

알튜비튜 그리디



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

/<> 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 $1 \leq h_i \leq 10,000$

/<> 19539번 : 사과나무 - Gold 5

문제 분석

- 각 나무는 한 번에 1 또는 2 또는 3만큼 성장할 수 있다.
- 성장 순서는 결과에 영향을 미치지 않는다. \Rightarrow 탐욕적 선택 속성
- 모든 양의 정수는 1과 2의 배수의 합으로 표현할 수 있다. \Rightarrow 부분 구조

1짜리 물뿌리개가 남아도, $1+1=2$ 이므로 2의 높이는 언제든지 만들 수 있다
 \Rightarrow 2짜리 물뿌리개부터 사용하고, 남은 높이는 1짜리 물뿌리개로 해결하자!

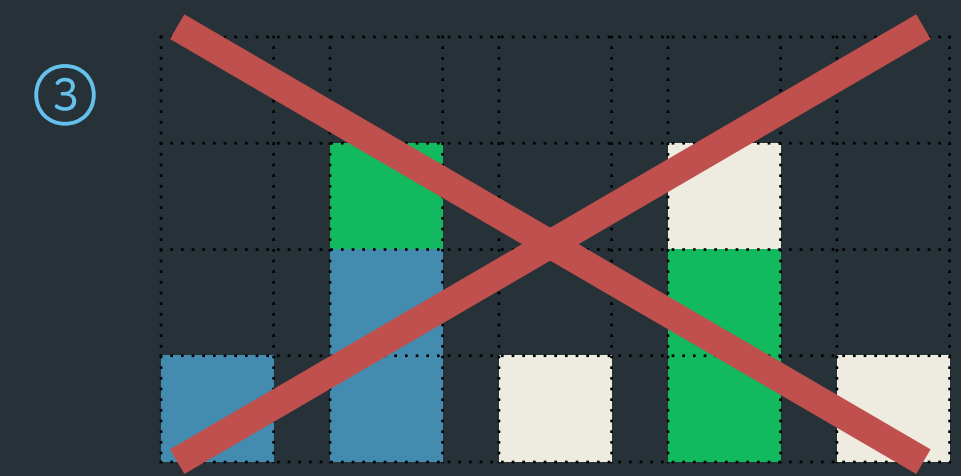
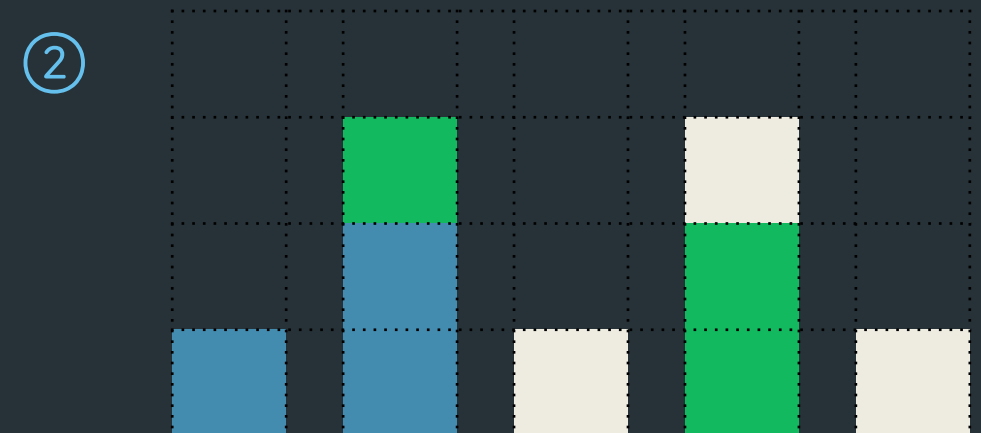
<> 19539번 : 사과나무 - Gold 5

예제 입력1

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

예제 출력1

```
NO
```



더 이상 2칸짜리 물뿌리개를 사용할 수 없다

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

문제를 더 단순화시킬 수 있지 않을까?

- 한 번 물을 줄 때마다 성장량의 합은 3이다.
- 물을 주는 횟수 $k = (\text{전체 나무 높이의 합})/3$
→ 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

/<> 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

/<> 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 → $0 \leq k$ 이므로 만족

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

프로그래머스 : 큰 수 만들기 - Lv.2

예제

number	k	return
"1231234"	3	"3234"

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

1	2	3	1	2	3	4

← 빠진 숫자 개수

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

1	2	3	1	2	3	4
1	2	2				

← 빠진 숫자 개수

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

프로그래머스 : 큰 수 만들기 - Lv.2

예제

number	k	return
"1231234"	3	"3234"

1	2	3	1	2	3	4
1	2	2	3	3		

← 빠진 숫자 개수

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

1	2	3	1	2	3	4
1	2	2	3	3	3	3

← 빠진 숫자 개수

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

/<> 18111번 : 마인크래프트 - Silver 2

문제

- 다음 두 종류의 작업을 통해 모든 칸의 높이를 동일하게 만들어야 함
- 1. (i, j)에서 블록을 제거하여 인벤토리에 넣기 - 2초 소요
- 2. 인벤토리에서 블록을 꺼내 (i, j)에 쌓기 - 1초 소요
- 땅 고르기 작업에 걸리는 최소 시간과 그 때의 땅의 높이를 구하는 문제

제한 사항

- 땅의 가로, 세로: $1 \leq M, N \leq 500$
- 초기 인벤토리에 들어있는 블록 개수 B: $0 \leq B \leq 6.4 * 10^7$
- 초기 땅의 높이 : 256보다 작거나 같은 자연수 또는 0

예제 입력

```
3 4 99
0 0 0 0
0 0 0 0
0 0 0 1
```

예제 출력

```
2 0
```

문제

- 다음 두 종류의 작업을 통해 모든 칸의 높이를 동일하게 만들어야 함
- 1. (i, j) 에서 블록을 제거하여 인벤토리에 넣기 - 2초 소요
- 2. 인벤토리에서 블록을 꺼내 (i, j) 에 쌓기 - 1초 소요
- 땅 고르기 작업에 걸리는 최소 시간과 그 때의 땅의 높이를 구하는 문제

접근

- 땅의 높이 범위가 매우 작음 (0 ~ 256)
→ 모든 높이로 다 만들어보고 걸리는 시간이 최소일 때를 찾으면 되지 않을까?

- 모든 칸을 높이 h 로 만드는 것이 불가능할 수도 있나?
→ 블록이 모자라는 경우

- h 보다 높은 칸을 먼저 깎아내서 인벤토리에 최대한 많은 블록을 확보해두자.
→ 인벤토리 내 블록 수: $B + \text{removed}$
- 이제 h 보다 낮은 칸들에 블록을 추가해주자.
→ 추가해야 하는 블록 수: added
- 만약 $\text{added} > (B + \text{removed})$ 라면
→ 블록 부족, 불가능

추가로 풀어보면 좋은 문제!

/<> 10610번 : 30 – Silver 4

/<> 16206번 : 롤케이크 – Silver 1

/<> 1339번 : 단어 수학 – Gold 4

/<> 2437번 : 저울 – Gold 2