러시아 농부(농민) 알고리즘 (HW4)

2017156017 영인우

과제를 수행한 인원 : 1명 (Alone)

검색 엔진 : Google

참고 사이트 · https://blog.naver.com/inocent_xx/80166[illegible]887

· https://gpdh.tistory.com/20

· https://www.wikihow.com/Multiply-Using-the-Russian-Peasant-Method

· 정의와 유래

러시아 농민 알고리즘은 무엇인가? 이름부터 생소한 이 알고리즘이 무엇인지 알기 전에 기원과 때 알고리즘 이름이 '러시아 농민 알고리즘'인지 먼저 알아보았다. 이는 곱셈이 세간에 나타나기 전, 러시아의 한 농부가 곱셈이 필요했는데 곱하기를 할 줄 모르니 자신만의 방식을 발전시킨 데에서 유래한 곱셈 알고리즘이다. 말 그대로 곱셈 알고리즘 즉 한마디이며, 2로 곱하고, 나누고, 더하면 모든 곱셈을 계산할 수 있는 곱셈 방법이다.

· 알고리즘 수행법

러시아 농민 알고리즘의 수행 순서는 다음과 같다

1. 곱하고자 하는 두 수 A, B를 제일 첫 줄에 쓴다
2. A부터 시작하여 2로 나누고 나머지는 버린다
3. 더 이상 나눌 것이 없을 때까지 2로 나누는 것을 반복하여 결과를 아래에 차례대로 쓴다.
4. A를 나눈 만큼 B를 2로 곱한다. 예를 들어, A를 6번 나누었다면 B를 6번 곱한다.
5. A와 마찬가지로 B를 2로 곱한 수들을 아래에 차례대로 쓴다
6. A란에서 홀수가 있는 줄에 있는 B란의 숫자들을 모두 더하면, 그것이 곱셈의 결과가 된다.

알고리즘 수행법만 보면 확 와닿지 않아 예시를 들어보겠다

146 x 37을 계산해보도록 하자. A, 즉 146이 1이 될 때까지 나누고, 나머지는 모두 버려주겠다.

140/2 = 73…, 73/2 = 36…, 36/2 = 18…, 18/2 = 9…, 9/2 = 4…

4/2 = 2…, 2/2 = 1… → 2로 7번 나누었다.

그리고, B, 즉 37을 2로 나눈 횟수만큼 2로 곱한다.

37×2 = 74, 74×2 = 148, 148×2 = 296, 296×2 = 592,
592×2 = 1184, 1184×2 = 2368, 2368×2 = 4736

그리고 A를 2로 나눈 값들 중, 홀수인 값과, 같은 행의 B 값은 지운다

A	B
~~146~~	~~37~~
73	74
~~36~~	~~148~~
~~18~~	~~296~~
9	592
~~4~~	~~1184~~
~~2~~	~~2368~~
1	4736

=> A를 나눈 값들이 홀수인 값과 같은 행에 있는 B값을 모두 더한 값은
74 + 592 + 4736 = 5402 이다.
이 값은 실제 146×37 의 값인 5402 와 같다.

이 일련의 과정이 바로 '러시아 농민 알고리즘'의 수행 과정이다

·코드 구현

러시아 농민 알고리즘을 파이썬으로 구현해 보았다.

```
def ruspeaMul(a,b):
    if(a<0 or b<0):
        print("음수가 입력되었습니다")
    result = 0  # 러시아 농민 알고리즘을 이용한 결과값

    while(a>0): # a가 2로 나눈 값이 0일 때 종료
        if(a%2==1): # a가 홀수일 때의 b값을 모두 더함
            result += b

        a = a//2 # a를 2로 나누고 나머지는 버림
        b= b*2 # a를 2로 나눌 때마다 b를 2로 곱함
    return result  # 러시아 농민 알고리즘 완성
In : ruspeaMul(146,37)
Out: 5402
```

• 수행 시간 비교

코드를 구현하고 나니 러시아 농민 알고리즘의 수행시간은 기존 파씬에서 제공하는 기본 곱셈 방식과 얼마나 차이가 날까 궁금해졌다.
먼저, 러시아 농민 알고리즘을 이용해 1억번 계산했을 때의 수행시간은 알아내기위한 함수는 다음과 같다.

```
import time

def exeTime_ruspeaMul():
    start = time.time()
    for i in range(1, 100000000):
        ruspeaMul(146,37)
    end = time.time()
    return f"{end - start:.5f} sec"
```

```
In: exeTime_ruspeaMul()
Out: 132.37036 sec
```

러시아 농민 알고리즘은 1억번 수행하는데 걸린 시간은 약 132.37036초이다
그리고, 파씬 기본 곱셈 방식을 1억번 수행했을 때 걸린 시간을 알아내는 함수는 다음과 같다.

```
def comMul(a,b): # 파씬 기본 곱셈방식 함수
    return a*b

def exeTime_comMul():
    start = time.time()
    for i in range(1, 100000000):
        comMul(146,37)
    end = time.time()
    return f"{end - start:.5f} sec"
```

```
In: exeTime_comMul()
Out: 12.34244 sec
```

파이썬에서 제공하는 기본 곱셈 방식을 1억번 수행하는데 걸린 시간은 약 12.34244초 만큼 걸렸다.

내가 짠 러시아 농민 알고리즘이 최적의 효율을 지닌 수행 루틴이 아니더라도 차이가 많이 나는 것을 보아, 파이썬 기본 곱셈 방식이 컴퓨터가 연산처리를 진행하는데 확실히 더 효율적이라는 것을 알 수 있었다.

·개인적인 견해

과외상황이 있는 두 정수의 곱셈을 컴퓨터가 편리하게 계산할 수 있는 알고리즘으로 소개되고 있는 러시아 농민 알고리즘은, 비록 존재하는 모든 언어로 수행시간을 비교해본 것은 아니지만, 적어도 파이썬에서는 기본 제공 곱셈 방식보다 수행시간이 비교적 느린 것으로 보아 현재 시점에선 효율이 떨어지는 알고리즘이라 생각된다