1

안녕하세요 저는 피보나치 넘버 발표를 진행할 20201635 컴퓨터공학과 전찬 입니다.

과제를 해결하는 방법, 피보나치 수를 구하는 3가지 방법, 각 방법에서의 속도 비교에 대해 발표를 진행하겠습니다.

2

첫 번째로, 과제의 출력 예시에서, 직접 실행하는 것과 import해서 실행하는 것이 차이가 존재함을 알 수 있습니다. run의 경우에는 바로 input을 받으며, import의 경우에는 함수를 call할 때만 피보나치 수 연산을 수행한다는 것을 알 수 있는데,

3

이는 교수님이 설명해주신 아래 형태로 구현 가능하며 코드는 아래와 같습니다.

4

이러한 피보나치 수를 구하는 방법은 다양하게 존재하며, 저는 그 중에서 3가지 방법을 소개하겠습니다. 첫 번째는 가장 쉬운 방법인 recursive function, 재귀 함수를 사용한 형태입니다. 피보나치의 점화 식을 바탕으로 함수를 구현해서 피보나치 수를 구하는 형태입니다.

5

이 경우에, 코드의 구현, 이해가 쉬울 수 있으나, 만약 N = 50, 100 정도로 커지는 경우 몇 년, 혹은 몇 세기가 걸릴 수도 있습니다.

6

n = 5인 경우에 구하는 형태를 살펴보면, 5->4,3 4->3,2 3->2,1 과 같은 형태로 계속 분할해서 총 15번의 계산이 필요하다는 것을 알 수 있습니다. 이는 결과적으로 F(n)을 구하기 위해서 2n의 시간이 필요하다는 큰 단점을 갖게 됩니다.

7

따라서 다른 형태를 생각할 수 있는데, 이와 같은 Loop 형태입니다. fibo\_n이라는 변수에 이전 두 값을 계속해서 할당해줌을 통해 약 n번의 반복으로 n번째 Fibonacci number을 구할 수 있는 형태입니다.

8

이 경우에는 재귀 함수에서 피보나치 수를 구할 때 F(n)->F(n-1)과 같이 위에서 아래로 구하는 것과 달리, F(2)->F(3) 처럼 아래에서 위로 구하게 됩니다. 따라서 코드를 작성, 이해하는 데에 재귀 함수 형태보다는 약간 어려움이 존재합니다. 하지만 n번의 반복을 통해서 n번째 피보나치 수를 알 수 있기 때문에, 재귀 함수 형태보다 훨씬 빠른 속도로 큰 피보나치 수를 구할 수 있습니다.

9

마지막 방법은 Dynamic programming을 바탕으로 구현한 형태인데, ~~~

10

코드로 구현하면 다음과 같으며, fibo\_data[]라는 list의 k번째 원소에 피보나치 k번째 값을 저장하는 형식으로 구현할 수 있습니다. 그 이외에는 재귀 함수 형태와 동일하게 진행됩니다.

11

이 경우에 n번째 수를 구하기 위해, fibo(0) ~fibo(n-1)을 한 번씩만 계산하면 되기 때문에, 약 n번의 계산이 필요하다는 것을 파악할 수 있습니다. 또한 저장하는 것을 제외하면 재귀 함수와 형태가 동일하기 때문에 이해하기 더 쉬울 수 있다는 장점도 가지고 있습니다.

12

그래서 재귀 함수에서 fibonacci(5)를 구하기 위해서는 많은 function call이 필요했는데, 이 경우에는, 왼쪽부터 fibo\_data[2], fibo\_data[3], fibo\_data[4](오타입니다.)에 저장되어 이후에는 저장된 데이터를 사용할 수 있기 때문에, 오른쪽의 과정을 대부분 생략할 수 있습니다. 따라서 이를 생략하면

13

위 형태로 9번의 function call로 fibonacci(5)를 구할 수 있습니다.

14

마지막으로 %%timeit 매직 커맨드를 통해 각 방법에서의 시간을 비교해보면, 다음과 같습니다.

15

16

기본적으로 loop, dynamic, recursive 순서로 속도가 빠르며, n번의 수행으로 구할 수 있는 loop, dynamic와 다르게, 2n의 수행이 걸리는 recursive가 훨씬 느리다는 것을 파악할 수 있습니다. 또한 n = 20 -> 30으로 증가시킬 때 loop, dynamic은 증가 폭이 크지 않으나, recursive는 100배가 넘게 증가한 형태임을 파악할 수 있습니다.

17

또한 import를 바탕으로 함수를 call하면 다음과 같은데, 그냥 함수를 호출할 때와 큰 차이는 존재하지 않다는 것을 파악할 수 있습니다.

감사합니다