소인수 분해 알고리즘 완전정복

08.

쇼어의 소인수 분해 알고리즘 구현

8. 쇼어의 소인수 분해 알고리즘 구현

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- 쇼어의 소인수 분해 알고리즘
  - 입력: 합성수 N ($N = p \times q$이고 p와 q는 소수)
  - 출력: N의 소인수 p, q
  - 입출력 사례:
    - $N = 15$
    - $p = 3, q = 5$ 또는 $p = 5, q = 3$

8. 쇼어의 소인수 분해 알고리즘 구현

- 쇼어의 소인수 분해 알고리즘
  1. 1보다 크고 N보다 작은 정수 a를 임의로 선택
  2. 만일, $\gcd(N, a) \neq 1$이면, p를 찾았다!
     - 따라서, $p = \gcd(N, a)$, $q = N/\gcd(N, a)$
  3. 함수 $f(x) = a^x (\bmod N)$의 주기(period) r을 찾는다.
     - 여기서 찾은 주기 r이 짝수가 아니면, 1번 단계부터 다시 시작한다.
  4. 주기 r로부터 두 개의 최대공약수 gcd_1, gcd_2 를 찾는다.
     - $gcd_1 = \gcd(N, a^{r/2} + 1)$, $gcd_2 = \gcd(N, a^{r/2} - 1)$
  5. gcd_1, gcd_2이 1과 N이라면, 1번 단계부터 다시 시작한다.
     - 아니면, 마침내 소인수들을 찾았으므로, $p = gcd_1$, $q = gcd_2$ 리턴

8. 쇼어의 소인수 분해 알고리즘 구현

1. 1보다 크고 N보다 작은 정수 a를 임의로 선택
2. 만일, $\gcd(N, a) \neq 1$이면, p를 찾았다!
   - 따라서, $p = \gcd(N, a)$, $q = N / \gcd(N, a)$

```
a = random.randint(2, N - 1)
gcd = math.gcd(N, a)
if (gcd != 1):
    return gcd, N // gcd
```

8. 쇼어의 소인수 분해 알고리즘 구현

3. 함수 $f(x) = a^x (\mathrm{mod}\, N)$의 주기(period) r을 찾는다.
   - 여기서 찾은 주기 r이 짝수가 아니면, 1번 단계부터 다시 시작.

```
r = findPeriod(N, a)
if (r % 2 != 0):
    continue
```

```
def findPeriod(N, a):
    for x in range(1, N):
        if ((a ** x) % N == 1):
            return x
    return -1
```

8. 쇼어의 소인수 분해 알고리즘 구현

4. 주기 r로부터 두 개의 최대공약수 gcd_1, gcd_2 를 찾는다.
   - $gcd_1 = \mathrm{gcd}(N, a^{r/2} + 1), gcd_2 = \mathrm{gcd}(N, a^{r/2} - 1)$
5. gcd_1, gcd_2이 1과 N이라면, 1번 단계부터 다시 시작한다.
   - 아니면, 마침내 소인수들을 찾았으므로, $p = gcd_1, q = gcd_2$ 리턴

```
gcd1 = math.gcd(N, a**(r//2) + 1)
gcd2 = math.gcd(N, a**(r//2) - 1)
if (gcd1 == 1 or gcd2 == 1):
    continue
return gcd1, gcd2
```

8. 쇼어의 소인수 분해 알고리즘 구현

```python
def factor3(N):
    while(True):
        a = random.randint(2, N - 1)
        gcd = math.gcd(N, a)
        if (gcd != 1):
            return gcd, N // gcd
        r = findPeriod(N, a)
        if (r % 2 != 0):
            continue
        gcd1 = math.gcd(N, a**(r//2) + 1)
        gcd2 = math.gcd(N, a**(r//2) - 1)
        if (gcd1 == 1 or gcd2 == 1):
            continue
        return gcd1, gcd2
```

8. 쇼어의 소인수 분해 알고리즘 구현

주니온TV@Youtube
자세히 보면 유익한 코딩 채널

- 왜 이렇게 느릴까?
  - 주기 찾기를 위한 모듈러 거듭제곱 연산

```
def findPeriod(N, a):
    for x in range(1, N):
        if ((a ** x) % N == 1):
            return x
    return -1
```

8. 쇼어의 소인수 분해 알고리즘 구현

- 빠른 모듈러 거듭제곱 연산

```
def findPeriodByModPow(N, a):
    for x in range(1, N):
        if (mod_pow(a, x, N) == 1):
            return x
    return -1

def mod_pow(a, x, N):
    y = 1
    while (x > 0):
        if ((x & 1) == 1):
            y = (y * a) % N
        x = x >> 1
        a = (a * a) % N
    return y
```

8. 쇼어의 소인수 분해 알고리즘 구현

- 쇼어의 소인수 분해 알고리즘

```
def factorize3(N):
    while(True):
        a = random.randint(2, N - 1)
        gcd = math.gcd(N, a)
        if (gcd != 1):
            return gcd, N // gcd
        r = findPeriodByModPow(N, a)
        if (r % 2 != 0):
            continue
        gcd1 = math.gcd(N, a**(r//2) + 1)
        gcd2 = math.gcd(N, a**(r//2) - 1)
        if (gcd1 == 1 or gcd2 == 1):
            continue
        return gcd1, gcd2
```

8. 쇼어의 소인수 분해 알고리즘 구현

▪ 양자 컴퓨터와 쇼어의 양자 알고리즘

- 쇼어의 알고리즘에서 지수 시간 복잡도를 가지는 부분
  - $f(x) = a^x \pmod N$ 함수의 **주기 찾기**
- **퀀텀 푸리에 변환** (QFT: Quantum Fourier Transform)
  - 양자 컴퓨터로 푸리에 변환을 병렬적으로 수행 (비트 수와 무관)
  - 역푸리에 변환(I-QFT: Inverse-QFT)으로 주기를 찾을 수 있음
- 쇼어 알고리즘을 수행하기 위한 양자 컴퓨터의 조건
  - N의 비트수(b) +I-QFT를 실행하기 위한 비트 수(2b) = 3b개의 큐비트 필요
  - RSA-2048을 깨기 위해서는 6,144 큐비트 필요
  - 단, 큐비트 수를 줄이기 위한 연구도 있음.

주니온TV@Youtube

자세히 보면 유익한 코딩 채널

https://bit.ly/2JXXGqz

- 여러분의 구독과 좋아요는 강의제작에 큰 힘이 됩니다.
- 강의자료 및 소스코드: 구글 드라이브에서 다운로드 (다운로드 주소는 영상 하단 설명란 참고)

https://bit.ly/3baJZBx