은 맥락을 이해하는 데 매우 중요하니 풀면서 따라오는 것이 좋습니다. 그리 어렵지 않아요 :)
- **Problem**은 실제로 풀어볼 수 있는 온라인 저지 문제들입니다. 문제에 대해서 궁금한 점은 메일로 질문해주시면 답변 드리겠습니다.
- **1.5와 1.6에 들어갈 내용이었던 Xudyh's Sieve와 Mertens Trick은 별도로 작성하고 있습니다.** Xudyh's Sieve를 이해하고 적절히 활용하기 위해서는 1.4 'Dirichlet Convolution'에 대한 감각이 필요합니다. 나중에 작성할 자료에는 Convolution에 대한 설명을 따로 하지 않을 예정입니다.

1 주요 개념과 연습문제들

1.1 Baby step Giant step

1.1.1 Procedure

Baby step Giant step은 정확히 어떤 '알고리즘'이라기보다는 원시근과 관련된 어려운 문제를 조금 덜 어렵게 해결해주는 방법론에 가깝습니다. 강의 시간에 말씀드렸듯, BsGs 알고리즘이 사용되는 대표적인 예시는 DLP(Discrete Logarithm Problem)입니다.

"소수 p에 대한 원시근 g가 주어져 있을 때, $a \equiv g^k$를 만족하는 k의 값은 무엇인가?"

추후에 말씀드리겠지만, 모든 $k = 0 \dots p-1$을 다 집어넣어보는 $\mathcal{O}(p)$ Brute - Force 방법은 **지수 시간**입니다. BsGs를 이용하면 이를 $\mathcal{O}(\sqrt{p})$라는 조금 향상된 시간에 처리할 수 있습니다. 적당한 B를 잡아서, $k = Bq + r$일 것이라고 생각할 수 있습니다. $0 \le r < B$, $0 \le q \le \frac{p-1}{B}$가 성립함에 주목합시다.

가능한 모든 r, 즉 $r = 0 \dots B-1$에 대해 ag^{-r}을 전부 구해둡니다. 그리고 Hash Map과 같이 어떤 값을 삽입, 탐색하는 데 $\mathcal{O}(1)$ 시간이 걸리는 자료구조 X를 데려와서 (ag^{-r}, r)값들을 전부 X에 때려박습니다.[1] 이 과정을 조금씩 걸음한다($\times g$)는 뜻에서 Baby step이라고 부릅니다.

이제 가능한 모든 q를 돌면서 g^{Bq}의 값을 구합니다. q가 증가할 때마다 g^B씩 곱하기 때문에 Giant step입니다. 이제 X에게 "g^{Bq}값 가지고 있냐?"라고 쿼리를 때립니다. g는 원시근이기 때문에 이 값은 유일할 수밖에 없음에 주목합시다. 만약 어떤 r^*가 X안에 있어서 $ag^{-r^*} = g^{Bq}$를 만족했다고 하면, $a = g^{Bq+r^*}$가 되어 우리가 원하는 $k = Bq + r^*$이 얻어집니다. 모든 $a \neq 0$에 대해서 이러한 k는 반드시 존재해야 하기 때문에, 이 알고리즘은 성공적으로 terminate한다는 것이 보장됩니다.

1.1.2 Analysis

이 알고리즘의 공간 복잡도는 $\mathcal{O}(B)$, 시간 복잡도는 $\mathcal{O}(B + p/B)$이기 때문에 $B = \sqrt{p}$로 잡아 시간, 공간 복잡도를 모두 $\mathcal{O}(\sqrt{p})$로 만들 수 있습니다. 간혹 메모리 제한이 더 중요한 경우 B를 더 작게 잡아 trade-off를 꾀할 수도 있습니다.

1.1.3 Problems

BsGs의 본질은 '분할 정복'입니다. 원하는 답 k를 $k = Bq + r$꼴로 놓고, q에 대한 문제와 r에 대한 문제를 해결한 뒤 둘을 합치는 데 시간이 오래 걸리지 않으면 됩니다. 문제는 난이도순이 아닙니다.

Problem 1. 자연수 n을 p진법으로 나타냈을 때의 자릿수 합을 $s_p(n)$이라 합시다. 주어진 $n, X \le 10^{12}$에 대해 $s_p(n) = X$를 만족하는 p의 최솟값을 (존재한다면) 출력하는 프로그램을 작성하세요.

(Scicom TST 2019 Day1 E)

Problem 2. BOJ 16645. Fraction Challenge (kipa cup F)

Problem 3. BOJ 7936. N의 존재 (AMPPZ 2008 I)

[1] 즉, 이 X는 "a가 X안에 있어?"라는 질문에 $\mathcal{O}(1)$ 시간에 Yes/No로 답할 수 있어야 합니다. C++/STL 중에서는 unordered_map, unordered_set 등이 이 기능을 수행할 수 있습니다.

Problem 4. BOJ 10958. Ice Hockey World ChampionShip (CEOI 2015)

1.2 Multiplicative function

이 단원부터, '함수'라고 하는 지칭하는 대상들은 특별한 언급이 없다면 정의역이 자연수인 함수입니다. 이런 함수들을 엮어서 수론함수(arithmetic function)라고 부르기도 합니다.

1.2.1 Definition

어떤 함수 f가 다음을 만족할 때, f를 곱셈함수(multiplicative function)라고 부릅니다.

$$f(1) = 1, \gcd(A, B) = 1 \implies f(A)f(B) = f(AB) \tag{1}$$

따라서 우리는 어떤 n에 대해 $f(n)$값을 알고 싶을 때, n을 소인수분해해서 p^k에 대한 함숫값만 알면 됩니다.

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k} \implies f(n) = f(p_1^{e_1}) \cdots f(p_k^{e_k}) \tag{2}$$

1.2.2 Erathosthenes' Sieve

1에서 N까지의 소수를 전부 구하고 싶을 때 가장 자연스럽게 사용할 수 있는 알고리즘입니다. 지면상 과정을 서술하지는 않겠지만, 이 알고리즘의 시간복잡도를 알아두는 것이 중요합니다. 1에서 n까지의 소수를 전부 구하는 데에 드는 시간은

$$T(n) = \mathcal{O}(\sum_{p<n} \frac{n}{p}) \tag{3}$$

가 되고, 다음이 알려져 있습니다.

$$\sum_{p<n} \frac{1}{p} = \ln \ln n + \mathcal{O}(1) \tag{4}$$

따라서 $T(n) = \mathcal{O}(n \ln \ln n)$이 됩니다.

1.2.3 Tabulating Multiplicative Functions

곱셈함수의 성질이 PS에서 유용한 이유는, 수론함수 $f(n)$이 곱셈함수일 때 $f(1), f(2), \cdots f(n)$을 tabulate 하기 매우 쉽기 때문입니다.

가령 n의 약수의 개수 $\tau(n)$에 대한 표를 구해놓고 싶다고 합시다. 다음이 알려져 있습니다.

$$n = \prod_i p_i^{e_i} \implies \tau(n) = \prod_i (1 + e_i) \tag{5}$$

이 때 $\tau(n)$을 에라토스테네스의 체를 이용하면 효과적으로 구할 수 있습니다. p의 배수들을 (소수의 후보에서) cancel out하는 과정에서 $1+e_i$를 간단하게 구할 수 있기 때문입니다.

Exercise. $\tau(n)$에 대한 표를 구하는 위 과정의 시간 복잡도를 구해 보세요.

뒤에서도 강조하겠지만, **multiplicative한 함수는 tabulate할 수 있습니다.**

1.2.4 Problems

Problem 1. BOJ 1016. 제곱 ㄴㄴ 수

Problem 2. BOJ 16136. 준하의 정수론 과제

Problem 3. Euler totient function $\varphi(n)$에 대한 설명은 아래 장을 참조하세요.

$$\varphi^*(n) := \begin{cases} \varphi(n) & (n : even) \\ \varphi(n)/2 & (n : odd) \end{cases} \tag{6}$$

과 같이 정의된 함수가 있을 때, 이 함숫값을 tabulate하는 프로그램을 작성하세요. 또, 이 함수는 곱셈함수가 아닙니다. 적절한 곱셈함수들을 이용해서 φ^*를 표현해보세요.

1.3 Mőbius function

1.3.1 Inclusion - Exclusion Principle

$$|B_1 \cup B_2 \cup \cdots B_n| = \sum_{P \subset \{1,2,\cdots,n\}} (-1)^{|P|} \left| \bigcap_{s \in P} B_s \right| \tag{7}$$

1.3.2 Euler totient function

N이하의 N과 서로소인 자연수의 개수를 $\varphi(N)$으로 표기하고, Euler totient function이라 부릅니다.

$\varphi(N)$을 구하는 공식의 유도법은 많이 있지만, 다음의 포함 - 배제 원리를 이용한 버전이 가장 확장성이 좋습니다. 어떤 수가 N과 서로소이려면 그 어떤 소수 p도 $\gcd(k, N)$을 나눌 수 없으므로, $B_p := \{k : p | \gcd(k, N)\}$으로 잡고 포함배제의 식을 열심히 전개하면 다음이 얻어집니다.

$$\varphi(N) = N - \left|\bigcup_{p\le N} B_p\right| \tag{8}$$

$$= N - \sum_{P\subset\{2,3,5,\cdots,p\le N\}} (-1)^{|P|}\left|\bigcap_{s\in P} B_s\right| \quad (P\neq\emptyset) \tag{9}$$

$$= \sum_{P\subset\{p\le N\}} (-1)^{|P|}\left|\bigcap_{s\in P} B_s\right| \tag{10}$$

$$= \sum_{P\subset\{\mathbf{p|N}\}} (-1)^{|P|}\left|\bigcap_{s\in P} B_s\right| \quad (\because \gcd(N,p)=1 \implies B_p=\emptyset) \tag{11}$$

$$= \sum_{P\subset\{p|N\}} (-1)^{|P|}\frac{N}{s_1 s_2\cdots s_{|P|}} \quad (P=\{s_1,s_2,\cdots s_{|P|}\}) \tag{12}$$

$$N\cdot \sum_{P\subset\{p|N\}} \left(-\frac{1}{s_1}\right)\left(-\frac{1}{s_2}\right)\left(-\frac{1}{s_3}\right)\cdots\left(-\frac{1}{s_{|P|}}\right) = N\cdot\prod_{p|N}\left(1-\frac{1}{p}\right) \blacksquare \tag{13}$$

이 '소인수에 대한 포함 배제'를 좀 더 체계적으로 해보고 싶었던 우리는 한 가지 아이디어를 냅니다. N의 모든 약수 d에 대해서 적당한 함수 $\mu(d)$가 있어서, $\varphi(N)$을

$$\varphi(N) = \sum_{d|N} \mu(d)\cdot\frac{N}{d} \tag{14}$$

과 같은 식으로 나타낼 수 있지 않을까요?

1.3.3 Mőbius function

우리의 바람이 반영된 함수를 뫼비우스 함수 (Mőbius function)라고 부릅니다. $\mu(n)$으로 표기되는 함수는 다음과 같이 정의됩니다:

$$\mu(n) = \begin{cases} 0 & (n = p^2\cdot m) \\ (-1)^k & (n = p_1p_2\cdots p_k) \end{cases} \tag{15}$$

Exercise 1. $\varphi(N) = \sum_{d|N}\mu(d)\frac{N}{d}$이 성립함을 확인하세요.

Exercise 2. $q(N)$을 $\gcd(x,y)=1,\ 1\le x,y\le N$을 만족하는 (x,y)의 개수로 정의합시다. $q(N)$을 divisor sum $(\sum_{d|N})$과 $\mu(*)$을 활용한 식으로 나타내어 보세요.

Exercise 3. $\sum_{d|N} \mu(d)$의 값을 계산해보세요. 그 값은 $N = 1$이면 1, 아니면 0이 됩니다. 이 값을 주는 함수를 $I(N)$이라고 부릅니다. (Hint : $N > 1$이면 N을 나누는 소수 p가 무조건 존재합니다.)

Exercise 4. $\sum_{d|N} \mu(d)\sigma\left(\frac{N}{d}\right) = N$이 성립함을 보이세요. $\sigma(N)$은 N의 약수의 합을 의미합니다. **Mőbius inversion을 명시적으로 사용하지 말고, 포함 배제의 원리로 접근해보세요.**

1.3.4 Mőbius inversion formula

Mőbius inversion을 사용하지 말라는 문제 바로 밑에 inversion을 넣기 좀 그렇지만, 인생이 원래 생각한 대로만 흘러가지 않습니다.

포함 - 배제 원리는 분명히 유용하지만, 고려해야 할 변수의 개수에 따라 복잡도가 exponential하게 증가한다는 단점이 있습니다. 우리의 계산량을 줄여주는 아름답고도 간편한 공식이 바로 Mőbius inversion formula입니다.

$$g(n) = \sum_{d|n} f(d) \Longleftrightarrow f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) \tag{16}$$

증명은 처음 보기엔 tricky하지만 결국 시그마를 열심히 뒤집으면 됩니다. 한쪽 방향만 적어드렸지만 반대 방향의 증명도 원리가 동일합니다.

$$\sum_{d|n} \mu(d) g\left(\frac{n}{d}\right) = \sum_{d|n} \mu(d) \sum_{e|\frac{n}{d}} g(e) \tag{17}$$

$$= \sum_{de|n} \mu(d) g(e) \tag{18}$$

$$= \sum_{e|n} g(e) \sum_{d|\frac{n}{e}} \mu(d) \tag{19}$$

$$= \sum_{e|n} g(e) I\left(\frac{n}{e}\right) = g(n) \blacksquare \tag{20}$$

이 사실을 이용하여 1.3.3의 **Exercise 4**를 간단히 해결할 수 있습니다.
또한, $\varphi(n) = \sum_{d|n} \mu(d)\frac{n}{d}$로부터 $\sum_{d|n} \varphi(d) = n$도 쉽게 얻어집니다. 함수들 간의 연결 관계가 풍성해졌네요!

1.3.5 Why Mőbius?

위의 논의로부터, $\mu(n)$이 수론함수들 간의 매개체로 유용하게 쓰인다는 것은 알 수 있습니다. 그런데 PS 러인 우리는 왜 $\mu(n)$을 알아야 할까요?

그 이유는 바로 $\mu(n)$이 그 자체로 곱셈함수이기 때문입니다. 곱셈함수에 대해서 우리는 대략 $\mathcal{O}(n)$만에 $1..n$의 모든 값을 구할 수 있습니다! 따라서 한 번 μ값을 전처리해서 쟁여두면, 함수값을 매번 구할 필요 없이 배열에서 $\mathcal{O}(1)$만에 가져다 쓸 수 있기 때문입니다.

1.3.6 Problems

Problem 1. BOJ 14958. 세 쌍 서로수

Problem 2. Codeforces 1139D. Steps to One

Problem 3. BOJ 1557. 제곱 ㄴㄴ

1.4 Dirichlet Convolution

1.4.1 Definition

우리의 위대한 수학자들은 다음의 아름다운 식을 보고 도저히 참을 수가 없었습니다.

$$\varphi(n) = \sum_{d|n} \mu(d)\frac{n}{d} \tag{21}$$

그래서 그들은 다음과 같은 연산을 정의함으로써, 더 일반적이고 더 많은 성질을 얻고자 했습니다. 이 연산을 Dirichlet Convolution이라고 부릅니다.

$$(f * g)(n) := \sum_{d|n} f(d)g(\frac{n}{d}) \tag{22}$$

Convolution의 언어로 표현하면 (21)은 $\varphi = \mu * N$과 같이 쓸 수 있습니다. ($N(n) := n$)

$*$는 '연산자'로서의 표준적인 성질들을 전부 만족합니다.

- $f * g = g * f$ - 교환법칙
- $(f * g) * h = f * (g * h)$ - 결합법칙 (증명해보세요!)
- $f * I = f$ - 항등원의 존재성 ($I(n)$의 정의를 생각해봅시다.)
- $f(1) \neq 0 \implies f^{-1}$은 유일하게 존재. (역원의 존재성. 귀납법으로 증명해 보세요!)

$u(n) = 1$이라고 정의하면, 1.3.3.의 **Exercise 3**은 다음과 같이 그 형태가 바뀝니다.

$$\mu * u = I \tag{23}$$

그리고 (16), 즉 Mőbius inversion theorem은 convolution의 언어로 비교적 자명하게 서술됩니다.

$$g = f * u \Longleftrightarrow f = g * \mu \tag{24}$$

사실 Xudyh's Sieve 등에서 다룰 Dirichlet Convolution은 Convolution이 가진 무수한 성질 중 극히 일부만을 이용합니다. 이 연산 자체에 흥미가 있으신 분들에게는 해석적 정수론(Apostol)을 읽는 것을 추천드립니다.

Alert. 이 장의 연습문제들은 앞으로 다룰 내용과 큰 연관이 없습니다. 흥미롭지 않다면 스킵해주세요.

Exercise 1. Dirichlet convolution은 곱셈함수의 성질을 보존함을 보이세요. 즉, f, g가 곱셈함수이면 $f * g$도 곱셈함수임을 보이세요.

Exercise 2. $f, f * g$가 곱셈함수이면, g도 곱셈함수임을 보이세요.

Exercise 3. f가 곱셈함수이고 $f(1) \neq 0$이면, f^{-1}역시 곱셈함수임을 보이세요.

Exercise 4. $f(n)$이 모든 $ab = n$ (a, b는 서로소일 필요 없음)에 대해 $f(n) = f(a)f(b)$를 만족할 때, f를 Completely multiplicative(c.m.)라고 부릅니다.

- f, g가 c.m.일 때, $f * g$도 c.m.인가요?
- 곱셈함수 f가 c.m.임은 모든 소수 p에 대해 $f(p^k) = f(p)^k$가 성립하는 것과 동치임을 보이세요.
- 곱셈함수 f가 c.m.임은 $f^{-1} = \mu \cdot f$임과 동치임을 보이세요. μ와 f의 convolution이 아님에 주의하세요.