تبدیل فوریه روی میدان های متناهی

چکیده:
تبدیل فوریه عملیاتی است که طی آن می‌توانیم یک سیگنال را از حوزه زمان به حوزه فرکانس (و یا بالعکس) منتقل کنیم. این تبدیل در علوم مهندسی اهمیت زیادی داشته و در انتقال داده و پردازش سیگنال ها کاربرد فراوانی دارد. تبدیل فوریه به طور عادی روی اعداد مختلط تعریف می‌شود ولی می‌توان آن را روی میدان های دیگری نیز تعریف کرد. در این گزارش به تعریف تبدیل فوریه روی میدان های متناهی (به طور خاص، میدان های عدد اول یا Prime Field ها) می‌پردازیم و نشان می‌دهیم که تبدیل فوریه می‌تواند در تسریع عملیاتی مانند ضرب اعداد بسیار بزرگ، یا پیدا کردن چندجمله‌ای های لاگرانژ، کاربرد داشته باشد.

مقدمه:

میدان متناهی $\mathbb{F}_p$ را تعریف می‌کنیم (p یک عدد اول است):

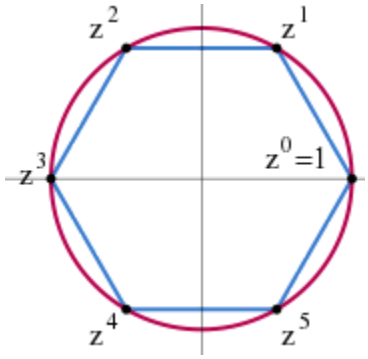
$$F_p = \{0, 1, \ldots, p-1\}$$

$a, b \in F_p$	$a + (b + c) = (a + b) + c$
$(a + b) \mod p$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
$(a * b) \mod p$	$a + b = b + a$
$a + (p - a) \mod p = 0$	$a \cdot b = b \cdot a$
$a * (1/a) \mod p = 1$	

عملیات جمع و ضرب روی این میدان به شکل بالا تعریف می‌شود. خاصیت شرکت‌پذیری و جابجایی روی این میدان برقرار است. همچنین عضو همانی ضرب و جمع وجود دارد. می‌توانیم بگوییم که یک نگاشت بین اعداد صحیح $\mathbb{Z}$ و اعضای این میدان وجود دارد، به طوری که اگر بجای اعداد صحیح موجود در یک رابطه جبری شامل ضرب و جمع، نسخه نگاشت شده این اعداد را قرار دهیم، رابطه برقرار می‌ماند. برای این تبدیل کافی است که به جای عدد n، باقی‌مانده تقسیم n بر p را قرار دهیم.

حال می‌خواهیم یک نگاشت بین اعداد مختلط به اعداد داخل میدان $\mathbb{F}_p$ پیدا کنیم. ساده ترین اعداد مختلط، اعدادی هستند که روی **دایره واحد** قرار دارند. اعداد داخل دایره واحد در میدان اعداد مختلط، به شکل $e^{j\omega}$ تعریف می‌شوند که ω یک عدد حقیقی است.

مثال: می‌خواهیم عدد مختلط z را به گونه ای بیابیم که $z^6 = 1$ شود. برای این معادله در میدان اعداد مختلط، ۶ جواب وجود دارد که می‌توان آنها را روی دایره واحد به شکل زیر نشان داد:



می‌توانیم این ریشه ها را به شکل زیر نیز نمایش دهیم:

$$e^{j2\pi k/6}$$
$$k \in \{0, 1, 2, 3, 4, 5\}$$

در این صورت، مقدار $e^{j2\pi/6}$ اصطلاحا، **ریشه واحد** یا Root of unity نامیده می‌شود که در محاسبه تبدیل فوریه کاربرد دارد. با ضرب ریشه واحد در خودش، متوجه یک تناوب می‌شویم. بعد از هر ۶ ضرب، به مقدار حقیقی 1 می‌رسیم و دوباره این تناوب تا بی‌نهایت تکرار می‌شود.

سوال: آیا رفتار تناوبی در میدان $\mathbb{F}_p$ نیز وجود دارد؟ به عبارتی، آیا برای مثال، می‌توان در یک عدد عضو میدان پیدا کرد به طوری که $r^6 = 1$ باشد؟ چنین اعدادی وجود دارند که به آنها نیز ریشه واحد می‌گویند.

برای مثال اگر $p = 13$، ریشه واحد می‌تواند 4 باشد. ($4^6 \ mod \ 13 = 1$) پس می‌توان گفت عدد 4 در میدان $\mathbb{F}_p$، معادل عدد مختلط $e^{j2\pi/6}$ در میدان مختلط است. بدیهی است که می‌توانیم بگوییم، معادل عدد مختلط $7e^{j2\pi 2/6}$ را می‌توانیم معادل $4^2 = 8 \ * \ 7$ بدانیم.

حال فرض کنید که یک چندجمله ای روی میدان $\mathbb{F}_{13}$ تعریف می‌کنیم. (این میدان یک ریشه واحد با مرتبه ۴ دارد که مقدار آن ۵ است. یعنی $5^4 = 1$) با این شرط که دامنه این تابع، ریشه های $r^4 = 1$ باشد. به عبارتی: $x \in \{1, 5, 12, 8\}$

$$P(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + a_3 x^3$$

به یک رفتار جالب می‌رسیم. می‌توانیم بگوییم که چند جمله‌ای بالا، معادل سیگنالی است که مولفه های فرکانسی مختلفی دارد. به عبارتی می‌توانیم این نگاشت را متصور شویم:

$$P(f) = a_0 + a_1 * e^{j2\pi f/4} + a_2 * e^{j2\pi f2/4} + a_3 * e^{j2\pi f3/4}$$

می‌دانیم که تبدیل فوریه روی اعداد مختلف به فرم زیر تعریف می‌شود:

$$G(f) = \int_{-\infty}^{\infty} g(t).e^{-2\pi jft}$$

با توجه به نگاشتی که از اعداد مختلط به اعداد عضو یک میدان متناهی به دست آوردیم، می‌توانیم تبدیل فوریه را داخل یک میدان متناهی نیز متصور شویم. کافی است که در فرمول ها و الگوریتم های تبدیل فوریه (مانند FFT) به جای مقادیر مختلط، نسخه‌ی نگاشت شده آن ها را قرار دهیم.

یک نسخه از تبدیل FFT که به جای اعداد مختلط روی اعداد $\mathbb{F}_p$ کار می‌کند داخل گزارش پیوست شده است:

```python
def fft(elems, inverse = False):
    log2_n = math.log2(len(elems))
    if not log2_n.is_integer():
        raise "Only 2^n number of elements accepted!"
    log2_n = math.ceil(log2_n)

    omega = # n-th root of unity

    n = len(elems)

    for k in range(0, n):
        rk = bitreverse(k, log2_n);
        if k < rk:
            elems[k], elems[rk] = elems[rk], elems[k]

    m = 1
    for _ in range(0, log2_n):
        w_m = omega.pow(n // (2 * m))
        k = 0
        while k < n:
            w = FieldP(1)
            for j in range(m):
                t = elems[k + j + m]
                t *= w;
                tmp = elems[k + j];
                tmp -= t;
                elems[k + j + m] = tmp
                elems[k + j] += t
                w *= w_m
            k += 2 * m
        m *= 2
```

تفسیر معنی خروجی ها:

همانطور که پیش‌تر گفته شد، تبدیل فوریه روی اعداد مختلط، سیگنال را از حوزه زمان به حوزه فرکانس منتقل می‌کند. تبدیل فوریه روی اعداد عضو $\mathbb{F}_p$، همانطور که نشان داده شد، میزان حضور درجات مختلف x را از روی خروجی های یک چندجمله‌ای فرضی استخراج می‌کند. به عبارتی، اگر

مقادیر خروجی یک چندجمله‌ای (که روی $\mathbb{F}_p$ تعریف شده است) به ازای ورودی های برابر با ریشه های دایره واحد را داشته باشیم، می‌توانیم آن چندجمله‌ای را با اجرای الگوریتم FFT پیدا کنیم. روش های دیگری نیز وجود دارند که با استفاده از آن‌ها می‌توان ضرایب این چندجمله‌ای متناظر را پیدا کرد، ولی پیچیدگی زمانی اجرای آن‌ها $O(n^2)$ می‌باشد. از آنجایی که الگوریتم FFT، پیچیدگی زمانی برابر با $O(n.log(n))$ دارد، از سایر روش های موجود مطلوب تر است.

نتایج:

در این گزارش به تعریف تبدیل فوریه روی میدان های متناهی پرداختیم، و سیر فکری برای رسیدن به این ایده را بررسی کردیم. دو پیاده‌سازی مختلف جهت پیدا کردن ضرایب چندجمله ای ها از روی خروجی آن‌ها را به زبان پایتون پیاده‌سازی کردیم که یکی از آنها با پیچیدگی زمانی $O(n^2)$، ضرایب چندجمله‌ای لاگرانژ را محاسبه می‌کند، و دیگری از طریق الگوریتم FFT این کار را انجام می‌دهد. صحت خروجی الگوریتم FFT را نیز بررسی کردیم. پیاده‌سازی ها در کنار گزارش پیوست شده‌اند.

```
65  elems = [FieldP(10), FieldP(120), FieldP(21), FieldP(30000)]
66  print("Expectation:", elems)
67  print('=' * 20)
68
69  omega = get_omega(2)
70  fft(elems, inverse=True)
71  print("Transform:")
72  pprint.pprint(elems)
73  print('=' * 20)
74
75  poly = PolynomialP(list(elems))
76  print("Evaluations:", [poly.evaluate(omega.pow(i)) for i in range(4)])
```

```
keyvan@keyvan-pc:~/pyrove$ python3 main.py
Expectation: [FieldP(10), FieldP(120), FieldP(21), FieldP(30000)]
====================
Transform:
[FieldP(131089687937815476198619351270464914594226381251319094556509146749846453036666),
 FieldP(393269063813446687442176561513222956370013086128342820457386857022077995581420),
 FieldP(131089687937815476198619351270464914594226381251319094556509146749846452886060),
 FieldP(393269063813446169749539546109566531195345201379571746881668023477000722186220)]
====================
Evaluations: [FieldP(10), FieldP(120), FieldP(21), FieldP(30000)]
```

منابع:

1. **Pinocchio: Nearly Practical Verifiable Computation** (2013)
2. **What are zk-SNARKs?** Zcash blog
3. **Quadratic Arithmetic Programs: from Zero to Hero**, Vitalik Buterin
4. **Zk-SNARKs, Under the hood**, Vitalik Buterin
5. **Fast Fourier Transforms**, Vitalik Buterin
6. **Field (mathematics)**, Wikipedia