6. Greedy Algorithm, Bitmask

2017010698
수학과 오서영

Greedy Algorithm

: 욕심쟁이 알고리즘.
미래를 생각하지 않고 그 순간에서 가장 최선의 선택을 하는 기법

5585번 거스름돈 문제

: 타로는 자주 JOI잡화점에서 물건을 산다. JOI잡화점에는 잔돈으로 500엔, 100엔, 50엔, 10엔, 5엔, 1엔이 충분히 있고, 언제나 거스름돈 갯수가 가장 적게 잔돈을 준다. 타로가 JOI잡화점에서 물건을 사고 카운터에서 1000엔 지폐를 한장 냈을 때, 받을 잔돈에 포함된 잔돈의 갯수를 구하는 프로그램을 작성하시오.

-> 가치가 큰 돈을 우선적으로 주면 된다.

```
int money = 1000 - sc.nextInt();
int[] array = {500,100,50,10,5,1};
int idx = 0; int ans = 0;
while(money != 0) {
    int change = money / array[idx];
    money -= change * array[idx++];
    ans += change; }

System.out.println(ans);
```

그리디 알고리즘은 무조건 최적이 아니라는 것

-> 성능이 빠르기 때문에 어느 정도 최적에 가깝게 구하는 근사적인 방법으로 사용 할 수 있다.

Bitmask

: 정수의 이진수 표현을 자료 구조로 쓰는 기법

장점

1. **더 빠른 수행 시간** : 거의 O(1)에 구현됨

2. **더 간결한 코드** : 반복문 없이 한 줄

3. **더 작은 메모리 사용량** :
같은 데이터를 더 적은 메모리를 사용해 표현가능

4. **연관 배열을 배열로 대체** :
map<vector<bool>, int> -> int[]

비트 연산자

: 비트마스크를 사용하기 위해서는 정수 변수를
비트별로 조작할 수 있는 비트 연산자를 사용해야함

1. **AND** : 두 정수의 해당 비트가 모두 켜져 있을 때만 결과의 비트를 켬
-> 1011 AND 1000 = 1000

2. OR : 두 비트중 하나라도 켜져 있을 경우
3. XOR : 하나는 켜져 있고 하나는 꺼져 있을 경우

4. SHIFT :
1100을 왼쪽으로 2비트 시프트 = 110000

비트 연산자를 이용한 집합 구현

: N비트 정수 변수는 0부터 N-1까지의 정수 원소를
가질 수 있는 집합이 된다
원소 i가 집합에 속해 있는지 여부
= 2^I 를 나타내는 비트가 켜져있는지 여부

Ex) 집합 {1, 4, 5, 6, 7, 9} 를 표현하는 정수
: 2^1 + 2^4 + 2^5 + 2^6 + 2^7 + 2^9
= 10 1111 0010 = 754

ex) 비트마스크 연산을 이용해 집합을 조작하는 예제 – 피자집

피자에 0~19번까지 번호를 갖는 스무가지 토핑을
주문 시 넣기/넣지 않기 선택 할 수 있다.

공집합 : 0
꽉 찬 집합 (스무 개의 토핑을 모두 포함하는 집합) ->
모든 비트가 켜진 숫자

```
// 1<<20은 1 뒤에 20개의 0이 있는 정수
// 여기서 1을 빼면 20개의 비트가 모두 켜진 수를 얻는다
int fullPizza = (1<<20) -1;
```

< 원소의 삭제 >

```
toppings &= ~(1<<p);
// ~ (1<<p) 은 Not연산을 하므로 해당 비트만 꺼지고
다 켜진 수가 된다.
// 이 수와 &연산을 하므로 나머지 비트는 유지되고 항상
p번 비트는 꺼지게 된다.
```

< 집합 크기 구하기 > : 함수 제공

```
//gcc,g++ __builtin_popcount(toppings)
//toppings는 집합
//visual c++ __popcnt(toppings)
```

Reference

[1] 그리디(Greedy) 알고리즘, https://mygumi.tistory.com/121

[2] 구종만.『프로그래밍 대회에서 배우는 알고리즘 문제해결전략』. 인사이트(2012)