알튜비튜
그리디

오늘은 '탐욕법'이라고도 불리는 그리디 알고리즘에 대해 배울 것입니다.
가장 직관적인 알고리즘 중 하나죠.

도전 문제

19539번 : 사과나무 - Gold 5

문제

- 원하는 사과나무 배치가 주어졌을 때 해당 배치를 만들 수 있는지 여부를 출력하기

- 한 물뿌리개는 나무 하나를 1만큼 성장시키고, 다른 물뿌리개는 나무 하나를 2만큼 성장시킨다.
- 두 개의 물뿌리개들을 동시에 사용해야 하며, 물뿌리개를 나무가 없는 토양에 사용할 수 없다.
- 두 물뿌리개를 한 나무에 사용하여 3만큼 키울 수도 있다.

제한 사항

- 뒷뜰에 심은 사과나무 개수 $1 \leq N \leq 100{,}000$
- 바라는 나무의 높이 $1 \leq h_i \leq 10{,}000$

도전 문제

#알튜비튜

/<> 19539번 : 사과나무 - Gold 5

문제 분석

- 각 나무는 한 번에 1 또는 2 또는 3만큼 성장할 수 있다.

- 성장 순서는 결과에 영향을 미치지 않는다. → 탐욕적 선택 속성
- 모든 양의 정수는 1과 2의 배수의 합으로 표현할 수 있다. → 부분 구조

1짜리 물뿌리개가 남아도, 1+1=2이므로 2의 높이는 언제든 만들 수 있다
→ 2짜리 물뿌리개부터 사용하고, 남는 높이는 1짜리 물뿌리개로 해결하자!

예제

19539번 : 사과나무 - Gold 5

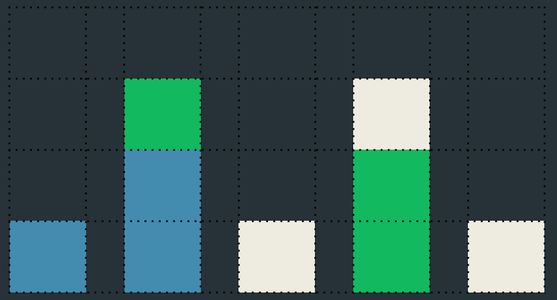
예제 입력1

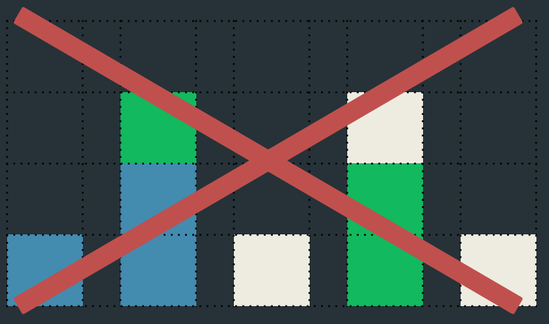
```
5
1 3 1 3 1
```

예제 출력1

```
NO
```

① 

② 

③ 

더 이상 2칸짜리 물뿌리개를 사용할 수 없다

도전 문제

2짜리 물뿌리개부터 사용하고, 남는 높이는 1짜리 물뿌리개로 해결하자!

문제를 더 단순화시킬 수 있지 않을까?

- 한 번 물을 줄 때마다 성장량의 합은 3이다.
- 물을 주는 횟수 k = (전체 나무 높이의 합)/3

  → 전체 나무 높이의 합은 3의 배수여야 한다.

- 합이 홀수가 되려면 홀수가 반드시 필요하므로 홀수 높이 개수에는 한계가 존재한다

  (홀+홀=짝, 짝+짝=홀, 짝+홀=홀)

  → 가능한 홀수 높이 개수의 최댓값 = k

예제

19539번 : 사과나무 – Gold 5

예제 입력1

```
5
1 3 1 3 1
```

예제 출력1

```
NO
```

(1+3+1+3+1) %3 == 0 → 나무 높이의 합 조건 만족

k = (1+3+1+3+1)/3 → k =3
홀수 나무 높이: 5개 → 5 > k 이므로 만족하지 않음

예제

19539번 : 사과나무 - Gold 5

예제 입력2

```
3
10000 1000 100
```

예제 출력2

```
YES
```

(10000+1000+100) %3 == 0 → 나무 높이의 합 조건 만족

k = (10000+1000+100)/3 → k = 3700
홀수 나무 높이: 0개 → 0 <= k 이므로 만족

도전 문제

프로그래머스 : 큰 수 만들기 - Lv.2

문제

- 어떤 숫자에서 k개의 수를 제거했을 때 얻을 수 있는 가장 큰 숫자를 구하기
- e.g. 숫자 1924에서 수 두 개를 제거하면 [19, 12, 14, 92, 94, 24]
  → 가장 큰 수: 94

제한 사항

- number는 2자리 이상, 1,000,000자리 이하인 숫자
- k는 1 이상, number의 자릿수 미만인 자연수

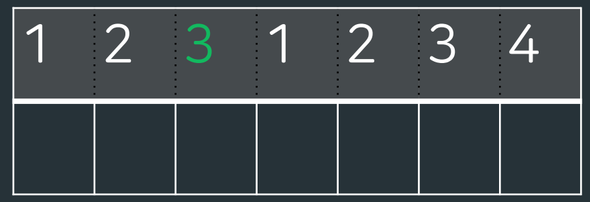
도전 문제

프로그래머스 : 큰 수 만들기 - Lv.2

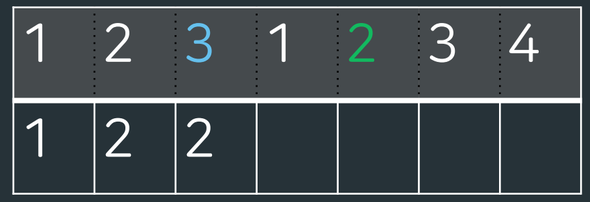
예제

number	k	return
“1231234”	3	“3234”

빠지는 숫자가 3개이므로, 남는 숫자는 4개
리턴하는 값의 앞자리부터 최대한 크게 만들어야 한다.



4칸 범위 내(=뺄 수 있는 숫자가 남은 범위)에서 최대값: 3 → 선택



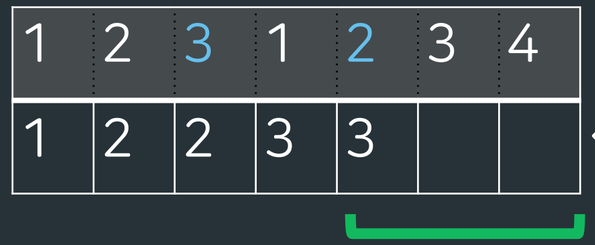
→ 기존 선택한 숫자의 앞은 선택해서는 안 된다. (최댓값 보장 안됨)
3칸 범위 내(=뺄 수 있는 숫자가 남은 범위)에서 최대값: 2 → 선택

도전 문제

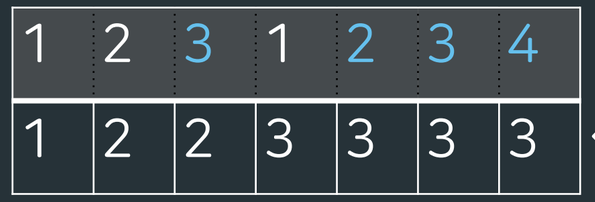
프로그래머스 : 큰 수 만들기 - Lv.2

예제

number	k	return
“1231234”	3	“3234”



→ 기존 선택한 숫자의 앞은 선택해서는 안 된다. (최댓값 보장 안됨)
더 이상 뺄 수 있는 숫자가 없다 → 모두 선택



프로그래머스 : 큰 수 만들기 - Lv.2

문제 분석

- (문자열 길이 - k)개 만큼의 숫자를 합쳐 리턴해주면 된다.
- 앞에서부터 시작하여, [뺄 수 있는 숫자가 남아 있는 범위 안]에서 최댓값을 선택한다.

- 앞에서부터 시작하는 이유: 높은 자리 숫자가 클 수록 결과값이 커진다. → 탐욕적 선택
- 뺄 수 있는 숫자가 남아 있는 범위 안: 뺄 수 있는 숫자의 개수 제한 → 부분 구조

/<> 15662번 : 톱니바퀴 (2) - Gold 5

문제

- 톱니바퀴를 총 K번 회전시키는 문제
- 톱니바퀴 A를 회전할 때, 톱니바퀴 B와 서로 맞닿은 톱니의 극이 다르다면, B는 A가 회전한 방향과 반대로 회전
- K개 줄에 회전시킨 방법이 존재 → 각 방법은 두 개의 정수로 이루어져 있음 (회전시킬 톱니바퀴 번호, 방향)

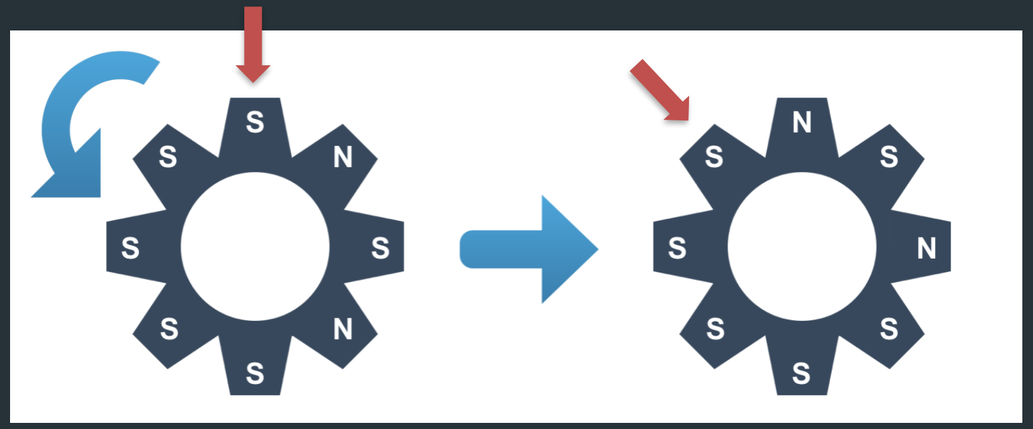
제한 사항

- 톱니바퀴의 개수 T(1<=T<=1000)
- 회전 횟수 K(1<=K<=1000)

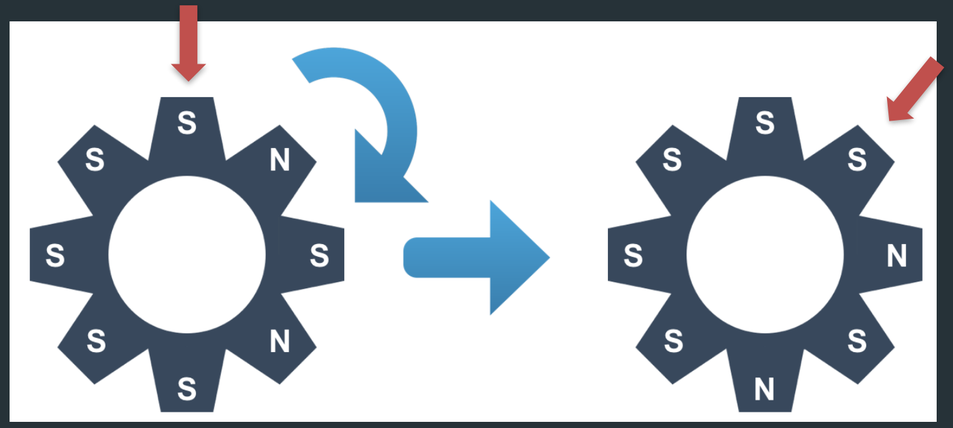
구현 문제

#알튜비튜

- 반시계방향



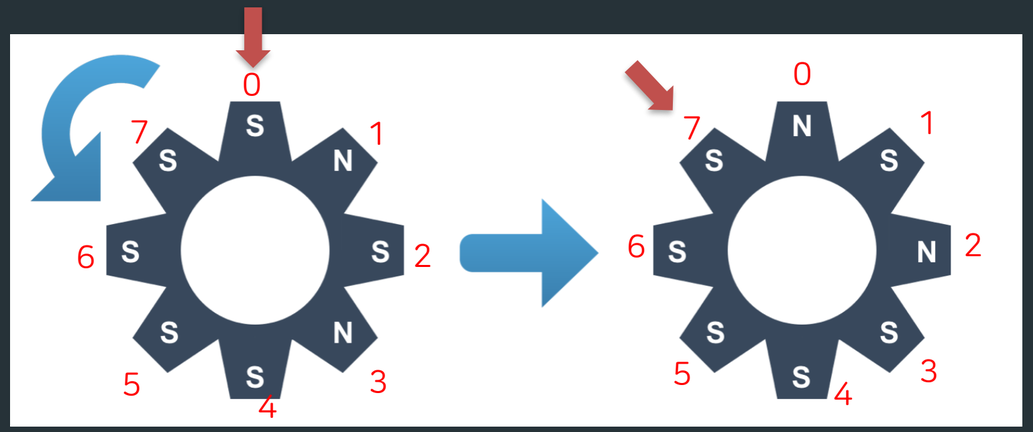
- 시계방향



구현 문제

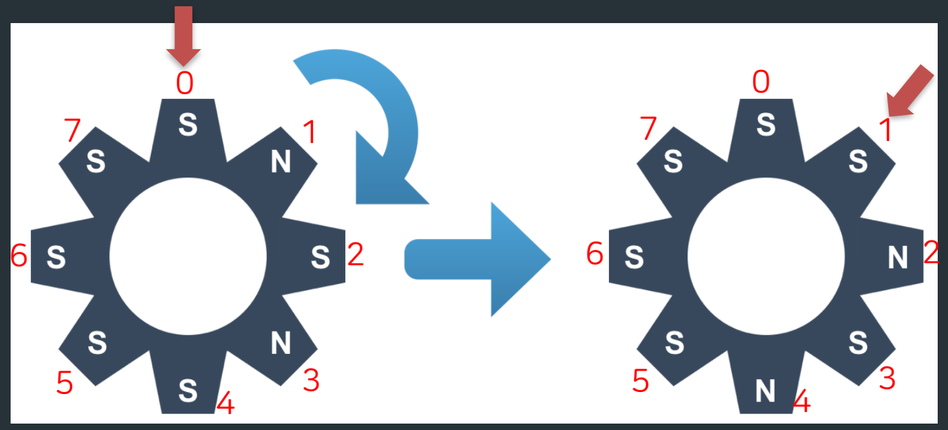
톱니바퀴 상태 입력은 12시 방향부터 시계 방향 순서로 주어짐

- 반시계방향



0번 인덱스 → 7번 인덱스

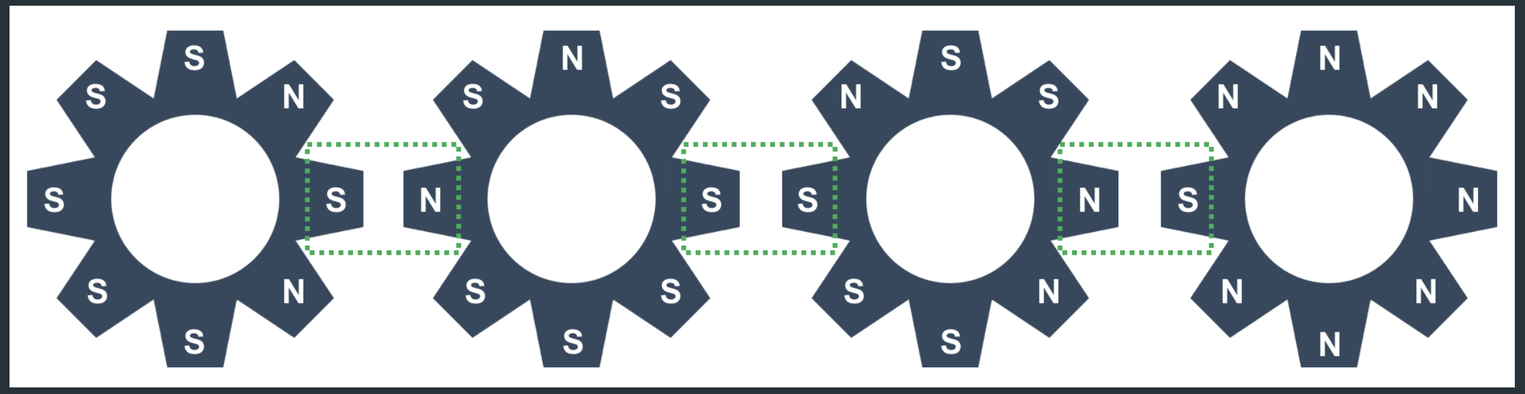
- 시계방향



0번 인덱스 → 1번 인덱스

예제

예제

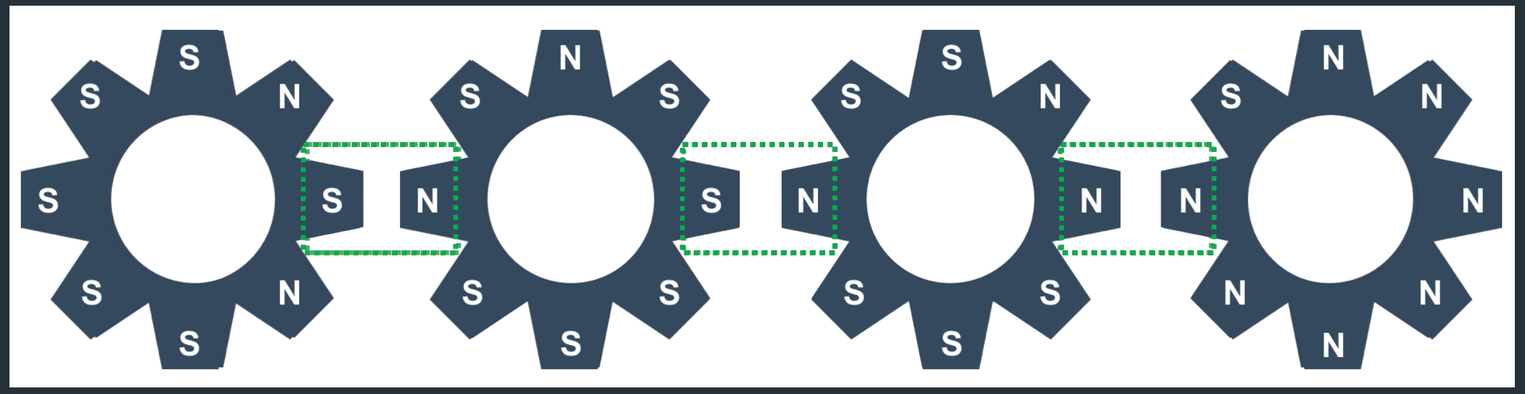


3번 톱니바퀴가 처음으로 반시계 방향으로 회전

3,4번이 회전

예제

예제

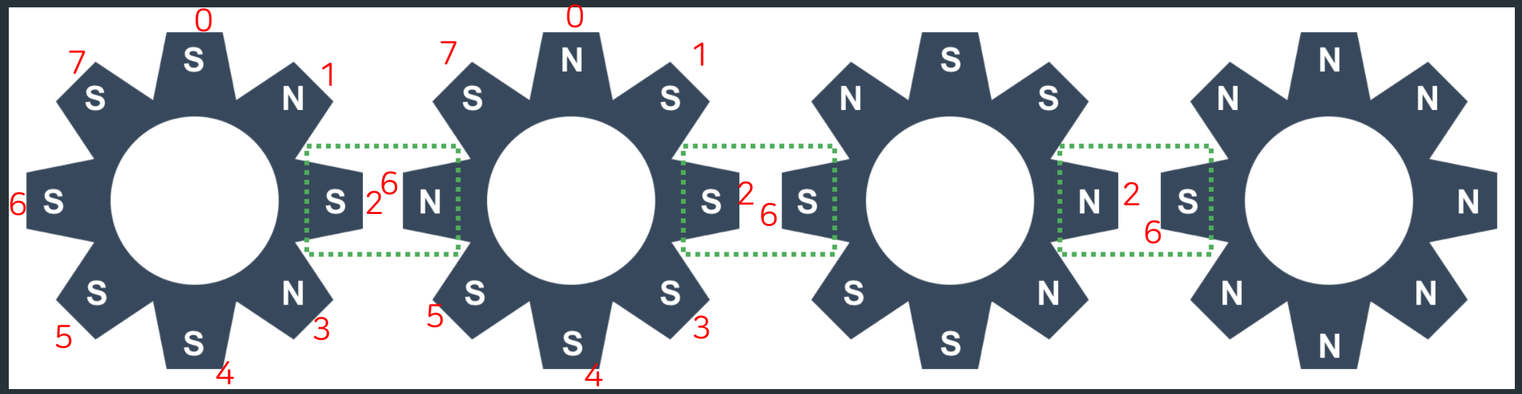


1번 톱니바퀴가 처음으로 시계 방향으로 회전

1,2,3번이 회전

풀이

풀이



- 2번과 6번 인덱스만 검사하면 됨
- 1번 톱니바퀴의 2번 인덱스 && 2번 톱니바퀴의 6번 인덱스
- 2번 톱니바퀴의 2번 인덱스 && 3번 톱니바퀴의 2번 인덱스

코드


```cpp
for (int i = 0; i < k; ++i) { //k번 돌면서 회전할 수 있는지에 대해 체크
        int num, direction;
        cin >> num >> direction; //방향 입력 (1이면 시계 방향, -1이면 반시계 방향)
        stack.clear();
        stack.push_back(make_pair(num - 1, direction));
        visited.assign(t, 0); //t개만큼 0을 초기화
        visited[num - 1] = 1;
        check_rotation(num - 1, direction);

        while (!stack.empty()) {
            pair<int, int> top = stack.back();
            stack.pop_back();
            int tmp_num = top.first; //돌려야 할 톱니바퀴의 인덱스
            int tmp_dir = top.second; //돌릴 방향
            if (tmp_dir == 1) { //시계방향으로 돌리기
                li[tmp_num] = li[tmp_num].back() + li[tmp_num].substr(0, 7);
            } else { //반시계방향으로 돌리기
                li[tmp_num] = li[tmp_num].substr(1) + li[tmp_num][0];
            }
        }
    }
```

코드

#알튜비튜

```
void check_rotation(int current_num, int current_dir) {
    if (current_num - 1 >= 0 && visited[current_num - 1] == 0) {
        if (li[current_num][6] != li[current_num - 1][2]) { // 맞닿아 있는 면이 서로 다른 극인지 체
크
            stack.push_back(make_pair(current_num - 1, current_dir * -1));
            visited[current_num - 1] = 1;
            check_rotation(current_num - 1, current_dir * -1);
        }
    }
    if (current_num + 1 < li.size() && visited[current_num + 1] == 0) {
        if (li[current_num][2] != li[current_num + 1][6]) { // 맞닿아 있는 면이 서로 다른 극인지 체
크
            stack.push_back(make_pair(current_num + 1, current_dir * -1));
            visited[current_num + 1] = 1;
            check_rotation(current_num + 1, current_dir * -1);
        }
    }
}
```

마무리

#알튜비튜

추가로 풀어보면 좋은 문제!

- 10610번 : 30 – Silver 4
- 16206번 : 롤케이크 – Silver 1
- 1339번 : 단어 수학 – Gold 4
- 2437번 : 저울 – Gold 2