〈 GBC_Algorithm PA3 〉

27 기 최 하영

1. FACT

- 각각 다른 가치를 가지고 있는 N가지 종류의 동전이 주어졌을 때,
  가치의 합이 k원이 되도록 하는 경우의 수 구하기

- 입력 : n(동전의 종류), k (원하는 합)
  단, 범위는 $(1 \leq n \leq 100,\ 1 \leq k \leq 10{,}000)$

- 출력 : 경우의 수 출력 (단, 경우의 수는 2^{31}보다 작음)

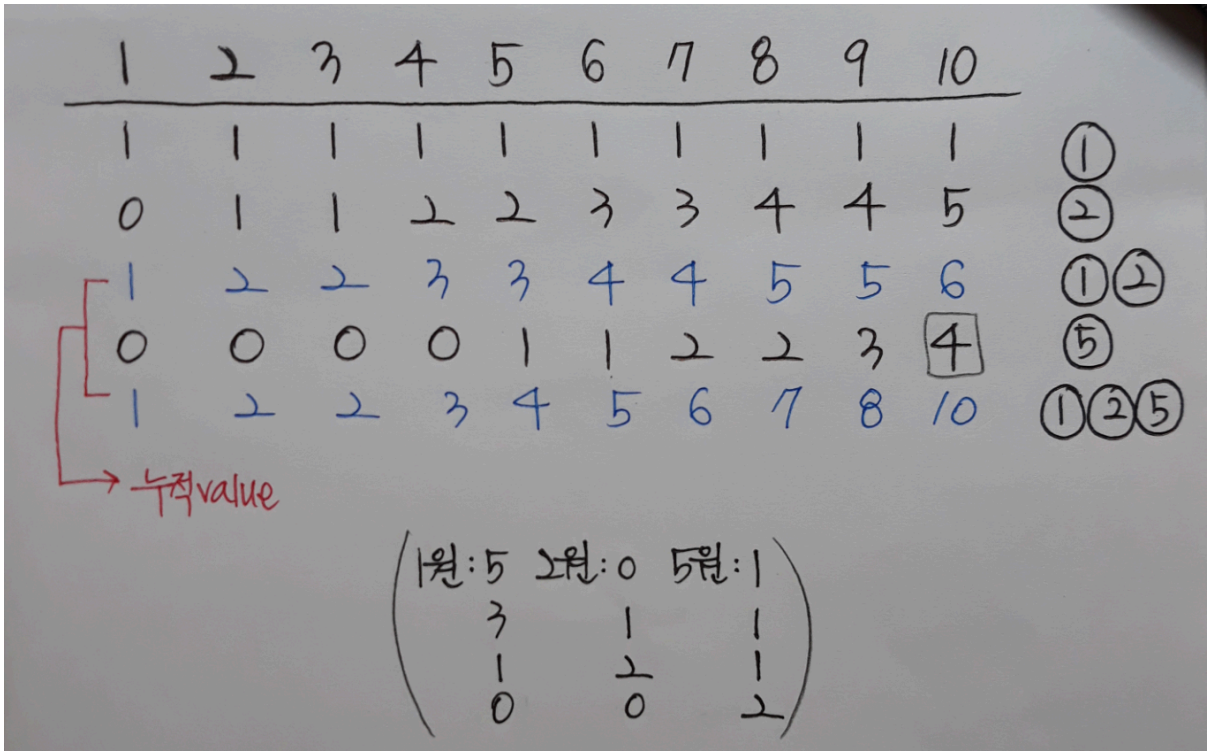
- 각각의 동전은 여러 번 사용할 수 있으며, 순서만 다른 경우는 같은 경우로 취급

2. Overviews

1) Dynamic Programming 을 사용
2) 동전의 가치 저장할 배열 필요
3) 합이 k 원이 되는 경우의 수를 저장할 배열 필요
4) base case 초기화, 점화식을 세워 for loop 안에서 조건 충족 시키기

3. Algorithm

1) 동전의 종류 n $(1 \leq n \leq 100)$과 원하는 합 K(, $1 \leq k \leq 10{,}000$) 입력 받기
2) dp[k]는 합이 k 가 되는 경우의 수를 의미한다고 가정
3) 다음과 같은 규칙을 통해 점화식을 유추한다
   i) 1 을 사용할 때 가능한 경우의 수
   ii) 2 를 사용할 때 가능한 경우의 수
   iii) 1, 2 를 모두 사용할 때 가능한 경우의 수
   iv) 5 를 사용할 때 가능한 경우의 수
   v) 1, 2, 5 를 사용할 때 가능한 경우의 수



4) 가치가 n 인 코인이 있을 때, j 원의 경우의 수에서 n 을 더해주면 j+n 원의 경우의 수가 됨 ➔ 점화식 : dp[j] = dp[j] + dp[j-coin[i]]

5) 가치의 합 dp[k] → 모든 코인을 사용해 봐야 함

6) 동전 전체 개수만큼 반복 → 현재 가치의 합에서 특정한 동전의 가치를 뺀 경우

```
for(int i = 1; i <= n; i++){
    for(int j = 1; j <= k; j++){
        if(j-coin[i] >= 0)
            dp[j] += dp[j-coin[i]];
    }
}
cout << dp[k];
```

→ 새로운 동전으로 만들 수 있는 가치의 개수 : 전 배열 값을 현재 배열

4. Time complexity

O(n) → n 개의 동전 종류를 K 번만큼 for loop 돌린다.