# لگاریتم گسسته

احمدرضا عبدلی

در این مقاله ابتدا به معرفی مفاهیمی از جبر مجرد پرداخته که در معرفی مفهوم لگاریتم گسسته نیاز بوده سپس لگاریتم گسسته را تعریف میکنیم. در ادامه دو روش برای حل مساله لگاریتم گسسته بیان کرده و به توضیح آنها میپردازیم. نظر به اینکه لگاریتم گسسته نقش به سزایی در حل روش رمزنگاری RSA دارد جز مسایل داغ روز به حساب آمده و همواره اهمیت خاصی برای روش‌های هر چه بهتر برای حل آن مدنظر قرار گرفته است.

## لگاریتم گسسته و مفاهیم جبری

در این بخش ابتدا به تعریف لگاریتم گسسته و روش هایی برای محاسبه‌ی آن می‌پردازیم.

**تعریف ۱.** *فرض کنید $(G,.)$ گروهی ضربی و متناهی باشد. گروه دوری تولید شده توسط عنصر $a$ از گروه $G$ را $<a>$ می‌نامیم و آنرا به شکل زیر تعریف می‌کنیم:*

$$<a>=\{a^i \mid a\in G, i\in\mathbb{N}\}$$

*واضح است که متناهی بودن گروه $G$، متناهی بودن $<a>$ را نتیجه می‌دهد. حال اگر $|<a>|=n$، در این‌صورت:*

$$<a>=\{e,a,a^۲,\dots,a^{n-۱}\}$$

*زیرگروه بودن $<a>$ به سادگی از تعریف بدست می‌آید.*

از تعریف نتیجه می‌شود اگر $g\in<a>$ در این‌صورت $j\in\{۰,\dots,n-۱\}$ وجود دارد به‌طوری که $g=a^j$.

**تعریف ۲.** *منظور از لگاریتم گسسته عنصر $g$، از گروه ضربی $(G,.)$، از مرتبه‌ی $n$ در مبنای $b$ عبارت است از عدد $۰\le i\le n-۱$، به طوری که $b^i=g$، یا به صورت معادل $\log_b^g=i$ می‌باشد. مشخصه لگاریتم گسسته در آن است که محاسبه آن سخت ولی چک کردن جواب به غایت آسان است.*

## الگوریتم‌های محاسبه لگاریتم گسسته

حال به بررسی الگوریتم‌هایی برای محاسبه‌ی لگاریتم گسسته خواهیم پرداخت. فرض کنید $(G,.)$ گروه ضربی و $<\alpha>$ زیرگروهی دوری از آن باشد به طوری‌که $\beta\in<\alpha>$ باشد. هدف بر آن است که $i$ را طوری پیدا کنیم که $\beta=\alpha^i$ و $i$ یکتا باشد.اولین الگوریتم و ساده‌ترین روش، چک کردن یک‌به‌یک عنصرها خواهد بود. زمان مورد نیاز برای هر ضرب در گروه $G$، از $O(۱)$ می‌باشد. اگر $|<\alpha>|=n$ آنگاه محاسبه‌ی لگاریتم گسسته از $O(n)$ خواهد بود.

به این صورت که با داشتن فضایی از $O(۱)$ هر دفعه که $\alpha^i$ محاسبه شد، آن را ذخیره کرده و با مقدار $\beta$ چک می‌کنیم و در صورت یکی نبودن، مقدار $\alpha^i$ را در $\alpha$ ضرب کرده و دوباره مقایسه می‌کنیم.

چون حداکثر $n$ بار ضرب نیاز است، زمان محاسبه از $O(n)$ خواهد بود و به $O(1)$ فضا نیاز داریم. راه دیگر این است که تمام مقادیر را قبلاً محاسبه و به صورت جفت مرتب منظم کنیم و سپس با جستجویی مانند جستجوی دودویی به مقایسه کردن بپردازیم.
این راه $O(n)$ برای محاسبه زمان نیاز دارد و در صورت استفاده از روش منظم کردن بهینه مانند quick sort به $O(n\log n)$ زمان نیز برای مرتب کردن نیاز دارد. فضایی از $O(n)$ برای ذخیره‌ی مقادیر و زمانی از $O(n)$ برای جستجو در این مقادیر، و در پایان با فضایی از $O(n)$ و زمانی از $O(n\log n)$ امکان‌پذیر خواهد بود.
الگوریتم‌های دیگر برای اینکار الگوریتم شنک[1] و الگوریتم پولارد رو[2] هستند که به بررسی کوتاهی در باب آن می‌پردازیم.

## الگوریتم شنک

فرض کنید $\beta$ عنصری است که می‌خواهیم لگاریتم گسسته‌ی آن را در مبنای $\alpha$ حساب کنیم. برای $j$های کوچکتر از $\sqrt{n}$ و بزرگتر یا مساوی صفر، $\alpha^{mj}$ را حساب کرده که در آن $m=[\sqrt{n}]+1$ و سپس آن‌ها را در زوج مرتب‌های $(j,\alpha^{mj})$ بر حسب مولفه‌ی دوم مرتب می‌کنیم ولیست $L_1$ به وجود می‌آید. سپس $\beta\alpha^{-j}$ را نیز حساب کرده، به همان شکل مرتب می‌کنیم و در $L_2$ می‌گذاریم و بین این دو لیست زوجی را پیدا می‌کنیم که مولفه‌ی دوم‌شان یکی باشد.
به این‌صورت که اگر $(i,y)\in L_2$ و $(j,y)\in L_1$ باشند آنگاه

$$\alpha^{mj}=\beta\alpha^{-i}\Rightarrow\alpha^{mj+i}=\beta$$

و اگر $\beta\in<\alpha>$ در نتیجه $0\le\log_\alpha\beta\le n-1$. حال $\log\frac{\beta}{\frac{\alpha}{m}}$ را محاسبه می‌کنیم و $\log_\alpha\beta=mj+i$.
به وضوح جستجو در قسمت آخر تمام می‌شود چون اگر نشود $\beta\notin<\alpha>$.

SHANKS$(G,n,\alpha,\beta)$
$m\leftarrow\lceil\sqrt{n}\rceil$
for $j\leftarrow 0$ to $m-1$
do compute $\alpha^{mj}$
Sort the $m$ ordered pairs $(j,\alpha^{mj})$ with respect to their second coordinates, obtaining a list $L_1$
for $i\leftarrow 0$ to $m-1$
do compute $\beta\alpha^{-i}$
Sort the $m$ oredered pairs $(i,\beta\alpha^{-1})$ with respect to their second coordinates, obtaininig in a list $L_2$
Find a pair $(j,y)\in L_1$ and a pair $(i,y)\in L_2$ (i.e., find two pairs having identical second coordinates)
$\log_\alpha\beta\leftarrow(mj+i)\bmod n$

حال به آنالیز مقدار زمان و فضای مورد نیاز می‌پردازیم. ضرب کردن توان‌های $\alpha$ از ۱ تا $\sqrt{n}$ از $O(m)$ بوده و مانند الگوریتم ابتدایی $O(m)$ حافظه نیاز دارد. به وضوح در حساب کردن $L_2$ نیز $O(m)$ فضا و زمان نیاز است، و منظم کردن از $O(m\log m)$ زمان نیاز دارد. همچنین مقایسه کردن لیست‌ها هم $O(m)$ زمان نیاز دارد. بنابراین الگوریتم شنک از لحاظ فضا و زمان، بسیار موثرتر از الگوریتم قبلی می‌باشد.

[1] Shank
[2] Pollord Rho

## الگوریتم پولارد رو

این الگوریتم مشابه الگوریتم قبلی است با این تفاوت که در اینجا اعضای $G$، به سه قسمت با کاردینال‌های تقریبا مساوی افراز می‌شود.

$$G = S_۱ \cup S_۲ \cup S_۳, | S_۱ | \cong | S_۲ | \cong | S_۳ |$$

حال تابعی از ضرب دکارتی $\mathbb{Z}_n$ و $\mathbb{Z}_n$ و $< \alpha >$ به خودش تعریف کرده و سعی در ساختن دنباله‌ای می‌کنیم که پس از پیدا کردن عناصر تکراری $\log_\alpha \beta$ بدست خواهد آمد.

$$f :< \alpha > \times \mathbb{Z}_n \times \mathbb{Z}_n \rightarrow < \alpha > \times \mathbb{Z}_n \times \mathbb{Z}_n$$

حال تابع زیر را در نظر بگیرید:

$$f(x, a, b) = \begin{cases} (\beta x, a, b + ۱) & x \in S_۱ \\ (x^۲, ۲a, ۲b) & x \in S_۲ \\ (\alpha x, a + ۱, b) & x \in S_۳ \end{cases}$$

شرط زیر را بر ورودی $(x, a, b)$ در نظر بگیرید:

$$x = \alpha^a \beta^b \quad (۱)$$

با مقدار اولیه‌ی $(۱, ۰, ۰)$، اگر ورودی خاصیت (۱) را داشته باشد خروجی $f$ هم این خاصیت را دارد. با شروع از مقدار اولیه‌ی $(۱, ۰, ۰)$ به دنباله‌ای از عناصر $< \alpha > \times \mathbb{Z} \times \mathbb{Z}$ می‌رسیم که به روش بازگشتی با $(x_{i+۱}, a_{i+۱}, b_{i+۱}) = f(x_i, a_i, b_i)$ تعریف می‌شوند.

عناصر $(x_i, a_i, b_i)$ و $(x_{۲i}, a_{۲i}, b_{۲i})$ بررسی می‌شوند تا زمانی که مولفه‌ی اول‌شان برابر شود.

اگر تساوی برقرار شود یعنی $\alpha^{a_i}\beta^{b_i} = x_i = x_{۲i} = \alpha^{a_{۲i}}\beta^{b_{۲i}}$ با جای‌گذاری $\alpha^c = \beta$ $(c = \log_\alpha^\beta)$ به تساوی $\alpha^{a_i + cb_i} = \alpha^{a_{۲i} + cb_{۲i}}$ می‌رسیم و در نتیجه $a_i + cb_i \equiv a_{۲i} + cb_{۲i}$ به پیمانه $n$.

$$\Rightarrow (a_{۲i} - a_i) \equiv -c(b_{۲i} - b_i) \mod n$$

و اگر

$$(b_{۲i} - b_i, n) = ۱$$

آنگاه

$$c = (a_i - a_{۲i})(b_{۲i} - b_i)^{-۱}$$

و در حالتی که $(b_{۲i} - b_i, n) = d$ و $d > ۱$ به $d$ جواب برای معادله می‌رسیم که در صورت بزرگ بودن نمی‌توان همه ی اینها را محاسبه نمود و در غیر این‌صورت لگاریتم گسسته محاسبه می‌شود. مادامی که $| < \alpha > | = n$، با محاسبات می‌توان نشان داد که در شرایط ایده آل شامل رندوم بودن تابع $f$ و ... می‌توان در $O(\sqrt{n})$ این الگوریتم را انجام داد.

POLLARD RHO DISCRETE LOG ALGORITHM $(G, N, \alpha, \beta)$
  procedure $f(x, a, b)$
   if $x \in S_1$
    then $f \leftarrow (\beta.x, a, (b + 1) \bmod n)$
    else if $x \in S_2$
    then $f \leftarrow (x^2, 2a \bmod n, 2b \bmod n)$
    else $f \leftarrow (\alpha.x, (a + 1) \bmod n, b)$
   return $(f)$

```
main
 define the partition G = S_1 ∪ S_2 ∪ S_3
  (x, a, b) ← f(1, 0, 0)
  (x', a', b') ← f(x, a, b)
 while x ≠ x' do
    (x, a, b) ← f(x, a, b)
    (x', a', b') ← f(x', a', b')
    (x', a', b') ← f(x', a', b')
  if gcd(b' − b, n) ≠ 1
   then return ("failure")
    else return ((a − a')(b − b')^{-1} mod n)
```

در پایان شایان ذکر است که مساله لگاریتم گسسته کاربرد گسترده‌ای در علم رمزنگاری داشته و همواره جستجو برای الگوریتم‌هایی با زمان بهتر در جریان بوده است.