CSE2011 Problem Solving, 2016 Spring

2013312343 이상헌

**Homework 3-1**

**1. 문제 이해**

(1) 문제

**Problem 3-1: Multiplication**

- For given positive integer values a and n, write an algorithm that calculates b=a^y with the minimum number of multiplications. To this end,

- For given y < 10000, find the array x, such that x[n]=y.

 x[0]=1, x[1]=1, x[2]=3

 If n>=3, x[n] = x[n1] + x[n2] (0 <= n1, n2 < n, n1 != n2)

- You will get more points with smaller n.

(2) 중요 정보

- 주어진 수열의 초기값은 x[0] =1, x[1] =1, x[2] =3 이다.

- 수열의 수는 그 왼쪽에 해당하는 수들 중에 서로 다른 두 수의 합이 될 수 있다.

(3) 문제 정의

- 위의 조건을 만족하는 수열로 입력된 수를 만드는 최단 경로를 생성하고, 수열의 각 수를 출력한다.

**2. 문제 해결**

(1) 해결 방법 결정

*Divide-And-Conquer 방식*

- 수열에서 새로 생성되는 수는 그 전에 존재하는 두 수의 합이다. 즉, 수열 내의 모든 수는 1과 3의 반복적인 덧셈으로 나타낼 수 있다. 이를 이용하여 입력된 수까지의 모든 수를 1과 3의 반복적인 덧셈으로 나누고, 여기서 나타나는 규칙성을 생각한다.

- 아이디어

수열 내에 나타날 수 있는 몇 가지의 수를 1과 3으로 나누어 생각해보자. Table 1은 수열에 나타날 수 있는 (수열의 성질을 만족하는) 숫자의 1과 3의 개수를 표시한다. 예시로 나타낸 수들은 x[n] = x[n-1] + x[n-2]의 성질을 만족하도록 하였다. 1과 3의 개수는 앞에서 더해진 수들 내의 1과 3의 개수를 더한 것이다. 예를 들어, x[8] = 47은 47 = 1\*8 + 3\*13으로 표현될 수 있다.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| n | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| x[n] | 4 | 7 | 11 | 18 | 29 | 47 | 76 |
| 1의 개수 | 1 | 1 | 2 | 3 | 5 | 8 | 13 |
| 3의 개수 | 1 | 2 | 3 | 5 | 8 | 13 | 21 |

Table 1. 수열에 나타날 수 있는 수의 1과 3의 개수.

Table 1의 1의 개수 행과 3의 개수 행에 나타나는 숫자들을 살펴보면, 규칙성을 발견할 수 있다. 이는 피보나치 수열이다. 사실, x[n] = x[n-1] + x[n-2]라는 규칙성을 바탕으로 x[n]의 숫자들을 생성하였기 때문에 당연히 나타나는 사실이다. 만일 피보나치 수열의 n번째 수를 a(n)이라 하면, x[n+2] = 1\* a(n) + 3\*a(n+1) 의 식을 만족한다.

입력된 수 y를 만드는 방법을 두 가지로 나누어 생각한다.

1. table 1과 같이 x[n] = x[n-1] + x[n-2]를 만족하는 수들을 y보다 크지 않은 최대의 수까지 생성한다.
2. 생성된 마지막 수와 y와의 차이를 생각한다. 두 수의 차보다 작지 않은 최대의 수까지, 생성 순서의 역순으로 수열 내의 수를 검사한다. 찾았다면 그 수와 마지막으로 생성된 수를 더하여 다음 수를 생성한다.
3. 마지막으로 생성된 수가 y가 될 때까지 (2)를 반복한다.

예를 들어 y값으로 45를 입력 받았다고 가정하자.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| X[n] | 1 | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 47 |

Table 2. y=45 일 때 (1)의 결과

(1)에서는 1, 1, 3부터 시작하여, 45보다 작으면서 최대의 수가 나타날 때까지 피보나치 수열을 만족하도록 수열의 다음 수를 계속하여 생성한다. 즉, 3 다음에 3+1=4를, 4 다음에 4+3=7을, 7 다음에 7+4=11을 생성한다. 29 다음으로 생성될 수 있는 47은 45보다 크기 때문에 생성되지 않는다. (1)의 결과 수열은 1, 1, 3, 4, 7, 11, 18, 29가 된다. (Table 2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| X[n] | 1 | 1 | 3 | 4 | 7 | 11 | 18 | 29 | 40 |

Table 3. y=45 일 때 (2)의 과정

(2)에서는 입력된 수와 (1)의 마지막 수와의 차이를 생각한다. 45-29 = 16이다. (1)의 결과로 생성된 수열의 오른쪽 끝에서부터, 16보다 크면서 최소의 수가 나타날 때까지 검사한다. 29 -> 18 -> 11 순으로 검사되어, 11이 조건을 만족하는 수가 된다. 이후 29와 11을 더한 40이 수열의 다음 수로 저장된다. 이와 같은 방법을 반복하면, 생성되는 수열의 결과는 1, 1, 3, 4, 7, 11, 18, 29, 40(=29+11), 43(=40+3), 45 가 된다.

- 이점

입력된 수 y가 굉장히 큰 경우, y까지의 접근을 최소화 해야한다. 이는 수열에 이미 존재하는 수들 중에 가장 큰 두 수를 계속하여 더하는 것으로 최소화할 수 있다. Y와 어느 정도 근접한 경우부터, y를 생성할 수 있는 세세한 경로를 파악한다. 세세한 경로를 파악하는 단계에서도 y를 넘지 않는, 가장 큰 수를 더함으로써 접근을 최소화한다. 즉, 본 아이디어는 수열의 규칙을 바탕으로 하면서 생각할 수 있는 모든 방면에서 접근을 최소화할 수 있도록 한다. 또한 정확한 검증을 통해 정확한 결과를 나타낼 수 있도록 한다.

**3. 코드 작성**

- 본 아이디어 및 설명을 바탕으로 코드를 작성하였다.

int cal\_x(int x[], int y) {

int n = 0;

int fib[1000], i = 0, k = 0, min, sub = 0;

fib는 피보나치 수열을 저장하는 int형 1차원 배열이다. I, k는 count variable이다. Min은 위의 알고리즘의 (1) 단계에서, y보다 작으면서 최대의 수를 저장하는 변수이다. Sub은 (2) 단계에서, y와 마지막으로 생성된 수의 차이를 저장하는 변수이다.

if (y == 1) {

n = 1;

return n;

}

if (y == 2) {

n = 4;

x[n - 1] = 2;

return n;

}

if (y == 3) {

n = 3;

return n;

}

if (y == 4) {

n = 4;

x[n - 1] = 4;

return n;

}

Y의 초기 값이 1, 2, 3, 4인 특이한 경우에 대해서는 오류가 발생할 수 있기 때문에, 직접 입력하여 수열을 생성하였다.

fib[0] = fib[1] = 1;

for (i = 0; i < 998; i++)

fib[i + 2] = fib[i + 1] + fib[i];

fib 수열을 입력하였다. 피보나치 수열은 첫 번째, 두 번째 숫자는 1이고, n>3에 대하여 a(n) = a(n-1) + a(n-2)를 만족한다.

i = 0;

n = 3;

while (1) {

k = 1 \* fib[i] + 3 \* fib[i + 1];

if (k > y) {

min = x[i + 2]; break;

}

else {

x[n] = k;

n++;

if (k == y) return n;

}

i++;

}

앞서 설명한 알고리즘의 (1) 단계에 해당한다. 1, 1, 3부터 시작하여, y보다 작지만 최대인 수가 나타날 때까지 피보나치 수열의 성질을 만족하도록 수열을 만들고 저장한다. 만일 y보다 큰 수가 나타난다면, 그 전의 수를 min 변수에 저장한다. 즉, min 변수는 y보다 작지만 최대인 수가 된다.

i = n - 1;

while (1) {

sub = y - min;

while (sub < x[i]) {

i--;

}

x[n] = x[n - 1] + x[i];

min = x[n];

n++;

if (min == y) return n;

}

return n;

}

앞서 설명한 알고리즘의 (2) 단계에 해당한다. Y와 min의 차를 sub 변수에 저장하고, sub보다 크면서 최소인 수가 나타날 때까지, 생성된 수열의 역순으로 검사한다. 해당하는 수는 x[i]가 되고, x[i]와 마지막으로 생성된 수를 더하여 새로운 수를 생성한다. Min은 다시 생성된 수가 되고, 위의 단계를 반복한다. 마지막으로 생성된 수가 y가 되면 n을 리턴 하고, 함수를 종료한다.

**4. 확인 및 분석**

- 본 아이디어를 이용하여 프로그램을 작성하였다. 이 프로그램이 정확한 결과값을 도출하는지를 알아보기 위해서 간단한 예시를 입력하고, 결과를 비교하였다.



Figure 1. 예시 1 Input

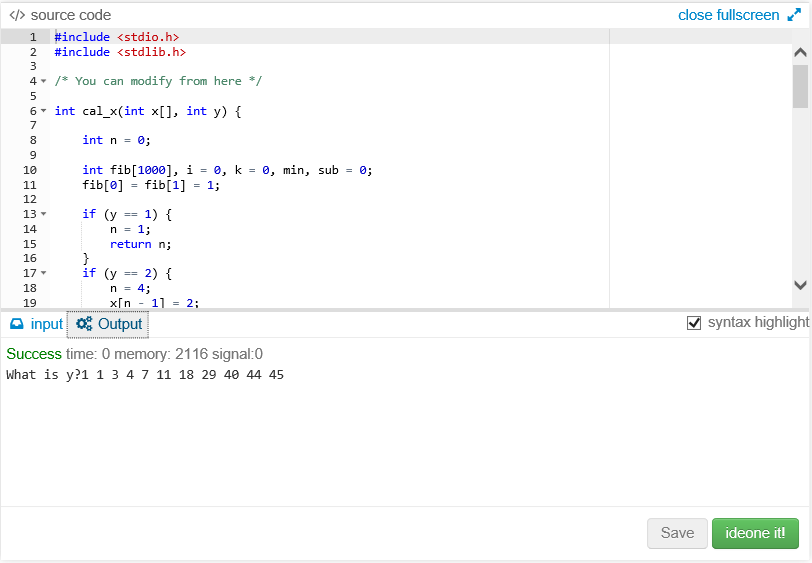


Figure 2. 예시 1 Output

- Figure 1은 y에 45를 입력한 예시이다. 45를 입력한 경우, 알고리즘에서 설명한 바와 같이 1, 1, 3, 4, 7, 11, 18, 29까지 피보나치 수열의 규칙을 바탕으로 y까지 접근한 후, y와 마지막 수의 차이를 비교하여, 40, 44, 45 순으로 생성한다. Figure2는 결과가 정확히 출력되었다는 것을 나타낸다.

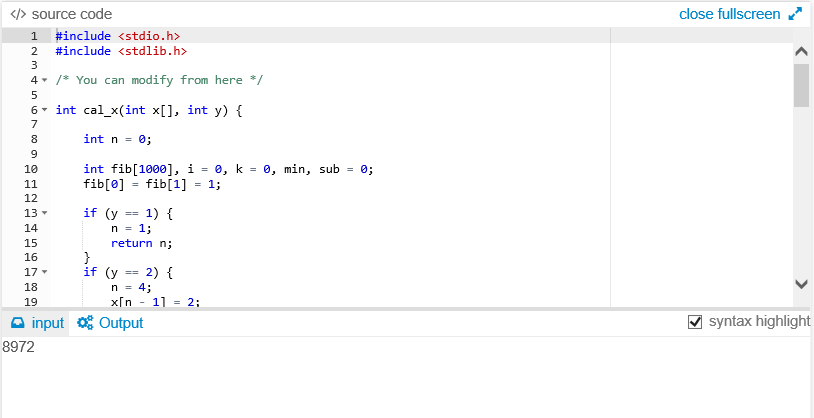


Figure 3. 예시 2 Input

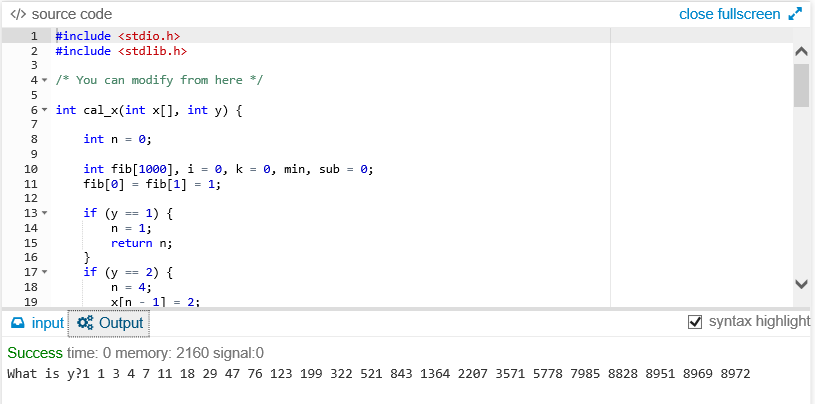


Figure 4. 예시 2 Output

- Figure 4는 비교적 높은 숫자를 입력한 예시이다. Y에 8972를 입력한 결과, Figure4와 같은 결과가 나타났다. 검산해 본 결과, 모든 수는 문제에서 주어진 수열의 성질을 만족하는 수들이고, 마지막으로 나타난 수는 입력된 수와 동일하기 때문에 정확한 결과가 나타났다고 할 수 있다.

**5. 고찰**

- 본 문제는 규칙적인 수열을 이용하여, 입력된 수를 생성하는 경로를 출력하는 프로그램을 구현하는 것이다. 수열의 초기 값이 1과 3으로 되어있기 때문에, 모든 수를 1과 3으로 생성되는 모든 수를 1과 3으로 나누어 생각할 수 있었다. 또한 최소의 경로를 결과로 출력해야하는 요소는, 피보나치 수열과 역순으로의 검사 등으로 고려하였다. 이는 최대의 수를 더하는 것이 입력된 수를 생성하는 최단 거리가 된다고 생각하였다. 만일, 최대의 수를 더하는 것이 아닌 다른 방법이 최단 경로가 된다면, 이러한 예외적인 요소에 대한 아이디어를 본 프로그램에 추가적으로 기입해야 한다.

**6. 참고문헌 및 사용**

- [www.ide.com](http://www.ide.com) : C언어 구현 및 실행.

- 이진규 교수님, [문제해결기법] CSE2011\_2016spring\_Lecture\_Note05, 2016