Algorithm

[Algorithm] 28강 : 다이나믹 프로그래밍의 정의와 구현

2020. 11. 16. 22:03 수정 삭제 공개

1. 다이나믹 프로그래밍

1-1. 다이나믹 프로그래밍(동적 계획법) 이란?

- 다이나믹 프로그래밍은 메모리를 적절히 사용하여 수행 시간 효율성을 비약적으로 향상 시키는 방법
- 이미 계산된 결과는 별도의 메모리 영역에 저장하여 아시 계산하지 않도론 한다.
- 두 가지 방식으로 구성 (탑다운 / 보텀업)

※동적(Dynamic) 할당 이란? - 프로그램이 실행되는 도중에 실행에 필요한 메모리를 할당하는 기법

1-2.다이나믹 프로그래밍의 조건

- 1.최적 부분구조 (Optimal Substructure)
  - 큰 문제를 작은 문제로 나눌수 있으며 작은 문제의 답을 모아서 큰 문제를 해결할 수 있다.
- 2.중복되는 문제(Overlapping Subproblem)
  - 동일한 작은 문제를 반복적으로 해결

2. 피보나치 수열(일반 재귀함수)

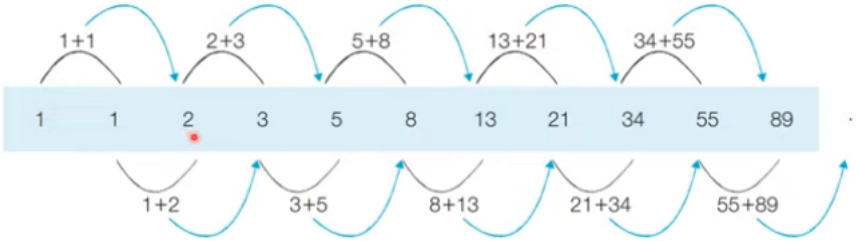
2-1.피보나치 수열이란?

피보나치 수열은 다음과 같은 형태의 수열이며, 다이나믹 프로그래밍으로 효과적으로 계산할 수 있다.

```
1,1,2,3,5,8,13,21,34,55,89
```

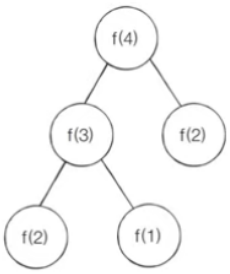
- 점화식이란 인접한 항들 사이의 관계식을 의미
- 피보나치 수열을 점화식으로 표현하면

An = An-1+An-2, A1=1,A2=1



피보나치 수열이 계산되는 과정

- 피보나치 수열이 계산되는 과정은 다음과 같이 표현할 수 있습니다.
  - n번째 피보나치 수를 $f(n)$라고 할 때 4번째 피보나치 수 $f(4)$를 구하는 과정은 다음과 같습니다.

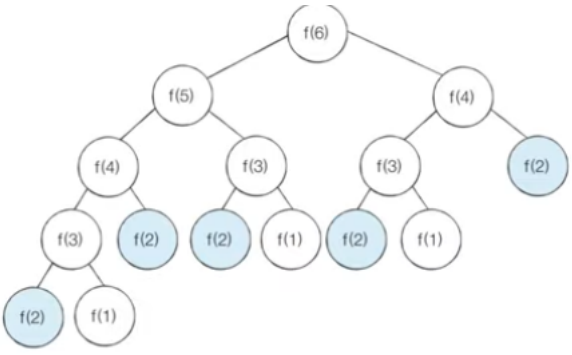


2-2.피보나치 수열 구현

```
# 피보나치 함수를 재귀함수로 구현

def fibo(x):
    if x == 1 or x == 2:
        return 1
    retrun fibo(x-1) + fibo(x-2)
print(fibo(4))
```

2-3.피보나치 수열의 시간 복잡도 분석

- 단순 재귀함수로 피보나치 수열을 해결하면 지수 시간 복잡도를 가지게 된다.
- 다음과 같이 f(2)가 여러 번 호출 되는 것을 확인할 수 있다.(중복되는 부분 문제)



피보 나치 수열의 시간복잡도

- 세타 표기법: θ(1.618)
- 빅오 표기법: O(2n승)

이때 빅오표기법을 기준으로 f(30) 을 계산하기위해서는 10억가량의 연산을 수행해야 한다.

3 피보나치 수열(다이나믹 프로그래밍)

3-1 피보나치 수열의 효율적인 해법

- 다이나믹 프로그래밍의 사용 조건을 만족하는지 확인

1.최적 부분구조: 큰 문제를 작은 문제로 나눌 수 있다.
2.중복되는 부분 문제 : 동일한 작은 문제를 반복적으로 해결

- 피보나치 수열은 다이나믹 프로그래밍의 사용 조건을 만족

3-2. 메모제이션(Memoization)

- **메모제이션은 다이나믹 프로그래밍을 구현하는 방법 중 하나**
- **한 번 계산한 결과를 메모리 공간에 메모하는 기법**
  - **같은 문제를 다시 호출하면 메모했던 결과를 그대로 가져온다.**
  - **값을 기록해 놓는다는 점에서 캐싱(Caching)이라고 한다.**

3-3.탑다운 VS 보텀업

- 탑다운(메모제이션) 방식은 하향식이라고 하며 보텀업 방식은 상향식이라고 한다.
- 다이나믹 프로그래밍의 전형적인 형태는 보텀업 방식이다.
  - 결과 저장용 리스트는 DP 테이블이라고 한다
- 엄밀히 말하면 메모제이션은 이전에 계산된 결과를 일시적으로 기록해 놓는 넓은 개념을 의미
  - 따라서 메모제이션은 다이나믹 프로그래밍에 국한된 개념은 아니다
  - 한 번 계산된 결과를 담아 놓기만 하고 다이나믹 프로그래밍을 위해 활용하지 않을 수 도 있다.

3-4.피보나치 수열 구현(탑 다운)

```
#한 번 계산된 결과를 메모제이션하기 위해 리스트 초기화
d = [0]*100

# 피보나치 함수를 재귀함수로 구현(탑 다운 다이나믹 프로그래밍)
def fibo(x):

#종료 조건(1 혹은 2일 때 1을 반환)
  if x == 1 or x == 2:
    return 1
```

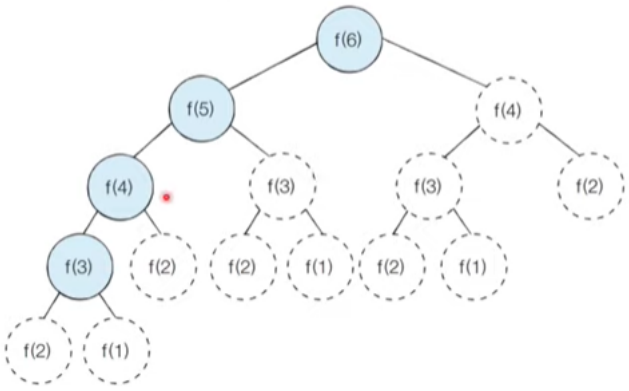
```
# 이미 계산한 적 있는 문제라면 그대로 반환
  if d[x] != 0 :
    return d[x]

# 아직 계산하지 않은 문제라면 점화식에 따라서 피보나치 결과 반환
  d[x] = fibo(x-1) + fibo(X-2)
  retrun d[x]

print(fibo(99))
```

3-5.피보나치 수열 구현(보텀업)

```
# 앞서 계산된 결과를 저장하기 위한 DP 테이블 초기화
d= [0] * 100

# 첫 번째 피보나치 수와 두 번째 피보나치 수는 1
d[1] = 1
d[2] = 1
n == 99

# 피보나치 함수(Fibonacci Function) 반복문으로 구현(보텀업 다이나믹 프로그래밍)
for i in range(3, n+1):
  d[i] = d[i-1] + d[i-2]

print(d[n])
```

3-6.피보나치 수열 : 메모이제이션 동작 분석

이미 계산된 결과를 메모리에 저장하면 다음과 같이 색칠된 노드만 처리한다.



이때의 시간 복잡도는 **O(N)** 이다.

4.다이나믹 프로그래밍 VS 분할정복

- 다이나믹 프로그래밍과 분할 정복은 모두 최적 부분 구조를 가질 때 사용할 수 있다.
  - 큰 문제는 작은 문제로 나눌 수 있으며 작은 문제의 답을 모아서 큰 문제를 해결할 수 있는 상황이다.

- 다이나믹 프로그래밍과 분할 정복의 차이점은 분할 정복의 차이점은 **부분 문제의 중복**이다.
  - 다이나믹 프로그래밍 문제에서는 각 부분 문제들이 서로 영향을 미치며 부분 문제가 중복된다.
  - 분할 정복 문제에서는 동일한 부분 문제가 반복적으로 계산된지 않는다.

5.다이나믹 프로그래밍 문제에 접근하는 방법

- 주어진 문제가 다니아믹 프로그래밍 유형임을 파악하는 것이 중요
- 가장 먼저 그리디, 구현,완전, 탐색등의 아이디어로 문제를 해결할 수 있는지 검토
  - 다른 알고리즘으로 풀이 방법이 떠오르지 않으면 다이나믹 프로그래밍 고려
- 일단 재귀함수로 비효율적인 오나전 탐색 프로그램을 작성한 뒤에 작은 문제에서 구한 답이 큰 문제에서 그대로 사용될 수 있으면, 코드를 개선하는 방법을 사용할 수 있다.

이 자료는 동빈 나 님의 이코 테 유튜브 영상을 보고 정리한 자료입니다.
www.youtube.com/watch?v=m-9pAwq1o3w&list=PLRx0vPvlEmdAghTr5mXQxGpHjWqSz0dgC

'**Algorithm**' 카테고리의 다른 글□

[Algorithm] 28강 : 다이나믹 프로그래밍의 정의와 구현□

[Algorithm] 27강 : 이진 탐색 기초 문제 풀이□

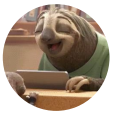
[Algorithm] 26강 : 이진 탐색 알고리즘 정의와 구현□

[Algorithm] 25강 : 정렬 알고리즘 복잡도 비교 및 기본 문제□

[Algorithm] 24강 : 계수 정렬의 정의와 구현코드□

[Algorithm] 23강 : 퀵(quick) 정렬의 정의와 구현코드□

다이나믹 프로그래밍 다이나믹 프로그래밍 구현 다이나믹 프로그래밍 정의 피보나치수열



나아무늘보

혼자 끄적끄적하는 블로그 입니다.