1. [2023 넥토리얼 1] 천공카드 입력장치의 자동완성

천공카드는 0과 1을 구멍을 뚫었는지 여부를 통해 저장하는 매체이다.
넥돌이는 바이너리 문자열을 입력하면 천공카드에 기록해 주는 입력장치 시스템을 개발하고 있다.
바이너리 문자열은 입력할 때 실수하기 쉽기 때문에 입력 편의를 위해 입력 도우미 기능을 추가하려고 한다.

도우미 기능은 다음과 같이 동작한다.
문자를 하나 입력할 때마다 기존 입력 기록을 기반으로 자동완성 시켜줄 수 있는 문자열을 보여준다.
만약 자동 완성해 줄 수 있는 기록이 여럿이라면 가장 최근의 것으로 보여준다.
자동 완성해 줄 수 있는 기록이 없다면 바로 직전의 문자열을 보여준다.
이전 기록이 하나도 없다면 빈 문자열을 보여준다.

구현 내용
도우미 기능을 테스트할 함수 autocompletes를 구현한다

autocompletes는 다음과 같은 인자를 입력 받는다:
string inputs[n]: 입력할 문자열의 목록. i번째 문자열은 i-1번째까지의 문자열을 이전 입력 기록으로 간주한다.

autocompletes는 다음과 같은 값을 반환한다:
int[n]: 각 문자열을 입력할 때 도우미 기능이 마지막으로 보여준 문자열의 입력 번호
* 입력 번호는 1부터 시작하기 때문에 inputs의 index + 1 이다
** 이전 기록이 하나도 없어 빈 문자열을 보여주는 경우 0으로 기록한다

제약 조건

- $2 \le n \le 10^5$
- $1 \le$ input 1개의 길이 ≤ 30

▼ 예제1

입력

```
inputs = ['100110', '1001', '1001111']
```

출력

```
[0, 1, 1]
```

설명
1. 100110 - 0 (이전 기록이 없음)
2. 1001 - 1 (전부 입력했을 때 1번이 자동완성)
3. 1001111 - 1 (10011 까지 입력했을 때 1번이 자동완성)

▼ 예제2

입력

```
inputs = ['000', '1110', '01', '001', '110', '11']
```

출력

```
[0, 1, 1, 1, 2, 5]
```

설명
1. 000 - 0 (이전 기록이 없음)
2. 1110 - 1 (자동 완성 시켜줄 기록이 없어 가장 최근 문자열을 자동완성)
3. 01 - 1 (0 을 입력했을 때 1번이 자동 완성)
4. 001 - 1 (00 을 입력했을 때 1번이 자동 완성)
5. 110 - 2 (11을 입력했을 때 2번이 자동 완성)
6. 11 - 5 (11을 입력했을 때 5번이 자동 완성)

2. [2023 넥토리얼 2] 미니 카트라이더

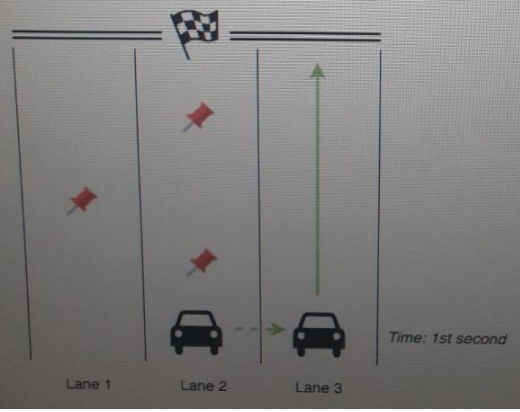
연동이는 미니 카트라이더를 플레이 중 입니다. 미니 카트라이더에서 카트 조작은 좌우 방향만 조작하며 카트는 앞으로 자동으로 전진 합니다. 현재 플레이 중인 게임 트랙에서는 3개의 차선이 존재하며 카트는 하나의 차선으로만 달릴 수 있습니다. 트랙에는 장애물들이 존재하는데, 카트는 좌우 이동을 통해 장애물을 피하는 것은 가능하지만 장애물이 있는 위치로 이동은 불가능 합니다

예를 들어 카트가 1차선에 있고 2차선에 장애물이 있는경우 3차선으로 이동하는 것은 가능하지만 2차선으로 이동하는 것은 불가능합니다. 시작 시 카트는 가운데 차선인 2차선에서 시작하며 최소의 좌우 이동 조작으로 골인에 도달 해야 합니다. 1차선에서 2차선으로 또는 1차선에서 3차선으로 이동하는 것은 각각 1번의 이동으로 간주합니다

예제

n = 3
obstacleLanes = [2, 1, 2]

연동이는 아래 예제에서 첫 조작으로 3차선으로 이동합니다. 이후 추가 조작 없이 골에 도달 합니다. 최소로 요구되는 조작은 1회 입니다.



함수 설명

함수 *minimumMovement* 를 작성하세요

minimumMovement 는 다음의 파라미터를 가진다:
int obstacleLanes[n]: 각 칸 별 장애물의 차선 위치

반환값

int: 연동이가 골인에 도착 할 수 있는 최소 좌우 여동 횟수

제약조건

- $1 \le n \le 10^5$
- $1 \le obstacleLanes[i] \le 3 \ (0 \le i < n)$
- 모든 레이스는 골인에 도착이 보장 됩니다.

▼ 예제 1

입력 값

```
[2, 1, 2]
```

출력 값

```
1
```

설명
처음에 3차선으로 이동하면 더이상 이동이 필요하지 않습니다. 자세한 사항은 상단의 그림을 참고하세요.

▼ 예제 2

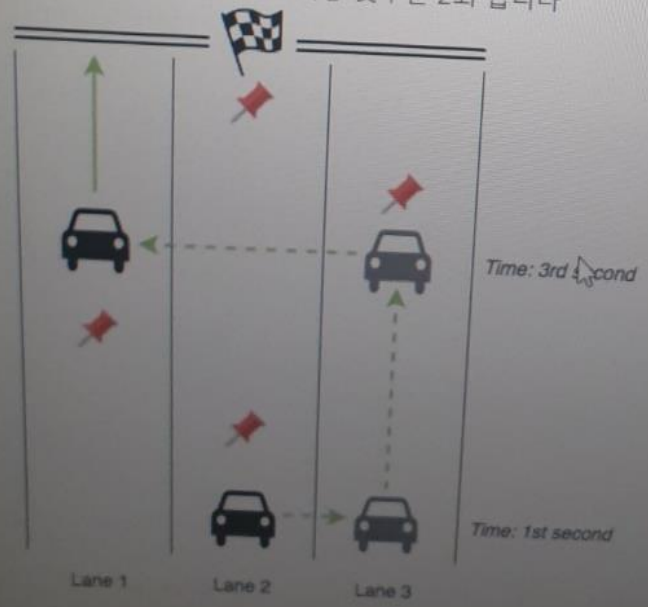
입력 값

```
[2, 1, 3, 2]
```

출력 값

```
2
```

설명
연동이는 처음에 3차선으로 이동 후 3번째 장애물 도달 전에 1차선으로 이동 합니다. 총 차선 이동 횟수는 2회 입니다



3. [2023 넥토리얼 3] 선긋기 귀찮아

지웅이는 수학 시간에 선생님이 주신 좌표대로 선분을 그리거나 점을 찍어야 한다. 그런데 선생님이 주신 좌표 리스트를 보니 불필요한 선들이 보인다. 불필요한 선들을 정리하고 작은 숫자부터 큰 숫자로 정리해서 최소한의 노력으로 선을 긋게 지웅이를 도와주자

예제

선분에 대한 정보는 [A, B] 형태로 주어지며 여러개의 선분의 리스트가 주어진다

lineSegments = [[15,15], [7, 7], [2, 3], [6, 11], [1, 2]]

선분 [1, 2] 와 [2, 3]의 경우 합쳐서 [1, 3] 으로 그리기가 가능하다 선분 [7, 7]의 경우 선분 [6, 11] 에 포함이 된다.
[15, 15]는 독립되어 점으로 찍어야 하므로 더 이상 겹쳐지는 부분이 없는 최소 선분 묶음은 [[1,3], [6, 11], [15,15]]

함수 설명

함수 getMergedLineSegments를 구현하라

getMergedLineSegments는 다음의 파라미터를 가진다

int lineSegments[[n][2]: 두점으로 이루어진 n개의 선분

반환값

int[][2]: 순서대로 정리된 최소 묶음의 선분들

제약조건

- $1 \le n \le 10^5$
- $1 \le lineSegments[i][2] \le 10^9$
- 모든 i에 대해서 $lineSegments[i][0] \le lineSegments[i][1]$

▼ 예제 1

입력 값

```
[[6, 9], [2, 3], [9, 11], [1, 5], [14, 18]]
```

출력 값

```
[[1, 5], [6, 11], [14, 18]]
```

설명

선분 [2, 3] 는 [1, 5] 에 포함 된다. 그리고 [6, 9] 와 [9, 11] 은 [6, 11] 로 합쳐진다. 최종 결과는 [1, 5], [6, 11], [14, 18].

▼ 예제 2

입력 값

```
[[4, 8],[2, 6],[5, 7]]
```

출력 값

```
{{}}
```

설명

세 선분은 [2, 8] 로 정리 된다.

4. [2023 넥토리얼 4] 발전소 신입사원 지혜 도와주기

첨단 발전소에 신입 사원으로 입사한 지혜는 사람들과 이야기 하다가 현재 하는 작업을 자동화 할 수 있다고 실수로 큰 소리 치고 말았습니다.

이 첨단 입자 발전소의 발전기는 에너지를 가지고 있는 입자 묶음을 발전기에 주입하면, 입자들의 에너지 총 합이 임계치 이상으로 유지 되는 한 발전기가 가동 됩니다.

따라서 지혜는 입자마다 고유의 에너지를 가지고 있는 에너지 입자들의 묶음을 발전기 속에 넣어서 묶음의 총 에너지 값이 임계치 미만으로 떨어지기 전에 에너지 입자 묶음을 자동으로 제거 하는 시스템을 개발하고자 합니다.

에너지 입자 묶음 속에 들어 있는 각각의 에너지 입자들은 시간이 1초 지날 때 마다 1 만큼의 에너지를 잃게 됩니다. 각 입자의 현재 에너지 값은 max(initialEnergy[i] − sec_time, 0) 입니다.

각각의 입자는 매 초마다 에너지를 잃고, 0 이하의 값으로도 떨어지기는 합니다. 하지만 0 이하일 경우 발전에 영향을 주지 않기 때문에 계산 대상이 아닙니다.

찾고자 하는 값은 입자 묶음이 발전기 속으로 들어가서 에너지를 잃기 시작 한 순간부터 주어지는 임계값(th) 보다 크거나 같은 상태를 최대한 유지 할 수 있는 발전기 속 최대 체류 시간입니다.

예제

예를 들어서 5개의 에너지 입자를 가진 하나의 묶음[4,8,7,1,2]이 있고 ,주어지는 임계값은 9 라고 하겠습니다.
n=5, initialEnergy = [4, 8, 7, 1, 2] , th = 9

매 초 시간이 흐름에 따라 최종 에너지 값이 주어진 임계값 보다 낮아지는 순간을 계산해 보면 다음과 같습니다.

진행 시간	에너지 묶음의 상태	에너지 총 합
0	4 8 7 1 2	22
1	3 7 6 0 1	17
2	2 6 5 -1 0	13
3	1 5 4 -2 -1	10
4	0 4 3 -3 -2	7

각 입자의 현재 에너지 값은 초기 값을 기반으로 계산 되어 집니다. 각 입자 에너지 값은 max(current value, 0) 이고, max 값을 이용해서 계산 하고 있으므로, 0 보다 작은 음수는 에너지의 총 합에 계산 되지 않습니다.

예제 에너지 묶음의 에너지 총 합이 임계값(9) 보다 낮아지기 직전의 시간 값은 총 합이 10 이었던 3 입니다.

GetMaxTime을 완성 해서 지혜를 도와주세요.

함수 설명

함수 **GetMaxTime** 을 구현하라

GetMaxTime 함수는 다음의 파라미터를 가지고 있습니다.
int initialEnergy[n]: 에너지 초기 값
int th: 임계치

반환값

에너지의 총 합이 임계치 보다 같거나 높은 상태를 유지 할 수 있는 최대 시간 integer 값

제약조건

- $2 \le n \le 10^5$
- $1 \le initialEnergy[i] \le 10^9$
- $1 \le th \le 10^{14}$
- 시간의 값은 음수를 가질 수 없습니다

5. [2023 넥토리얼 5] 삼형제의 숫자 놀이

삼형제가 엄마와 숫자 카드 놀이를 하고 있다.
엄마가 정해진 숫자(t) 와 함께 숫자 카드(d[n])을 제시하면
동생부터 카드를 한장씩 골라 세장을 더했을 때 t를 넘겨야 한다.
다만 형은 언제나 동생이 선택한 것보다 큰 숫자 카드를 골라야 한다
이 때 삼형제가 만들 수 있는 조합은 몇가지인지 구하려고 한다.

구현 내용
함수 three_numbers 를 만듭니다.

three_numbers 는 다음과 같은 인자를 받습니다:
int t: 엄마가 제시한 숫자
int d[n]: 숫자 카드들

three_numbers 는 다음과 같은 값을 반환합니다:
long: 조건을 만족하는 조합의 가지 수

제약 조건

- $1 \le n \le 10^4$
- $0 \le d[i] < 10^9$
- $0 < t < 3 \times 10^9$

▼ 예제1

입력

- $0 < t < 3 \times 10^9$

▼ 예제1

입력

```
t = 8
d = [1, 2, 3, 4, 5]
```

출력

```
4
```

설명
다음과 같은 4개의 조합이 가능하다.

(1,2,3) → 1 + 2 + 3 = 6 ≤ 8
(1,2,4) → 1 + 2 + 4 = 7 ≤ 8
(1,2,5) → 1 + 2 + 5 = 8 ≤ 8
(1,3,4) → 1 + 3 + 4 = 8 ≤ 8

▼ 예제2

입력

```
t = 7
d = [3, 1, 2, 4]
```

출력

```
2
```

설명
다음과 같은 2개의 조합이 가능하다.

(1, 2, 3) → 1 + 2 + 3 ≤ 7
(1, 2, 4) → 1 + 2 + 4 ≤ 7