**PA3**

**서준표**

***PA 3 :***

n가지 종류의 동전이 있다. 각각의 동전이 나타내는 가치는 다르다. 이 동전을 적당히 사용해서, 그 가치의 합이 k원이 되도록 하고 싶다. 그 경우의 수를 구하시오. 각각의 동전은 몇 개라도 사용할 수 있다.

사용한 동전의 구성이 같은데, 순서만 다른 것은 같은 경우이다.

***Facts :***

* 각 코인의 개수와 가치를 저장할 배열
* 다이나믹 프로그래밍을 위한 계산 정보 저장 배열
* 만들고 싶은 금액

***OverView :***

일반적으로 다이나믹 프로그래밍을 구현하기 위해 낮은 단계 부터의 계산을 실행하고, 그 정보를 저장해 분할 접근과 메모이제이션을 이용한 다이나믹 프로그래밍을 구현한다.

동전의 개수와 원하는 수 입력받기 -> 배열 생성 후 base case 초기화하기 -> 점화식을 이용한 bottom – top 반복문으로 조합의 수 찾기.

***Algorithm :***

**1) 규칙성을 발견해 점화식을 세운다.**

1원 2원 3원 4원 5원 6원 7원 8원 9원 10원

1원 1 1 1 1 1 1 1 1 1 1

1,2원 1 2 2 3 3 4 4 5 5 6

1,2,5원 1 2 2 3 4 5 6 7 8 10

2개사용 규칙 : n원(2개사용) = n원(1개사용) + n-2원(2개사용)

1원 2원 3원 4원 5원 6원 7원 8원 9원 10원

1원 1 1 1 1 1 1 1 1 1 1

1,2원 1 2 2 3 3 4 4 5 5 6

1,2,5원 1 2 2 3 4 5 6 7 8 10

3개사용 규칙 : n원(3개사용) = n원(2개사용) + n-5원(3개사용)

* ***n개 사용 규칙 : x원(n개사용) = x원(n-1개사용) + x-n원(n개사용)***

의 점화식 유추 가능하다.

\*\*\* x-n원에서 제일 작은 동전의 가치 즉 n원이 1일경우 반복문에서 (x-n원) 에서의 충돌을 피하기 위해 다이나믹 프로그래밍 계산 정보 배열의 0번째값을 1으로 두고, 반복문의 시작을 배열 1부터 한다. (저장할 때 또한 마찬가지)

또한 막연히 1부터 ~ 원하는 금액 만큼 단순히 반복문을 돌릴 필요 없이,

동전의 정보 배열에서 존재하는 동전의 가치만 뽑아와서 반복문을 실행하면 시간을 절약할 수 있다.

***Time Complexity :***

***n 개의 동전을 k(원하는 수)번만큼 -> O(n\*k)***