TÜBİTAK - BİLGEM
Selman Pınar
Haziran 2025

1 Discrete Logarithm Problem (DLP) - Ayrık Logaritma Problemi

Herhangi bir G grubunda g ve h grubun elemanları ise $g^x = h$ eşitliğini sağlayan x değerini bulmaya *ayrık logaritma problemi* denir. Her zaman böyle bir x bulunmayabilir. Örneğin, $(\mathbb{Z}_7, \cdot)$ grubunda $2^x = 3$ eşitliğini sağlayan x değeri yoktur. Eğer böyle bir x varsa o zaman bu x değeri $x = \log_g h$ olarak gösterilir. Örneğin, $(\mathbb{Z}_7, \cdot)$'de $3^x = 6$ eşitliği $x = 3$ sağladığı için $(\mathbb{Z}_7, \cdot)$'de $\log_3 6 = 3$ olmuş olur.

Eğer bir grubun tüm elemanları, grubun bir elamanı olan g'nin kuvvetleri şeklinde yazılabiliyorsa bu g'ye *ilkel kök* (primitive root) denir. O halde $g^x = h$ denkleminin her zaman çözümünün olmasını istiyorsak g'yi ilkel kök seçmemiz lazım. Örneğin $(\mathbb{Z}_7, \cdot)$'de sadece 3 ve 5 ilkel köklerdir. Dolayısıyla her $h \in (\mathbb{Z}_7, \cdot)$ için $3^x = h$ ve $5^x = h$ denklemlerinin her zaman çözümü vardır.

Belirli bir asal moduluste üst almak neredeyse rastgeleye yakın çıktılar verir. Bu rastgelelik ayrık logaritma problemini çözmeyi zorlaştırır. Örneğin mod 997'de 7^i değerinin farklı i'ler için çıktıları Figure 1'de verilmiştir.

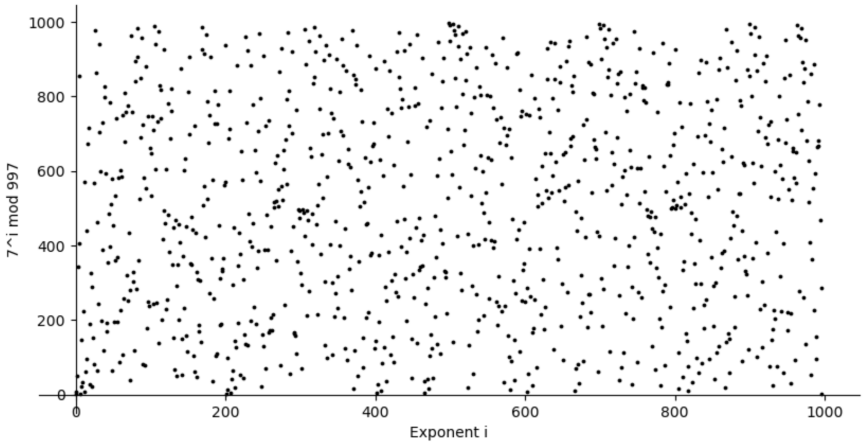

Figure 1: mod 997'de 7^i

2 Brute Force Attack - Deneme Yanılma Saldırısı

Bir G grubunun elemanı olan g sayısının mertebesi N olsun. (Yani $g^N = 1$ ve N'den küçük başka bir n için $g^n \neq 1$) O halde $g^x = h$ denklemini sağlayan x değeri, yani ayrık logaritma problemi, $\mathcal{O}(N)$ çarpma işlemi ve $\mathcal{O}(1)$ hafıza turarak çözülür. Zira $g, g^2, g^3, \cdots, g^N$ elemanlarından biri h sayısına eşit olacaktır. Sürekli bir önceki sayının üstünü aldığımız için de aynı ayna 2 sayıyı hafızada tutmak yeterli olacaktır. Eğer bir elemanın mertebesini bilmiyorsak Lagrange teoremi yardımı ile mertebesini deneme yanılma ile bulabiliriz.

3 BabyStep GiantStep Algorithm - Küçük Adım Dev Adım Algoritması

Yukarıdaki deneme yanılma yöntemi $\mathcal{O}(N)$ işlemde ayrık logaritmayı bulmamızı sağlıyordu. Ancak büyük N'ler için bu kullanışsızdır. BabyStep GiantStep algoritması bunu $\mathcal{O}(\sqrt{N}\log N)$ adıma düşürür ancak $\mathcal{O}(\sqrt{N})$ hafıza tutmamız gerekir. Algoritma şu şekilde işliyor:
G bir grup, $g \in G$ ve g'nin mertebesi N olsun. $g^x = h$ denklemini çözmek için

1. $n = 1 + \lfloor N \rfloor$ olsun.
2. İki tane liste oluşturalım.
   (a) Liste 1: $1, g, g^2, \cdots, g^n$ (Baby Step)
   (b) Liste 2: $h, hg^{-n}, hg^{-2n}, \cdots, hg^{-n^2}$ (Giant Step)
3. Bu iki listede birbirine denk gelen iki elemanı bulalım. Diyelim ki $g^i = hg^{-jn}$
4. O halde $x = i + jn$ ifadesi $g^x = h$ denkleminin çözümü olmuş olur.

İspat:
Şimdi bu algoritmanın neden işe yaradığına bakalım. Diyelim ki x yukarıdaki ayrık logaritmanın sonucu olsun. Bölme algoritmasından $x = nq + r, 0$ ve $\leq r < n$ olacak şekilde q ve r vardır. O halde

$$q = \frac{x - r}{n} < \frac{N}{n} < n$$

olur zira $x < N$ ve $\sqrt{N} < n$. Yani $0 \leq r < n$ ve $0 \leq q < n$ olacak şekilde q ve r vardır. Dolayısıyla

$$g^x = h \implies g^{nq+r} = h \implies g^r = hg^{-nq}$$

g^r birinci listede hg^{-nq} de ikinci listededir.
Algoritmanın $\mathcal{O}(\sqrt{N}\log N)$ hamlede olmasının sebebi ise 2. hamledeki her bir listeyi oluşturken n çarpma işlemi yani totalde $2n$ çarpma işlemi kullanmamız ve 3. hamledeki iki listenin ortak elamanını bulma algoritmasının (sorting and searching algorithm) $\mathcal{O}(n\log n)$ adımda olmasıdır. $n \approx \sqrt{N}$ olduğundan $\mathcal{O}(n\log n) = \mathcal{O}(\sqrt{N}\log N)$ olur.
Liste 1 ve Liste 2, n uzunluğunda olduğundan $\mathcal{O}(\sqrt{N})$ hafıza tutmamız gerekir. $\mathcal{O}(\sqrt{N}\log N)$ ifadesinde büyük N'ler için $\log N$ ifadesi $\sqrt{N}$ ifadesine göre ihmal edilebilir düzeyde küçük olduğu için bu algoritma yaklaşık olarak $\mathcal{O}(\sqrt{N})$ hamleye düşürüyor diyebiliriz. □

Örneğin, $9704^x = 13896$ denklemini $\mathbb{Z}^*_{17389}$ grubunda çözmeye çalışalım.

1. 9704 elemanının mertebesi 1242'dir. (1242 sayısının 17388 sayını tam böldüğüne dikkat ediniz, (Lagrange teoremi)) $n = 1 + \left\lfloor \sqrt{1242} \right\rfloor = 36$
2. Liste 1 ve Liste 2'yi oluşturalım. $u = g^{-n}$ için Figure 2'de Liste 1 ve Liste 2 tablo şeklinde verilmiştir.
3. Figure 2'de Liste 1 ve Liste 2'de ortak olan sayı 14567'dir. Yani $i = 7$ ve $j = 32$ olur.
4. $x = i + jn = 7 + 36 \cdot 32 = 1159$ ayrık logaritmanın çözümüdür.

k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$	k	g^k	$h \cdot u^k$
1	9704	347	9	15774	16564	17	10137	10230	25	4970	12260
2	6181	13357	10	12918	11741	18	17264	3957	26	9183	6578
3	5763	12423	11	16360	16367	19	4230	9195	27	10596	7705
4	1128	13153	12	13259	7315	20	9880	13628	28	2427	1425
5	8431	7928	13	4125	2549	21	9963	10126	29	6902	6594
6	16568	1139	14	16911	10221	22	15501	5416	30	11969	12831
7	**14567**	6259	15	4351	16289	23	6854	13640	31	6045	4754
8	2987	12013	16	1612	4062	24	15680	5276	32	7583	**14567**

Figure 2: Liste 1 ve Liste 2

4 Pohlig - Hellman Algorithm

Ayrık logaritma probleminin $(\mathbb{Z}_p, \cdot)$ grubunda eğer bir tane çözümü varsa $\mathbb{Z}$'de sonsuz tane çözümü olur. Zira Fermat'ın küçük teoremine göre $g^{p-1} \equiv 1 \pmod{p}$ olduğundan eğer $g^x = h$ denkleminde x bir çözümse $x + k(p-1)$ de bir çözümdür. Dolayısıyla $(\mathbb{Z}_p, \cdot)$'de ayrık logaritmanın fonksiyon olmasını istiyorsak mod $p-1$'de tanımlamamız gerekir. Eğer $p-1$ çarpanlarına iyi ayrılabilen bir sayı ise o zaman ayrık logaritma problemini bu çarpanlarda çözüp daha sonrasında çin kalan teoremi ile genel çözümü bulabiliriz. Pohlig - Hellman algoritması tam olarak bunu yapmaya çalışır. Daha genel ifade ile G bir grup ve g bu grubun N mertebeli elamanı ise G'de $g^x = h$ denkleminin sonucu mod N'de incelenir. Yani N'yi çarpanlarına ayırmak mantıklı olabilir. Şimdi algoritmaya bakalım.

Diyelim ki G grubunda elimizde ayrık logaritma problemini asal mertebeli bir elaman için çözen bir algoritma olsun. Yani eğer a, grubun asal kuvvet mertebeli (q^e mertebeli) bir elemanı ise $a^x = h$ denklemini $O(S_{q^e})$ hamlede çözüyor olalım. Mesela BabyStep GiantStep yaklaşık olarak $O(q^{\frac{e}{2}})$ hamlede çözdüğünü görmüştük. O halde en kötü ihtimalle $\mathcal{O}(S_{q^e}) = \mathcal{O}(q^{\frac{e}{2}})$ olarak alabiliriz.

Şimdi diyelim ki g, bu grubun N mertebeli bir elemanı olsun ve $N = q_1^{e_1} q_2^{e_2} \cdots q_t^{e_t}$ şeklinde çarpanlarına ayrılsın. $g^x = h$ denklemi

$$\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}} + \log N\right)$$

hamlede aşağıdaki algoritma kullanılarak çözülebilir:

1. Her $1 \leq i \leq t$ için
$$g_i = g^{N/q_i^{e_i}} \quad \text{ve} \quad h_i = h^{N/q_i^{e_i}} \quad \text{olsun.}$$
g_i asal kuvvet mertebeli ($g_i^{(q_i^{e_i})} = 1$) olduğu için yukarıda bahsedilen algoritma ile $g_i^y = h_i$ denklemini çöz. Çözüm $y = y_i$ olsun.

2. Çin-Kalan teoremini kullanarak
$$x \equiv y_1 \pmod{q_1^{e_1}}, \quad x \equiv y_2 \pmod{q_2^{e_2}}, \quad \cdots, \quad x \equiv y_1 \pmod{q_t^{e_t}}$$
siteminin çöz.

İspat:
$x = y_i + q_i^{e_i} z_i$ yukarıdaki sistemin çözümü olsun.

$$\begin{aligned}(g^x)^{N/q_i^{e_i}} &= (g^{y_i + q_i^{e_i} z_i})^{N/q_i^{e_i}} \\ &= (g^{N/q_i^{e_i}})^{y_i} \cdot g^{N z_i} \\ &= (g^{N/q_i^{e_i}})^{y_i} \\ &= g_i^{y_i} \\ &= h_i \\ &= h^{N/q_i^{e_i}}\end{aligned}$$

olmuş olur. Her tarafın g tabanına göre ayrık logaritmasını alalım.

$$\frac{N}{q_i^{e_i}} \cdot x = \frac{N}{q_i^{e_i}} \cdot \log_g(h) \pmod N$$

ayrık logaritma mod N'de tanımlı olduğu için en sonda mod N aldık. Şimdi

$$\frac{N}{q_1^{e_1}}, \quad \frac{N}{q_2^{e_2}}, \quad \cdots \quad \frac{N}{q_t^{e_t}}$$

terimlerinin hiçbirinin ortak çarpanı olmadığından hepsi ayrı ayrı aralarında asallar. O halde genelleştirilmiş çin kalan teoreminden

$$\frac{N}{q_1^{e_1}} \cdot c_1 + \frac{N}{q_2^{e_2}} \cdot c_2 + \cdots + \frac{N}{q_t^{e_t}} \cdot c_t = 1$$

eşitliğini yazabiliriz. Yukarıdaki eşitliği c_i ile çarpıp tüm i'ler altında toplarsak

$$\sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot x = \sum_{i=1}^{t} \frac{N}{q_i^{e_i}} \cdot c_i \cdot \log_g(h) \pmod N,$$

eşitliğini elde ederiz. İki üsteki denklemden dolayı

$$x = \log_g(h) \pmod N.$$

elde edilir ve bu da x'in çözüm olduğunu göstermiş olur.

Birinci adım $\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}}\right)$ hamlede tamamlanır. İkinci adım da çin kalan teoremi olduğundan $O(\log N)$ hamlede tanımlanır ve dolayısıyla adım sayısı da $\mathcal{O}\left(\sum_{i=1}^{t} S_{q_i^{e_i}} + \log N\right)$ olmuş olur. □

Eğer grubun mertebesi küçük çarpanlara ayrılabiliyorsa Pohlig-Hellman algoritması bize ayrık logaritma probleminin çok da güvenli olmadığını gösteriyor. Yani $g^x = h$ denklemini çözmek eğer g sayısının mertebesi küçük çarpanlara ayrılıyorsa kolaydır. $(\mathbb{Z}_p, \cdot)$ grubunda grubun mertebesi $p-1$ olduğunu biliyoruz. Ve büyük asallar için $p-1$ her zaman çift sayı olduğundan $p-1$'in 2 ile bölünmüş halini asal seçersek bu problemden kurtuluruz. Yani q asalı için p asalını $2q+1$ formatında seçmek elimizdeki en iyi seçenek olur. Böylece ayrık logaritma problemi $\mathcal{O}(\sqrt{q})$ hamlede çözülebilir. Eğer q sayısını büyük seçersek problemi çözmek epey zorlaşır.

Pohlig–Hellman algoritması, kabaca, keyfi mertebeden elemanlar için ayrık logaritma problemini, asal kuvvet mertebeli (p asalı a pozitif sayısı için p^a) elemanlar için ayrık logaritma

problemine indirger. Aşağıdaki teoremle birlikte bunu asal kuvvet mertebesinden sadece asal mertebeye indireceğiz.

Teorem:
G bir grup ve a bu grubun asal q mertebeli bir elamanı olsun. Diyelim ki $a^x = h$ denklemini S_q hamlede çözen bir algoritmamız var. O halde $g \in G$ ve g'nin mertebesi q^e ise $g^x = h$ denklemi $\mathcal{O}(eS_q)$ hamlede çözülebilir.

İspat:
x değerini aşağıdaki gibi yazalım:

$$x = x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1} \quad \text{ve } 0 \leq x_i < q$$

ve ardından sırasıyla $x_0, x_1, x_2, \ldots$ değerlerini belirlersek problemi çözmüş oluruz. $g^{q^{e-1}}$ elemanının q mertebesinde olduğundan

$$\begin{aligned} h^{q^{e-1}} &= (g^x)^{q^{e-1}} \\ &= \left(g^{x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}}\right)^{q^{e-1}} \\ &= g^{x_0 q^{e-1}} \cdot \left(g^{q^e}\right)^{x_1 + x_2 q + \cdots + x_{e-1} q^{e-2}} \\ &= \left(g^{q^{e-1}}\right)^{x_0} \end{aligned}$$

şeklinde yazabiliriz.

$$\left(g^{q^{e-1}}\right)^{x_0} = h^{q^{e-1}}$$

Varsayım gereği yukarıdaki problemi S_q adımda çözebiliriz. Bu çözüldüğünde,

$$g^{x_0 q^{e-1}} = h^{q^{e-1}}$$

x_0 değerini bulmuş oluruz. Daha sonra benzer bir hesaplamayla

$$\begin{aligned} h^{q^{e-2}} &= (g^x)^{q^{e-2}} \\ &= \left(g^{x_0 + x_1 q + x_2 q^2 + \cdots + x_{e-1} q^{e-1}}\right)^{q^{e-2}} \\ &= g^{x_0 q^{e-2}} \cdot g^{x_1 q^{e-1}} \cdot \left(g^{q^e}\right)^{x_2 + x_3 q + \cdots + x_{e-1} q^{e-3}} \\ &= g^{x_0 q^{e-2}} \cdot g^{x_1 q^{e-1}} \end{aligned}$$

değerini elde ederiz. x_0 değerini zaten biliyoruz ve yine $g^{q^{e-1}}$ elemanının G'de q mertebesine sahip olduğundan x_1'i bulmak için,

$$\left(g^{q^{e-1}}\right)^{x_1} = \left(h \cdot g^{-x_0}\right)^{q^{e-2}}$$

denklemini çözmeliyiz. Verilen algoritmayı tekrar uygulayarak, bunu S_q adımda çözebiliriz. Böylece $2S_q$ adımda,

$$g^{(x_0 + x_1 q) q^{e-2}} = h^{q^{e-2}}$$

şartını sağlayan x_0 ve x_1 değerlerini belirlemiş olduk. Benzer şekilde, x_2'yi

$$\left(g^{q^{e-1}}\right)^{x_2} = \left(h \cdot g^{-x_0 - x_1 q}\right)^{q^{e-3}}$$

problemini çözerek buluruz. Genel olarak, $x_0, \ldots, x_{i-1}$ değerlerini belirledikten sonra, x_i değeri

$$\left(g^{q^{e-1}}\right)^{x_i} = \left(h \cdot g^{-x_0 - x_1 q - \cdots - x_{i-1} q^{i-1}}\right)^{q^{e-i-1}}$$

çözülerek elde edilir. Bunların her biri tabanı q mertebesinde olan ayrık logaritma problemleridir, dolayısıyla her biri S_q adımda çözülebilir. Sonuç olarak $\mathcal{O}(eS_q)$ adım sonunda, $g^x = h$ eşitliğini sağlayan ve böylece orijinal ayrık logaritma problemini çözen bir $x = x_0 + x_1 q + \cdots + x_{e-1} q^{e-1}$ değerini elde ederiz. □

Örnek:

$$\mathbb{F}^*_{11251}\text{'de} \quad 5448^x = 6909$$

denklemini $\mathbb{F}^*_{11251}$ grubunda çözmeye çalışalım. $p = 11251$ asal sayısının $p - 1$ değeri 5^4'e bölünebilir ve 5448'in 5^4 mertebesine olduğu kolayca doğrulanabilir. İlk adım,

$$\left(5448^{5^3}\right)^{x_0} = 6909^{5^3},$$

denklemini çözmektir. Bu denklem $11089^{x_0} = 11089$ şeklinde sadeleşir. Bariz olduğu üzere cevap $x_0 = 1$'dir, dolayısıyla x'in başlangıç değeri $x = 1$ olur. Sonraki adım,

$$\left(5448^{5^3}\right)^{x_1} = \left(6909 \cdot 5448^{-x_0}\right)^{5^2} = \left(6909 \cdot 5448^{-1}\right)^{5^2},$$

denklemini çözmektir. Bu denklem $11089^{x_1} = 3742$ şeklinde sadeleşir. x_1 değerini bulmak için yalnızca 1'den 4'e kadarki değerleri kontrol etmemiz yeterlidir zira 11089'un mertebesi 5'tir. Ancak q büyük olsaydı, BabyStep GiantStep gibi daha hızlı bir algoritma kullanmak faydalı olurdu. Her durumda, çözüm $x_1 = 2$'dir, dolayısıyla x değeri şimdi $x = 11 = 1 + 2 \cdot 5$ olur. Devam edelim:

$$\left(5448^{5^3}\right)^{x_2} = \left(6909 \cdot 5448^{-x_0 - x_1 \cdot 5}\right)^5 = \left(6909 \cdot 5448^{-11}\right)^5$$

denklemini çözüyoruz. Bu denklem $11089^{x_2} = 1$ şeklinde sadeleşir. Böylece $x_2 = 0$ olur, bu da x değerinin $x = 11$ olarak kalması anlamına gelir. Son adım,

$$\left(5448^{5^3}\right)^{x_3} = 6909 \cdot 5448^{-x_0 - x_1 \cdot 5 - x_2 \cdot 5^2} = 6909 \cdot 5448^{-11}$$

denklemini çözmektir. Bu denklem $11089^{x_3} = 6320$ şeklinde sadeleşir ve çözümü $x_3 = 4$'tür. Dolayısıyla son cevabımız:

$$x = 511 = 1 + 2 \cdot 5 + 4 \cdot 5^3$$

Şimdi Pohlig-Hellman algoritması için bir örnek yapalım.

Örnek:

Aşağıdaki ayrık logaritma problemini ele alalım:

$$\mathbb{F}^*_{11251}\text{'te} \quad 23^x = 9689$$

23 tabanı $\mathbb{F}^*_{11251}$ içinde bir ilkel köktür, yani mertebesi 11250'dir. $11250 = 2 \cdot 3^2 \cdot 5^4$ küçük asal sayıların çarpımı olduğundan, Pohlig-Hellman algoritmasını kullanabiliriz. Değerler açık gözüksün diye tekrar yazalım:

$$p = 11251, \quad g = 23, \quad h = 9689, \quad N = p - 1 = 2 \cdot 3^2 \cdot 5^4.$$

İlk adım, aşağıdaki tabloda gösterildiği gibi üç yardımcı ayrık logaritma problemini çözmektir.

q	e	$g^{(p-1)/q^e}$	$h^{(p-1)/q^e}$	Çözülecek denklem $\left(g^{(p-1)/q^e}\right)^x = h^{(p-1)/q^e}$
2	1	11250	11250	$x = 1$
3	2	5029	10724	$x = 4$
5	4	5448	6909	$x = 511$

İlk problemin çözümü bariz, üçüncü problemi yukarıdaki örnekte çözmüştük. Yani her adım, yukarıdaki teoremin ispatındaki taktikle çözülebilir. İkinci adım, aşağıdaki sistemi Çin Kalan Teoremi'ni kullanarak çözmektir:

$$x \equiv 1 \pmod 2, \quad x \equiv 4 \pmod{3^2}, \quad x \equiv 511 \pmod{5^4}.$$

Çin Kalan teoremi sonucu çözümü $x = 4261$ olarak buluruz.

5 Probabilistic Collision Algorithm

Elimizde N eleman içerek bir liste olsun. Bu listeden rastgele n **farklı** eleman seçelim ve seçtiklerimizi liste 1'e ekleyelim. Seçtiğimiz elemanlar $x_1, x_2, \cdots, x_n$ olsun. Yani

$$\text{Liste 1: } (x_1, x_2, \cdots, x_n)$$

Daha sonra listeden rastgele 1 eleman seçelim, ve liste 2'e ekleyelim. Tekrardan rastgele bir eleman seçelim ve yine liste 2'e ekleyelim ve bunu n defa tekrar edelim. Seçtiğimiz elemanlar $z_1, z_2, \cdots, z_n$ olsun (Hepsinin farklı olmasına gerek yok). Yani

$$\text{Liste 2: } (z_1, z_2, \cdots, z_n)$$

Liste 1 ve liste 2'de en az bir tane ortak elemanın olma olasılığı kaçtır?
Çözüm:

$$\begin{aligned} Pr\left(\text{En az bir tane ortak eleman olması}\right) &= 1 - Pr\left(\text{Hiç ortak eleman olmaması}\right) \\ &= 1 - \prod_{i=1}^{n} Pr\left(z_i\text{'nin liste 1'de olmaması}\right) \\ &= 1 - \left(\frac{N-n}{N}\right)^n \\ &= 1 - \left(1 - \frac{n}{N}\right)^n \\ &\geq 1 - e^{-n^2/N} \end{aligned}$$

Eğer $n = 3\sqrt{N}$ seçersek olasılığımız %99.98 olur. Yani çok büyük ihtimalle ortak eleman bulacağız. Şimdi bu taktiği ayrık logaritma problemi için kullanalım.
Discrete Logarithm Collision Algorithm
G grubunda g elemanının mertebesi N olsun. $g^x = h$ denkleminin çözümü varsa, her hamle G grubunda üst alma olduğunda $\mathcal{O}(\sqrt{N})$ hamlede aşağıdaki şekilde bulunabilir:

1. $x = y - z$ şeklinde yazalım ve $g^y = hg^z$ denkleminin sonucunu bulmak için g^y'lerden ve hg^z'lerden oluşan 2 liste hazırlayalım.

2. $y_1, y_2, \cdots, y_n$ 1 ile N arasında rastgele farklı tam sayılar olsun.

$$\text{Liste } 1 = (g^{y_1}, g^{y_2}, \cdots g^{y_n})$$

3. $z_1, z_2, \cdots, z_n$ de 1 ile N arasında rastgele tam sayılar olsun.

$$\text{Liste } 2 = (hg^{z_1}, hg^{z_2}, \cdots hg^{z_n})$$

4. Eğer $n = 3\sqrt{N}$ seçersek bu iki listede ortak eleman olma ihtimali %99.98'dan büyük olur ve eğer ortak elemanlar $g^{y_i} = hg^{z_j}$ ise $x = y_i - z_j \pmod N$ çözüm olmuş olur.

Liste 1 ve liste 2, n eleman içerir, bu nedenle her bir listeyi oluşturmak yaklaşık $2n$ adım alır. Daha kesin olarak, her listedeki her eleman için 1 ile N arasında bir i değeri için g^i hesaplamamız gerekir ve g^i'yi hesaplamak, hızlı üs alma algoritması kullanılarak yaklaşık $2\log_2(i)$ grup çarpımı alır. Böylece, iki listeyi oluşturmak yaklaşık $4n\log_2(N)$ çarpım işlemi gerektirir. Ayrıca, ikinci listedeki bir elemanın ilk listede olup olmadığını kontrol etmek yaklaşık $\log_2(n)$ adım alır, dolayısıyla toplamda $n\log_2(n)$ karşılaştırma yapılır. Bu nedenle, toplam hesaplama süresi yaklaşık olarak

$$4n\log_2(N) + n\log_2(n) = n\log_2(N^4 n)$$

adımdır. $n \approx 3\sqrt{N}$ alırsak, hesaplama Süresi $\approx 13.5 \cdot \sqrt{N} \cdot \log_2(1.3 \cdot N) = \mathcal{O}(\sqrt{N}\log_2(N))$
Örnek: $2^x = 390 \mod 659$ denklemini çözmeye çalışalım. 2 sayısının modulo 659'daki mertebesi 658'dir, bu nedenle bir ilkel köktür. Bu örnekte $g = 2$ ve $h = 390$'dır. Rastgele t üsleri seçip g^t ve $h \cdot g^t$ değerlerini bir eşleşme bulana kadar hesaplıyoruz. Sonuçlar aşağıda figure 3'te gösterilmiştir. Görüyoruz ki:

$$2^{83} = 390 \cdot 2^{564} \mod 659$$

Böylece, uzunluğu 18 olan iki liste kullanarak mod 659'de bir ayrık logaritma problemini çözdük. (Uzunluğu 18 olan listelerle bir eşleşme bulma şansımız %39'du, bu yüzden biraz şanslıydık.) O halde çözüm:

$$2^{83} \cdot 2^{-564} = 2^{-481} = 2^{177} = 390 \mod 659$$

t	g^t	$h \cdot g^t$
564	410	**422**
469	357	181
276	593	620
601	416	126
9	512	3
350	445	233

t	g^t	$h \cdot g^t$
53	10	605
332	651	175
178	121	401
477	450	206
503	116	428
198	426	72

t	g^t	$h \cdot g^t$
513	164	37
71	597	203
314	554	567
581	47	537
371	334	437
83	**422**	489

Figure 3: $2^x \equiv 390 \pmod{659}$ denkleminin Collision Algoritması ile çözümü

Not:
Zaten BabyStep - GiantStep algoritması $\mathcal{O}(\sqrt{N})$ hamlede ayrık logatirma problemini çözüyordu. Neden probabilistic collision algoritmasına ihtiyacımız var ki diye sorulacak olursa aşağıda konuşacağımız Pollard ρ algoritmasına temel atmak için diye cevap verilebilir.

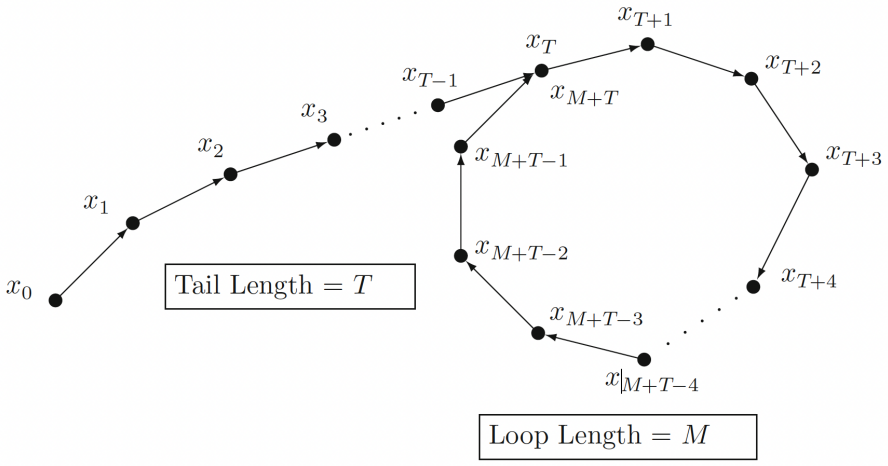

Figure 4: Pollard ρ metodu

6 Pollard's ρ Algorithm

Pollard ρ algoritması yine ayrık logaritmayı $\mathcal{O}(\sqrt{N})$ hamlede çözer ancak bunu güzel yapan şey hafızada sadece aynı anda çok az sayı tutmamız gerektiğidir. BabyStep - Giantstep ve yukarıdaki collision algotrimaları hafızada $\mathcal{O}(\sqrt{N})$ sayı tutarlar ve sayı büyük olduğu zaman bu çok fazla yer kaplayabilir. Pollard ρ algoritması buna güzel bir çözüm sunar. Şimdi biraz algoritmanın ne yapmaya çalıştığına bakalım.
S sonlu bir küme olsun ve

$$f : S \longrightarrow S$$

f fonksiyonu S'nin elemanlarını iyi bir şekilde karıştıran bir fonksiyon olsun. Bir $x \in S$ elemanı ile başlayıp f'yi tekrar tekrar uygulayarak bir dizi oluşturduğumuzu varsayalım:

$$x_0 = x, \quad x_1 = f(x_0), \quad x_2 = f(x_1), \quad x_3 = f(x_2), \quad x_4 = f(x_3), \ldots$$

Başka bir deyişle,

$$x_i = \underbrace{(f \circ f \circ f \circ \cdots \circ f)}_{f\text{'nin } i \text{ kez uygulanması}}(x).$$

$$x_0, x_1, x_2, x_3, x_4, \ldots$$

dizisine f dönüşümü altında x'in (ileri) yörüngesi denir ve $O_f^+(x)$ ile gösterilir. S kümesi sonlu olduğundan, eninde sonunda $O_f^+(x)$ yörüngesinde çakışan bir eleman olacaktır. Yörüngeyi figure 4'de gösterildiği gibi resmedebiliriz. Algoritmaya neden ρ dendiği de şekle bakılarak anlaşılabilir. Bir süre boyunca $x_0, x_1, x_2, x_3, \ldots$ noktaları tekrarlanmadan bir yol izler, ta ki bir eleman tekrarlanınca bir döngü oluşana kadar. Daha sonra döngü etrafında hareket etmeye devam ederler. Şekilde gösterildiği gibi, döngüye ulaşmadan önceki "kuyruk" kısmındaki eleman sayısını T ile ve döngüdeki eleman sayısını M ile gösterelim. Matematiksel olarak, T ve M aşağıdaki gibi tanımlanır:

$$T = \begin{pmatrix} x_{T-1}\text{'in } O_f^+(x)\text{'de yalnızca bir kez} \\ \text{görüldüğü en büyük tamsayı} \end{pmatrix} \quad M = \begin{pmatrix} x_{T+M} = x_T \text{ koşulunu sağlayan} \\ \text{en küçük tamsayı} \end{pmatrix}$$

Pollard'ın zekice fikri, tüm değerleri saklamadan $O(\sqrt{N})$ adımda bir çakışmayı tespit etmenin mümkün olmasıdır. Bunu başarmanın çeşitli yolları vardır. Biz bunlardan birini açıklayacağız.

Fikir, yalnızca x_i dizisini değil, aynı zamanda

$$y_0 = x_0 \quad \text{ve} \quad y_{i+1} = f(f(y_i)) \quad i = 0, 1, 2, 3, \ldots \text{ için}$$

şeklinde tanımlanan ikinci bir y_i dizisini de hesaplamaktır. Başka bir deyişle, x_i dizisinin bir sonraki elemanını üretmek için f'yi her uyguladığımızda, y_i dizisinin bir sonraki elemanını üretmek için f'yi iki kez uygularız. Açıktır ki

$$y_i = x_{2i}.$$

$x_{2i} = x_i$ koşulunu sağlayan bir i indeksini bulmak ne kadar sürer? Genel olarak, $j > i$ için

$$x_j = x_i \quad \text{ancak ve ancak} \quad i \geq T \quad \text{ve} \quad j \equiv i (\text{mod } M).$$

Bu durum, figure 4'te ρ şeklinden açıkça görülebilir, çünkü $x_j = x_i$ yalnızca x_T'yi geçtiğimizde, yani $i \geq T$ olduğunda ve x_j, x_i'yi geçerek döngüyü tam sayılı sayıda tamamladığında, yani $j - i$, M'nin bir katı olduğunda gerçekleşir. Dolayısıyla, $x_{2i} = x_i$ ancak ve ancak $i \geq T$ ve $2i \equiv i (\text{mod } M)$ olduğunda doğrudur. İkinci koşul $M|i$ ile eşdeğerdir, bu nedenle $x_{2i} = x_i$ yalnızca i, T'den büyük olan ilk M katına eşit olduğunda sağlanır. $T, T+1, \ldots, T+M-1$ sayılarından biri M'ye bölünebildiğinden, bu

$$x_{2i} = x_i \quad 1 \leq i < T + M \text{ olacak şekilde bir } i \text{ için}$$

olduğunu kanıtlar.

Teorem(Pollard'ın ρ Yöntemi)
S, N eleman içeren sonlu bir küme olsun, $f : S \to S$ bir fonksiyon olsun ve $x \in S$ başlangıç noktası olsun.
(a) x'in ileri yörüngesi $O_f^+(x) = \{x_0, x_1, x_2, \ldots\}$'nin uzunluğu T olan bir kuyruk ve uzunluğu M olan bir döngü içerdiğini varsayalım (figure 4'te gösterildiği gibi). Bu durumda,

$$x_{2i} = x_i \quad 1 \leq i < T + M \text{ koşulunu sağlayan bir } i \text{ için.}$$

(b) Eğer f fonskiyonu yeterince rastgele davranıyorsa, $T + M$'nin beklenen değeri yaklaşık olarak

$$E(T + M) \approx 1.2533 \cdot \sqrt{N}$$

şeklindedir. Dolayısıyla, N büyük olduğunda, çakışmayı $O(\sqrt{N})$ adımda bulmamız olasıdır. Burada bir "adım", f fonksiyonunun bir kez değerlendirilmesine karşılık gelir.
İspat: a adımını yukarıda ispatlamıştık, b adımı bol düzeyde olasılık teorisi ve analiz içerdiği için burada ifade etmeyeceğiz. İspata bakmak isteyenler [1] dip notundaki ilgili yere bakabilir.
Ayrık Logaritma Porblemini Pollard ρ ile Çözmek
$g^t = h$ denklemini $\mathbb{Z}_p^*$'de çözmeye çalışıyoruz. g'nin modulo p bir ilkel kök olduğunda her zaman çözüm olduğunu biliyoruz. O yüzden g'yi ilkel kök alalım. Temel fikir, bilinen üsler i, j, k, ℓ için $g^i h^j$ ve $g^k h^\ell$ arasında bir çakışma bulmaktır. Bu durumda $g^{i-k} = h^{\ell-j}$ olur ve $\mathbb{Z}_p^*$ içinde kök almak, h'yi g'nin bir kuvveti olarak ifade etme problemini büyük ölçüde çözer. Buradaki asıl zorluk, $\mathbb{Z}_p^*$'nin elemanlarını karıştıracak kadar karmaşık, ancak yörüngelerini takip edebilecek kadar basit bir $f : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ fonksiyonu bulmaktır. Pollard, aşağıdaki fonksiyonu kullanmayı önerir:

$$f(x) = \begin{cases} gx & \text{eğer } 0 \leq x < p/3, \\ x^2 & \text{eğer } p/3 \leq x < 2p/3, \\ hx & \text{eğer } 2p/3 \leq x < p. \end{cases}$$

[1] Jeffrey Hoffstein, Jill Pipher, ve Joseph H. Silverman, *An Introduction to Mathematical Cryptography*, Springer, 2014, Teorem 5.48, s. [256].

Not: $f(x)$'in değerini belirlemek için kullanılmadan önce x'in $0 \leq x < p$ aralığına modulo p indirgenmesi gerekir. Başlangıç noktası $x_0 = 1$ olarak ile verilen f fonksiyonunu tekrar tekrar uyguladığımızda ne olduğunu düşünelim. Her adımda ya g ile çarparız, ya h ile çarparız, ya da önceki değerin karesini alırız. Dolayısıyla, her adımın sonunda g'nin bir kuvveti ile h'nin bir kuvvetinin çarpımını elde ederiz. Örneğin, i adım sonra:

$$x_i = \underbrace{(f \circ f \circ f \circ \cdots \circ f)}_{f\text{'nin } i \text{ kez uygulanması}}(1) = g^{\alpha_i} \cdot h^{\beta_i}.$$

α_i ve β_i değerlerini önceden tahmin edemeyiz, ancak f'in tanımını kullanarak x_i'leri hesaplarken bu değerleri de eş zamanlı olarak hesaplayabiliriz. Açıkça $\alpha_0 = \beta_0 = 0$'dır ve sonraki değerler şu şekilde hesaplanır:

$$\alpha_{i+1} = \begin{cases} \alpha_i + 1 & \text{eğer } 0 \leq x < p/3, \\ 2\alpha_i & \text{eğer } p/3 \leq x < 2p/3, \\ \alpha_i & \text{eğer } 2p/3 \leq x < p, \end{cases}$$

$$\beta_{i+1} = \begin{cases} \beta_i & \text{eğer } 0 \leq x < p/3, \\ 2\beta_i & \text{eğer } p/3 \leq x < 2p/3, \\ \beta_i + 1 & \text{eğer } 2p/3 \leq x < p. \end{cases}$$

α_i ve β_i'yi hesaplarken, bu değerleri modulo $p-1$ olarak takip etmek yeterlidir, çünkü $g^{p-1} = 1$ ve $h^{p-1} = 1$'dir. Bu önemlidir, aksi takdirde α_i ve β_i değerleri aşırı büyük hale gelir. Benzer şekilde, aşağıdaki diziyi hesaplarız:

$$y_0 = 1 \quad \text{ve} \quad y_{i+1} = f(f(y_i)).$$

Bu durumda,

$$y_i = x_{2i} = g^{\gamma_i} \cdot h^{\delta_i},$$

burada γ_i ve δ_i üsleri, α_i ve β_i için kullanılan yinelemelerin iki kez tekrarlanmasıyla hesaplanabilir. Yukarıdaki prosedürü uygulayarak, sonunda x ve y dizilerinde bir çakışma buluruz, diyelim ki $y_i = x_i$. Bu şu anlama gelir:

$$g^{\alpha_i} \cdot h^{\beta_i} = g^{\gamma_i} \cdot h^{\delta_i}.$$

Eğer

$$u \equiv \alpha_i - \gamma_i \pmod{p-1} \quad \text{ve} \quad v \equiv \delta_i - \beta_i \pmod{p-1},$$

olarak tanımlarsak, $\mathbb{Z}_p^*$'de $g^u = h^v$ elde ederiz. Başka bir deyişle,

$$v \cdot \log_g(h) \equiv u \pmod{p-1}$$

Eğer $EBOB(v, p-1) = 1$ ise, yukarıdaki denklemin her iki tarafını v'nin modulo $p-1$'deki tersi ile çarparak ayrık logaritma problemini çözebiliriz. Daha genel olarak, eğer $d = EBOB(v, p-1) \geq 2$ ise, genişletilmiş Öklid algoritmasını kullanarak

$$s \cdot v \equiv d \pmod{p-1}$$

koşulunu sağlayan bir s tamsayısı buluruz. yukarıdaki denklemin her iki tarafını s ile çarparsak,

$$d \cdot \log_g(h) \equiv w \pmod{p-1}$$

elde ederiz, burada $w \equiv s \cdot u \pmod{p-1}$ şeklindedir. Bu denklemde $\log_g(h)$ hariç tüm değerler bilinmektedir. d'nin $p-1$'i bölmesi, d'nin w'yi de bölmesini gerektireceğinden, w/d yukarıdaki denklemin bir çözümüdür, ancak başka çözümler de vardır. yukarıdaki denklemin tüm çözüm kümesi, w/d ile başlayıp $(p-1)/d$'nin katlarını ekleyerek elde edilir:

$$\log_g(h) \in \left\{ \frac{w}{d} + k \cdot \frac{p-1}{d} : k = 0, 1, 2, \ldots, d-1 \right\}.$$

Pratikte d genellikle oldukça küçük olacaktır, bu nedenle $\log_g(h)$ için d tane olasılığı teker teker kontrol edilerek doğru değer bulunabilir. Pratikte yukarıdaki f fonksiyonundan daha karmaşık ama daha verimli fonksiyonlar vardır. Bazı olası durumlar için yukarıdaki f fonksiyonunun verimsiz olduğu gösterilmiştir ama çoğu zaman için işe yarar şekilde sonuçlar verir.

Örnek:

Pollard'ın ρ yöntemini,

$$19^t \equiv 24717 \pmod{48611}$$

ayrık logaritma problemini çözerek örnekleyeceğiz. İlk adım, bir eşleşme $y_i = x_i$ bulana kadar x ve y dizilerini hesaplamak, aynı zamanda $\alpha, \beta, \gamma, \delta$ üs dizilerini de hesaplamaktır. Bu sürecin başlangıç aşamaları ve bir çakışma bulunmadan önceki son birkaç adım figure 5'da verilmiştir.

i	x_i	$y_i = x_{2i}$	α_i	β_i	γ_i	δ_i
0	1	1	0	0	0	0
1	19	361	1	0	2	0
2	361	33099	2	0	4	0
3	6859	13523	3	0	4	2
4	33099	20703	4	0	6	2
5	33464	14974	4	1	13	4
6	13523	18931	4	2	14	5
7	13882	30726	5	2	56	20
8	20703	1000	6	2	113	40
9	11022	14714	12	4	228	80
			⋮			
542	21034	46993	13669	2519	27258	30257
543	20445	37138	27338	5038	27259	30258
544	40647	33210	6066	10076	5908	11908
545	28362	21034	6066	10077	5909	11909
546	36827	40647	12132	20154	23636	47636
547	11984	36827	12132	20155	47272	46664
548	33252	33252	12133	20155	47273	46665

Figure 5: $19^t \equiv 24717 \pmod{48611}$ denkleminin Pollard ρ yöntemi ile çözümü

Tablodan görüldüğü üzere, $\mathbb{F}^*_{48611}$ cisminde $x_{1096} = x_{548} = 33252$ olmaktadır. İlgili üs

değerleri şöyledir:

$$\alpha_{548} = 12133, \quad \beta_{548} = 20155, \quad \gamma_{548} = 47273, \quad \delta_{548} = 46665,$$

dolayısıyla şu eşitliği biliyoruz:

$$19^{12133} \cdot 24717^{20155} = 19^{47273} \cdot 24717^{46665} \mod 48611$$

19'un kuvvetlerini bir tarafa, 24717'nin kuvvetlerini diğer tarafa alarak $19^{-35140} = 24717^{26510}$ elde ederiz ve 19'un üssüne $48610 = p - 1$ ekleyerek:

$$19^{13470} = 24717^{26510} \mod 48611$$

Daha sonra şu gözlemi yaparız:

$$\gcd(26510, 48610) = 10 \quad \text{ve} \quad 970 \cdot 26510 \equiv 10 \pmod{48610}.$$

her iki tarafının 970. kuvvetini alarak:

$$19^{13470 \cdot 970} = 19^{13065900} = 19^{38420} = 24717^{10} \mod 48611$$

Böylece:

$$10 \cdot \log_{19}(24717) \equiv 38420 \pmod{48610},$$

bu da şu anlama gelir:

$$\log_{19}(24717) \equiv 3842 \pmod{4861}.$$

Ayrık logaritmanın olası değerleri, 3842'ye 4861'in katları eklenerek bulunur, dolayısıyla $\log_{19}(24717)$ şu kümedeki sayılardan biridir:

$$\{3842, 8703, 13564, 18425, 23286, 28147, 33008, 37869, 42730, 47591\}.$$

Çözümü tamamlamak için, 19'un bu 10 değerin her birine karşılık gelen kuvvetlerini 24717'ye eşit olan bulana kadar hesaplarız:

$$19^{3842} = 16580, \quad 19^{8703} = 29850, \quad 19^{13564} = 23894, \quad 19^{18425} = 20794,$$

$$19^{23286} = 10170, \quad 19^{28147} = 32031, \quad 19^{33008} = 18761, \quad 19^{37869} = \boxed{24717}.$$

Bu bize $\log_{19}(24717) = 37869$ çözümünü verir.

7 Index Calculus

Sadece $\mathbb{Z}_p^*$ grubunda çalışan subeksponansiyel zamanda çözülen bir algoritma sunar. Şu ana kadar yaptığımız tüm algoritmalar eksponansiyel zamanda DLP'yi çözüyordu. Bu algoritma bize kayda değer bir hız sağlar. Şimdi algoritmaya geçmeden önce kullanacağımız bir tanıma bakalım.

Bir n tam sayısı, tüm asal çarpanları B'den küçük veya eşitse, B-düzgün (B-Smooth) olarak adlandırılır.

Örnek: İlk birkaç 5-düzgün sayı ve 5-düzgün olmayan sayılar aşağıda verilmiştir.

5-düzgün sayılar:

2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32, 36, ...

5-düzgün olmayan sayılar:

7, 11, 13, 14, 17, 19, 21, 22, 23, 26, 28, 29, 31, 33, 34, 35, 37, ...
İndeks hesabının arkasındaki fikir oldukça basittir. Ayrık logaritma problemini çözmek istiyoruz:

$$g^x \equiv h \pmod{p},$$

burada p asal sayısı ve g ile h tamsayıları verilmiştir. Basitlik için, g'nin mod p için bir ilkel kök olduğunu varsayacağız, böylece üsleri $\mathbb{F}_p^*$'ın tamamını verir. DLP doğrudan çözmek yerine, bir B değeri seçer ve aşağıdaki ayrık logaritma problemini çözeriz:

$$g^x \equiv \ell \pmod{p} \qquad \text{tüm } \ell \leq B \text{ asalları için.}$$

Başka bir deyişle, $\ell \leq B$ koşulunu sağlayan her asal sayı için $\log_g(\ell)$ ayrık logaritmasını hesaplarız. Bunu yaptıktan sonra,

$$h \cdot g^{-k} \pmod{p} \qquad k = 1, 2, \ldots \text{ için}$$

değerlerine bakarız ve $h \cdot g^{-k} \pmod{p}$'nin B-düzgün olduğu bir k değeri bulana kadar devam ederiz. Bu k değeri için,

$$h \cdot g^{-k} \equiv \prod_{\ell \leq B} \ell^{e_\ell} \pmod{p}$$

eşitliği belirli e_ℓ üsleriyle sağlanır. eşitliğini ayrık logaritmalar cinsinden yeniden yazarsak,

$$\log_g(h) \equiv k + \sum_{\ell \leq B} e_\ell \cdot \log_g(\ell) \pmod{p-1},$$

elde ederiz. Ancak zaten tüm $\ell \leq B$ asalları için $\log_g(\ell)$ değerlerini hesapladığımızı varsayıyoruz. Dolayısıyla denklemi bize $\log_g(h)$ değerini verir. Geriye, küçük asallar için $\log_g(\ell)$ değerlerini nasıl bulacağımızı açıklamak kalıyor. Fikir yine basittir. Rastgele seçilmiş i üsleri için,

$$g_i \equiv g^i \pmod{p} \qquad \text{ve } 0 < g_i < p$$

değerini hesaplarız. Eğer g_i B-düzgün değilse, onu kullanmayız. Ancak g_i B-düzgün ise, onu şu şekilde çarpanlarına ayırabiliriz:

$$g_i = \prod_{\ell \leq B} \ell^{u_\ell(i)}.$$

Ayrık logaritmalar açısından, bu bize aşağıdaki ilişkiyi verir:

$$i \equiv \log_g(g_i) \equiv \sum_{\ell \leq B} u_\ell(i) \cdot \log_g(\ell) \pmod{p-1}.$$

Yukarıda tek bilinmeyenler $\log_g(\ell)$ ayrık logaritma değerleridir. Bu yüzden $\pi(B)$'den daha fazla denklem bulabilirsek, $\log_g(\ell)$ "değişkenleri" için liner cebir kullanarak çözebiliriz.
Örnek:

$$37^x \equiv 211 \pmod{18443}.$$

problemini index hesabı ile çözmeye çalışalım. $g = 37$'nin $p = 18443$ modülüne göre bir ilkel köktür. $B = 5$ seçelim, bu nedenle çarpan tabanımız $\{2, 3, 5\}$ asal sayılarından oluşur. İşleme, $g = 37$'nin rastgele üslerini mod 18443 alarak başlıyoruz ve B-düzgün olanları seçiyoruz. Birkaç deneme sonucunda dört denklem elde ediyoruz:

$$g^{12708} \equiv 2^3 \cdot 3^4 \cdot 5 \pmod{18443}, \quad g^{11311} \equiv 2^3 \cdot 5^2 \pmod{18443},$$

$$g^{15400} \equiv 2^3 \cdot 3^3 \cdot 5 \pmod{18443}, \quad g^{2731} \equiv 2^3 \cdot 3 \cdot 5^4 \pmod{18443}$$

Bu denklemler, sırasıyla 2, 3 ve 5 sayılarının g tabanına göre ayrık logaritmaları için doğrusal ilişkiler verir. Örneğin, ilk denklem şunu söyler:

$$12708 = 3 \cdot \log_g(2) + 4 \cdot \log_g(3) + \log_g(5).$$

$$x_2 = \log_g(2), \quad x_3 = \log_g(3), \quad \text{ve} \quad x_5 = \log_g(5).$$

olsun. Böylece, dört denklik aşağıdaki dört doğrusal ilişkiye dönüşür:

$$\begin{array}{llll} 12708 & = 3x_2 & +4x_3 & +x_5 \pmod{18442}, \\ 11311 & = 3x_2 & + & +2x_5 \pmod{18442}, \\ 15400 & = 3x_2 & +3x_3 & +x_5 \pmod{18442}, \\ 2731 & = 3x_2 & +x_3 & +4x_5 \pmod{18442}. \end{array}$$

$$p - 1 = 18442 = 2 \cdot 9221,$$

9221 sayısı asaldır, bu nedenle doğrusal denklem sistemini mod 2 ve mod 9221 olarak çözmemiz gerekir. Bu, Gaussian Elimination yöntemiyle kolayca yapılabilir. Çözümler şöyledir:

$$(x_2, x_3, x_5) \equiv (1, 0, 1) \pmod{2},$$

$$(x_2, x_3, x_5) \equiv (5733, 6529, 6277) \pmod{9221}.$$

Bu çözümleri birleştirerek şunu elde ederiz:

$$(x_2, x_3, x_5) \equiv (5733, 15750, 6277) \pmod{18442}.$$

Çözümleri, aşağıdaki hesaplamaları yaparak kontrol edelim:

$$37^{5733} \equiv 2 \pmod{18443}, \quad 37^{15750} \equiv 3 \pmod{18443}, \quad 37^{6277} \equiv 5 \pmod{18443}.$$

Asıl amacımızın ayrık logaritma problemini çözmekti:

$$37^x \equiv 211 \pmod{18443}.$$

$211 \cdot 37^{-k} \pmod{18443}$ ifadesinin değerini rastgele k değerleri için hesaplayarak B-düzgün olan bir değer bulana kadar devam ederiz. Birkaç denemeden sonra şunu buluruz:

$$211 \cdot 37^{-9549} \equiv 2^5 \cdot 3^2 \cdot 5^2 \pmod{18443}.$$

Yukarıdaki 2, 3 ve 5'in ayrık logaritma değerlerini kullanarak şunu elde ederiz:

$$\log_g(211) = 9549 + 5\log_g(2) + 2\log_g(3) + 2\log_g(5)$$

$$= 9549 + 5 \cdot 5733 + 2 \cdot 15750 + 2 \cdot 6277 \equiv 8500 \pmod{18442}.$$

Son olarak, $\log_g(211) = 8500$ cevabımızı şu hesaplamayla kontrol ederiz:

$$37^{8500} \equiv 211 \pmod{18443}. \quad \checkmark$$

Teorem: $L(X) = e^{\sqrt{(\ln X)(\ln \ln X)}}$ olsun, N büyük bir tam sayı ve $B = L(N)^{1/\sqrt{2}}$ olsun. B-düzgün olan $\pi(B)$ sayıyı bulmak için modülo N'de yaklaşık olarak $L(N)^{\sqrt{2}}$ rastgele sayı kontrol etmek gerekir.

İndeks hesabı yönteminin çalışma süresini kabaca şu şekilde tahmin edebiliriz: B'den küçük asallardan oluşan bir çarpan tabanı kullanarak, yaklaşık $\pi(B)$ tane $g^i \pmod p$ formundaki ve B-düzgün olan sayı bulmamız gerekir. Yukarıdaki teorem, $B = L(p)^{1/\sqrt{2}}$ seçmemizi önerir ve bu durumda yaklaşık $L(p)^{\sqrt{2}}$ tane i değerini kontrol etmemiz gerekecektir. İndeks hesabı yöntemi $\mathbb{F}_p^*$'da ayrık logaritma problemini çözmek için subeksponansiyel bir algoritmadır. Bu durum, eliptik eğri gruplarındaki ayrık logaritma problemiyle belirgin bir şekilde ayrılır. Şu anda, eliptik eğri gruplarındaki genel ayrık logaritma problemini çözmek için bilinen en iyi algoritmalar tamamen eksponansiyeldir.

Victor Shoup, grubun mertebesini bölen en büyük asal p olmak üzere, herhangi bir sonlu grupta DLP'yi $\mathcal{O}(\sqrt{p})$ adımdan daha az adımda çözecek genel bir algoritmanın olamayacağını göstermiştir. Yani tüm gruplarda $\mathcal{O}(\sqrt{p})$'dan daha hızlı çalışan bir algoritma bulunamaz. Ancak bazı belli gruplarda (yukarıdaki index hesabı gibi) bu algoritma hızlandırılabilir. Bu durum ilginçtir, zira bir grupta üst almak her ne kadar kolay ve temel bir şey olsa dahi bunun nasıl bir biçimde gerçekleştiğinin örüntüsünü bilemiyoruz.