코딩역량 강화 과정 Java & SQL

2일차 - 1

장은실

0

코딩 역량 강화 1일차 되돌아 보기

1 배열과 리스트

2 문자열 처리

3 스택과 큐

4 정렬

1 배열

주식가격

<문제 설명>

초 단위로 기록된 주식가격이 담긴 배열 prices가 매개변수로 주어질 때, 가격이 떨어지지 않은 기간은 몇 초인지를 return 하도록 solution 함수를 완성하세요.

<제한 조건>

- prices의 각 가격은 1 이상 10,000 이하인 자연수입니다.
- prices의 길이는 2 이상 100,000 이하입니다.

<입출력 예>

prices	return
[1, 2, 3, 2, 3]	[4, 3, 1, 1, 0]

입출력 예 설명

- 1초 시점의 ₩1은 끝까지 가격이 떨어지지 않았습니다.
- 2초 시점의 ₩2은 끝까지 가격이 떨어지지 않았습니다.
- 3초 시점의 ₩3은 1초 뒤에 가격이 떨어집니다. 따라서 1초간 가격이 떨어지지 않은 것으로 봅니다.
- 4초 시점의 ₩2은 1초간 가격이 떨어지지 않았습니다.
- 5초 시점의 ₩3은 0초간 가격이 떨어지지 않았습니다.

2 문자열 처리

최댓값과 최솟값

<문제 설명>

문자열 s에는 공백으로 구분된 숫자들이 저장되어 있습니다. str에 나타나는 숫자 중 최소값과 최대값을 찾아 이를 "(최소값) (최대값)"형태의 문자열을 반환하는 함수, solution을 완성하세요.

예를들어 s가 "1 2 3 4"라면 "1 4"를 리턴하고, "-1 -2 -3 -4"라면 "-4 -1"을 리턴하면 됩니다.

<입출력 예>

s	return
"1 2 3 4"	"1 4"
"-1 -2 -3 -4"	"-4 -1"
"-1 -1"	"-1 -1"

<제한 조건>

- s에는 둘 이상의 정수가 공백으로 구분되어 있습니다.

3 스택

올바른 괄호

<문제 설명>

괄호가 바르게 짝지어졌다는 것은 '(' 문자로 열렸으면 반드시 짝지어서 ')' 문자로 닫혀야 한다는 뜻입니다. 예를 들어

- "()()" 또는 "(())()" 는 올바른 괄호입니다.
- ")()(" 또는 "(()(" 는 올바르지 않은 괄호입니다.

'(' 또는 ')' 로만 이루어진 문자열 s가 주어졌을 때, 문자열 s가 올바른 괄호이면 true를 return 하고, 올바르지 않은 괄호이면 false를 return 하는 solution 함수를 완성해 주세요.

<제한 조건>

- 문자열 s의 길이 : 100,000 이하의 자연수
- 문자열 s는 '(' 또는 ')' 로만 이루어져 있습니다.

<입출력 예>

s	answer
"()()"	true
"(())()"	true
")()("	false
"(()("	false

4-1 정렬

배열(Array) 정렬

오름차순 정렬 : Arrays.sort(배열명)

내림차순 정렬 : Arrays.sort(배열명, Comparator.reverseOrder()) 또는 **Arrays.sort(배열명, Collections.reverseOrder())**

- byte, char, int, short, double, float, long 의 기본 타입 배열은 내림차순 정렬 안되기 때문에 Interger같은 wrapper class 적용

```
Integer aa[] = {1,2,3,4,5};
    Arrays.sort(aa, Collections.reverseOrder());
```

리스트(ArrayList) 정렬

오름차순 정렬 : Collections.sort(리스트명)

내림차순 정렬 : Collections.reverse(리스트명)

4-2 정렬과 Comparable / Comparator

- **객체를 정렬하는데 필요한 메서드를 정의한 인터페이스(정렬 기준 제공)**
  - Comparable : 기본 정렬 기준을 구현하는데 사용

```
public interface Cpmparable {
    int compareTo (Object o);              // 주어진 객체(o)를 자신과 비교
}
```

  - Comparator : 기본 정렬 기준 외에 다른 기준으로 정렬하고자 할 때 사용

```
public interface Cpmparator {
    int compare (Object o1, Object o2);  // o1, o2 두 객체를 비교
    boolean equals (Object obj);         // equals 를 오버라이딩 하라는 의미
}
```

  - compare()와 compareTo()는 두 객체의 비교 결과를 반환하도록 작성
    - 같으면 0, 오른쪽이 크면 음수(-), 작으면 양수(+)

```
Ex) Arrays.sort(arr, (a, b) -> (a+b).compareTo(b+a));      // 오름차순
```

4 정렬

가장 큰 수

<문제 설명>

0 또는 양의 정수가 주어졌을 때, 정수를 이어 붙여 만들 수 있는 가장 큰 수를 알아내 주세요.

예를 들어, 주어진 정수가 [6, 10, 2]라면 [6102, 6210, 1062, 1026, 2610, 2106]를 만들 수 있고, 이중 가장 큰 수는 6210입니다.

0 또는 양의 정수가 담긴 배열 numbers가 매개변수로 주어질 때, 순서를 재배치하여 만들 수 있는 가장 큰 수를 문자열로 바꾸어 return 하도록 solution 함수를 작성해주세요.

<제한 조건>

- numbers의 길이는 1 이상 100,000 이하입니다.
- numbers의 원소는 0 이상 1,000 이하입니다.
- 정답이 너무 클 수 있으니 문자열로 바꾸어 return 합니다.

<입출력 예>

numbers	return
[6, 10, 2]	"6210"
[3, 30, 34, 5, 9]	"9534330"

04

정수론

04-1 정수론

04-2 약수 및 소수

04-3 유클리디안 호제법

04 정수론

수학에서 정수론은 수의 성질을 탐구하고 공부하는 분야를 뜻한다.

실제 코딩 테스트에서는 정수론 영역에서 소수 부분과 유클리디안 호제법 부분이 자주 등장한다.

소수

소수(prime number)는 자신보다 작은 2개의 자연수를 곱해 만들 수 없는 1보다 큰 자연수를 말한다.

같은 의미로 1과 자기 자신 외에 약수가 존재하지 않는 수를 말한다.

유클리드 호제법

유클리드 호제법(Educlidean-algorithm)은 두 수의 최대 공약수를 구하는 알고리즘이다.

일반적으로 소인수분해를 이용하여 최대 공약수를 구하지만 유클리디안 호제법을 이용하면 좀 더 간단하게 구할 수 있다.

04 정수론

1. 약수의 합

<문제 설명>

정수 n을 입력받아 n의 약수를 모두 더한 값을 리턴하는 함수, solution을 완성해주세요.

<제한 조건>

- n은 0 이상 3000이하인 정수입니다.

<입출력 예>

n	return
12	28
5	6

입출력 예#1

12의 약수는 1, 2, 3, 4, 6, 12입니다. 이를 모두 더하면 28입니다.

입출력 예 #2

5의 약수는 1, 5입니다. 이를 모두 더하면 6입니다.

04 정수론

2. 약수의 개수와 덧셈

<문제 설명>

두 정수 left와 right가 매개변수로 주어집니다. left부터 right까지의 모든 수들 중에서, 약수의 개수가 짝수인 수는 더하고, 약수의 개수가 홀수인 수는 뺀 수를 return 하도록 solution 함수를 완성해주세요.

<제한 조건>

- $1 \le left \le right \le 1,000$

< 입출력 예 >

left	right	result
13	17	43
24	27	52

<입출력 예>

수	약수	약수의 개수
13	1, 13	2
14	1, 2, 7, 14	4
15	1, 3, 5, 15	4
16	1, 2, 4, 8, 16	5
17	1, 17	2

수	약수	약수의 개수
24	1, 2, 3, 4, 6, 8, 12, 24	8
25	1, 5, 25	3
26	1, 2, 13, 26	4
27	1, 3, 9, 27	4

입출력 예#1

13 + 14 + 15 - 16 + 17 = 43을 return

입출력 예 #2

24 - 25 + 26 + 27 = 52를 return

04 정수론

3. 소수 찾기

<문제 설명>

1부터 입력받은 숫자 n 사이에 있는 소수의 개수를 반환하는 함수, solution을 만들어 보세요.

소수는 1과 자기 자신으로만 나누어지는 수를 의미합니다. (1은 소수가 아닙니다.)

<제한 조건>

- n은 2이상 1000000이하의 자연수입니다.

<입출력 예>

n	result
10	4
3	3

입출력 예#1

1부터 10 사이의 소수는 [2,3,5,7] 4개가 존재하므로 4를 반환

입출력 예 #2

1부터 5 사이의 소수는 [2,3,5] 3개가 존재하므로 3를 반환

04 정수론

4. 소수 만들기

<문제 설명>

주어진 숫자 중 3개의 수를 더했을 때 소수가 되는 경우의 개수를 구하려고 합니다. 숫자들이 들어있는 배열 nums가 매개변수로 주어질 때, nums에 있는 숫자들 중 서로 다른 3개를 골라 더했을 때 소수가 되는 경우의 개수를 return 하도록 solution 함수를 완성해주세요.

<제한 조건>

- nums에 들어있는 숫자의 개수는 3개 이상 50개 이하입니다.
- nums의 각 원소는 1 이상 1,000 이하의 자연수이며, 중복된 숫자가 들어있지 않습니다.

<입출력 예>

nums	result
[1,2,3,4]	1
[1,2,7,6,4]	4

입출력 예#1
[1,2,4]를 이용해서 7을 만들 수 있습니다.

입출력 예 #2
[1,2,4]를 이용해서 7을 만들 수 있습니다.
[1,4,6]을 이용해서 11을 만들 수 있습니다.
[2,4,7]을 이용해서 13을 만들 수 있습니다.
[4,6,7]을 이용해서 17을 만들 수 있습니다.

04 정수론

5. 하샤드 수

<문제 설명>

양의 정수 x가 하샤드 수이려면 x의 자릿수의 합으로 x가 나누어져야 합니다. 예를 들어 18의 자릿수 합은 1+8=9이고, 18은 9로 나누어 떨어지므로 18은 하샤드 수입니다. 자연수 x를 입력받아 x가 하샤드 수인지 아닌지 검사하는 함수, solution을 완성해주세요.

<제한 조건>

- x는 1 이상, 10000 이하인 정수입니다.

<입출력 예>

arr	return
10	true
12	true
11	false
13	false

입출력 예#1

10의 모든 자릿수의 합은 1입니다. 10은 1로 나누어 떨어지므로 10은 하샤드 수입니다.

입출력 예 #2

12의 모든 자릿수의 합은 3입니다. 12는 3으로 나누어 떨어지므로 12는 하샤드 수입니다.

입출력 예#3

11의 모든 자릿수의 합은 2입니다. 11은 2로 나누어 떨어지지 않으므로 11는 하샤드 수가 아닙니다.

입출력 예 #4

13의 모든 자릿수의 합은 4입니다. 13은 4로 나누어 떨어지지 않으므로 13은 하샤드 수가 아닙니다.

04 정수론

6. 콜라츠 추측

<문제 설명>

1937년 Collatz란 사람에 의해 제기된 이 추측은, 주어진 수가 1이 될 때까지 다음 작업을 반복하면, 모든 수를 1로 만들 수 있다는 추측이며, 작업은 다음과 같습니다.

```
1-1. 입력된 수가 짝수라면 2로 나눕니다.
1-2. 입력된 수가 홀수라면 3을 곱하고 1을 더합니다.
2. 결과로 나온 수에 같은 작업을 1이 될 때까지 반복합니다.
```

예를 들어, 주어진 수가 6이라면 6 → 3 → 10 → 5 → 16 → 8 → 4 → 2 → 1 이 되어 총 8번 만에 1이 됩니다. 위 작업을 몇 번이나 반복해야 하는지 반환하는 함수, solution을 완성해 주세요. 단, 주어진 수가 1인 경우에는 0을, 작업을 500번 반복할 때까지 1이 되지 않는다면 –1을 반환해 주세요

<제한 조건>

- 입력된 수, num은 1 이상 8,000,000 미만인 정수입니다.

<입출력 예>

n	result
6	8
16	4
626331	-1

입출력 예#2

16 → 8 → 4 → 2 → 1 이 되어 총 4번 만에 1이 됩니다.

입출력 예 #3

626331은 500번을 시도해도 1이 되지 못하므로 -1을 리턴해야 합니다.

04 정수론

7. 최대공약수와 최소공배수

<문제 설명>

두 수를 입력받아 두 수의 최대공약수와 최소공배수를 반환하는 함수, solution을 완성해 보세요. 배열의 맨 앞에 최대공약수, 그다음 최소공배수를 넣어 반환하면 됩니다. 예를 들어 두 수 3, 12의 최대공약수는 3, 최소공배수는 12이므로 solution(3, 12)는 [3, 12]를 반환해야 합니다.

<제한 조건>

- 두 수는 1이상 1000000이하의 자연수입니다.

<입출력 예>

n	m	return
3	12	[3, 12]
2	5	[1, 10]

입출력 예#2

자연수 2와 5의 최대공약수는 1, 최소공배수는 10이므로 [1, 10]을 리턴해야 합니다.

05

해시

05-1 해시 특징 및 주요 클래스

05-2 완주하지 못한 선수

05-3 포켓몬

05 해시 개념 및 주요 클래스

해시 개요

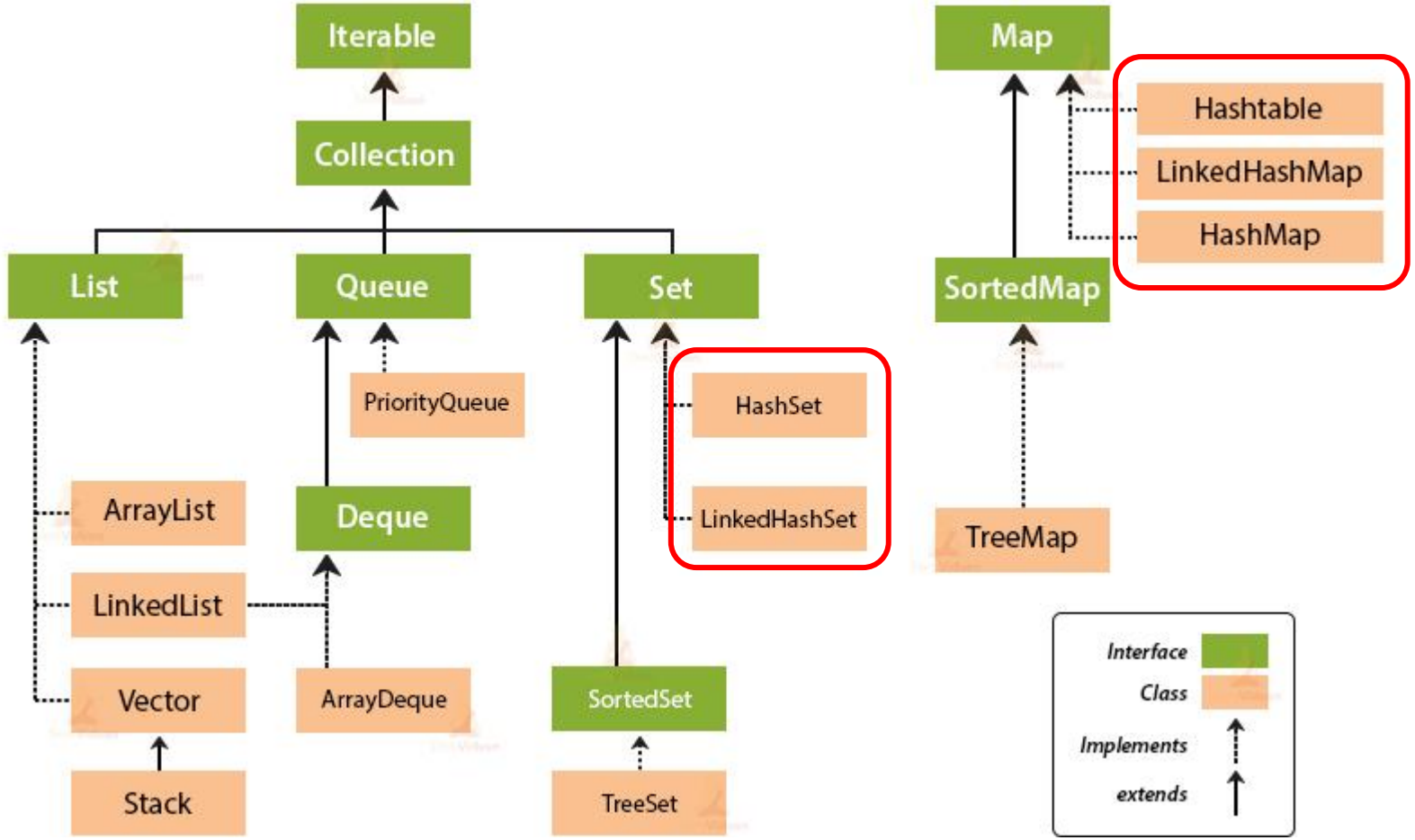
해시 개념

해시 함수 또는 해시 알고리즘 또는 해시 함수 알고리즘은 임의의 길이의 데이터를 고정된 길이의 데이터로 매핑하는 함수이다. 해시 함수에 의해 얻어지는 값은 해시 값, 해시 코드, 해시 체크섬 또는 간단하게 해시라고 한다. 그 용도 중 하나는 해시 테이블이라는 자료구조에 사용되며, 매우 빠른 데이터 검색을 위한 컴퓨터 소프트웨어에 널리 사용된다.

자바의 해시 관련 클래스

- **Collection**
  - HashSet : 중복해서 저장하지 않는 집합으로 사용할 수 있는 클래스
  - LinkedHashSet : 집합의 특징은 중복을 허용하지 않고, 순서를 가지지 않는다는 특징이 있지만, LinkedHashSet은 중복은 허용하지 않지만, 순서를 가진다는 특징이 있다.
- **Map**
  - HashMap : Key-Value를 사용하여 Hash Table의 값을 찾는다.
  - HashTable : HashMap과 동일한 특징을 가지지만, 차이점이라고 하면 Thread-Safe하여 동기화를 지원한다.
  - LinkedHashMap : 일반적으로 Map 자료구조는 순서를 가지지 않지만, LinkedHashMap은 들어온 순서대로 순서를 가진다.

자바의 컬렉션 프레임워크 계층



05 해시

1. 두 개 뽑아서 더하기

<문제 설명>

정수 배열 numbers가 주어집니다. numbers에서 서로 다른 인덱스에 있는 두 개의 수를 뽑아 더해서 만들 수 있는 모든 수를 배열에 오름차순으로 담아 return 하도록 solution 함수를 완성해주세요.

<입출력 예>

numbers	result
[2,1,3,4,1]	[2,3,4,5,6,7]
[5,0,2,7]	[2,5,7,9,12]

<제한 조건>

- numbers의 길이는 2 이상 100 이하입니다.
  - numbers의 모든 수는 0 이상 100 이하입니다.

입출력 예#1

2 = 1 + 1 입니다.
(1이 numbers에 두 개)
3 = 2 + 1 입니다.
4 = 1 + 3 입니다.
5 = 1 + 4 = 2 + 3 입니다.
6 = 2 + 4 입니다.
7 = 3 + 4 입니다.
따라서 [2,3,4,5,6,7] 을 return

입출력 예#2

2 = 0 + 2 입니다.
5 = 5 + 0 입니다.
7 = 0 + 7 = 5 + 2 입니다.
9 = 2 + 7 입니다.
12 = 5 + 7 입니다.
따라서 [2,5,7,9,12] 를 return

05 해시

2. 완주하지 못한 선수

<문제 설명>

수많은 마라톤 선수들이 마라톤에 참여하였습니다. 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주하였습니다.
마라톤에 참여한 선수들의 이름이 담긴 배열 participant와 완주한 선수들의 이름이 담긴 배열 completion이 주어질 때, 완주하지 못한 선수의 이름을 return 하도록 solution 함수를 작성해주세요.

<제한 조건>

- 마라톤 경기에 참여한 선수의 수는 1명 이상 100,000명 이하입니다.
- completion의 길이는 participant의 길이보다 1 작습니다.
- 참가자의 이름은 1개 이상 20개 이하의 알파벳 소문자로 이루어져 있습니다.
- 참가자 중에는 동명이인이 있을 수 있습니다.

<입출력 예>

participant	completion	return
["leo", "kiki", "eden"]	["eden", "kiki"]	"leo"
["marina", "josipa", "nikola", "vinko", "filipa"]	["josipa", "filipa", "marina", "nikola"]	"vinko"
["mislav", "stanko", "mislav", "ana"]	["stanko", "ana", "mislav"]	"mislav"

예제 #1
"leo"는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #2
"vinko"는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #3
"mislav"는 참여자 명단에는 두 명이 있지만, 완주자 명단에는 한 명밖에 없기 때문에 한명은 완주하지 못했습니다.

05 해시

3. 포켓몬

<문제 설명>

폰켓몬 총 N 마리의 폰켓몬 중에서 N/2마리를 가져가도 좋다고 했습니다.
폰켓몬은 종류에 따라 번호를 붙여 구분합니다. 따라서 같은 종류의 폰켓몬은 같은 번호를 가지고 있습니다.
예를 들어 연구실에 총 4마리의 폰켓몬이 있고, 각 폰켓몬의 종류 번호가 [3번, 1번, 2번, 3번]이라면 이는 3번 폰켓몬 두 마리, 1번 폰켓몬 한 마리, 2번 폰켓몬 한 마리가 있음을 나타냅니다. 이때, 4마리의 폰켓몬 중 2마리를 고르는 방법은 다음과 같이 6가지가 있습니다.

- 첫 번째(3번), 두 번째(1번) 폰켓몬을 선택
- 첫 번째(3번), 세 번째(2번) 폰켓몬을 선택
- 첫 번째(3번), 네 번째(3번) 폰켓몬을 선택
- 두 번째(1번), 세 번째(2번) 폰켓몬을 선택
- 두 번째(1번), 네 번째(3번) 폰켓몬을 선택
- 세 번째(2번), 네 번째(3번) 폰켓몬을 선택

이때, 첫 번째(3번) 폰켓몬과 네 번째(3번) 폰켓몬을 선택하는 방법은 한 종류(3번 폰켓몬 두 마리)의 폰켓몬만 가질 수 있지만, 다른 방법들은 모두 두 종류의 폰켓몬을 가질 수 있습니다.
따라서 위 예시에서 가질 수 있는 폰켓몬 종류 수의 최댓값은 2가 됩니다.

당신은 최대한 다양한 종류의 폰켓몬을 가지길 원하기 때문에, 최대한 많은 종류의 폰켓몬을 포함해서 N/2마리를 선택하려 합니다. N마리 폰켓몬의 종류 번호가 담긴 배열 nums가 매개변수로 주어질 때, N/2마리의 폰켓몬을 선택하는 방법 중, 가장 많은 종류의 폰켓몬을 선택하는 방법을 찾아, 그때의 폰켓몬 종류 번호의 개수를 return 하도록 solution 함수를 완성해주세요.

05 해시

3. 포켓몬

<제한 조건>

- nums는 폰켓몬의 종류 번호가 담긴 1차원 배열입니다.
- nums의 길이(N)는 1 이상 10,000 이하의 자연수이며, 항상 짝수로 주어집니다.
- 폰켓몬의 종류 번호는 1 이상 200,000 이하의 자연수로 나타냅니다.
- 가장 많은 종류의 폰켓몬을 선택하는 방법이 여러 가지인 경우에도, 선택할 수 있는 폰켓몬 종류 개수의 최댓값 하나만 return 하면 됩니다.

<입출력 예>

nums	result
[3,1,2,3]	2
[3,3,3,2,2,4]	3
[3,3,3,2,2,2]	2

입출력 예 #2

6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.
가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리, 2번 폰켓몬 한 마리, 4번 폰켓몬 한 마리를 고르면 되며, 따라서 3을 return 합니다.

입출력 예 #3

6마리의 폰켓몬이 있으므로, 3마리의 폰켓몬을 골라야 합니다.
가장 많은 종류의 폰켓몬을 고르기 위해서는 3번 폰켓몬 한 마리와 2번 폰켓몬 두 마리를 고르거나, 혹은 3번 폰켓몬 두 마리와 2번 폰켓몬 한 마리를 고르면 됩니다. 따라서 최대 고를 수 있는 폰켓몬 종류의 수는 2입니다.

06

탐색 알고리즘

06-1 탐색 알고리즘 개요
06-2 완전 탐색
06-3 이분 탐색

06 탐색 알고리즘

탐색 알고리즘 개요

탐색 개념

수많은 데이터 속에서 원하는 데이터를 찾는 것이다.

데이터가 정렬된 상태인지, 정렬되지 않은 상태인지에 따라 탐색 알고리즘이 달라진다

탐색 종류

- 완전 탐색
- 이분 탐색
- 문자열 탐색
- 해시 탐색
- 이진 탐색 트리

⋮

06 탐색 알고리즘

완전 탐색

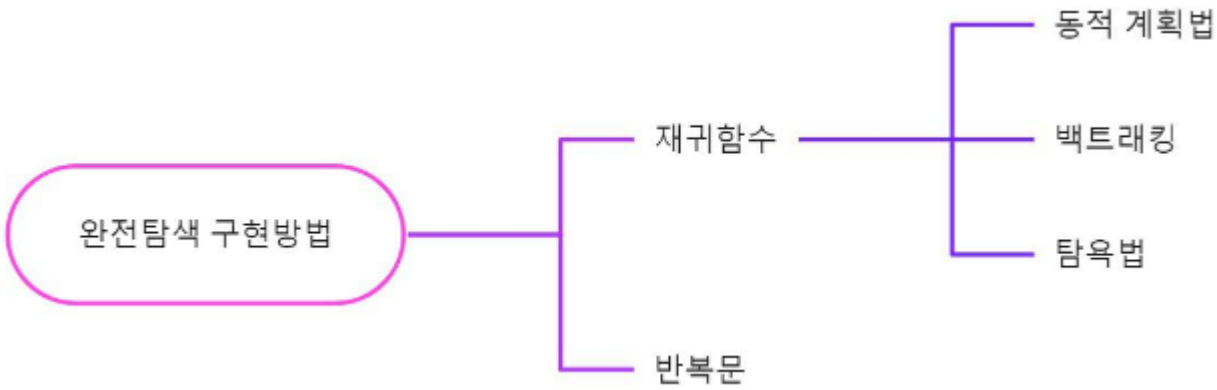
완전 탐색 개념

컴퓨터의 빠른 계산 능력을 이용하여 가능한 경우의 수를 일일이 나열하면서 답을 찾는 방법을 의미한다. '무식하게 푼다'라는 의미인 Brute-Force (브루트 포스)라고도 부른다.

답을 찾기 위한 경우의 수가 많은 경우에는 이용하기가 어려우므로 완전 탐색을 이용할 수 있는 경우인지 잘 파악하는 게 중요하다. 대신에 이 방식은 절대 답을 못 구하는 경우는 없으므로 나름 강력한 방법이다.

완전 탐색 기법

- 단순 Brute-Force
- 백트래킹 (Backtracking)
- 순열 (Permutation)
- 비트마스크(Bitmask)
- 재귀 함수
- DFS(깊이 우선 탐색) / BFS(너비 우선 탐색)



06 탐색 알고리즘

그래프 탐색

하나의 정점으로부터 시작하여 차례대로 모든 정점들을 한 번씩 방문하는 것

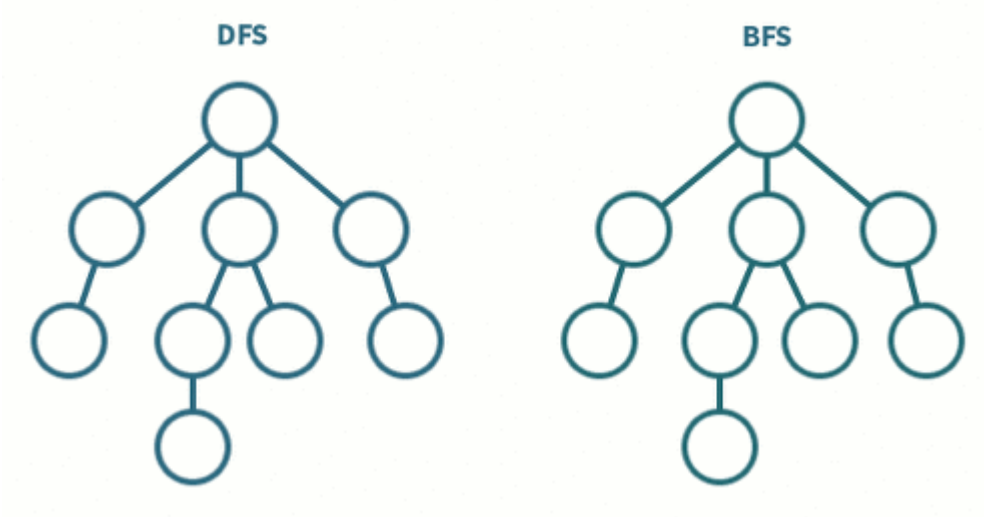
Ex) 특정 도시에서 다른 도시로 갈 수 있는지 없는지, 전자 회로에서 특정 단자와 단자가 서로 연결되어 있는지

깊이 우선 탐색 (DFS, Depth-First Search)

- 루트 노드(혹은 다른 임의의 노드)에서 시작해서 다음 분기(branch)로 넘어가기 전에 해당 분기를 완벽하게 탐색하는 방법
- 자기 자신을 호출하는 재귀적 순환 알고리즘 형태이며, 스택(Stack)를 사용하여 후입선출(LIFO) 원칙으로 탐색
- BFS보다 간단하지만, BFS에 비해 속도는 느림

너비 우선 탐색 (BFS, Breadth-First Search)

- 루트 노드(혹은 다른 임의의 노드)에서 시작해서 인접한 노드를 먼저 탐색하는 방법
- 재귀적으로 동작하지 않으며, 큐(Queue)를 사용하여 선입선출(FIFO) 원칙으로 탐색



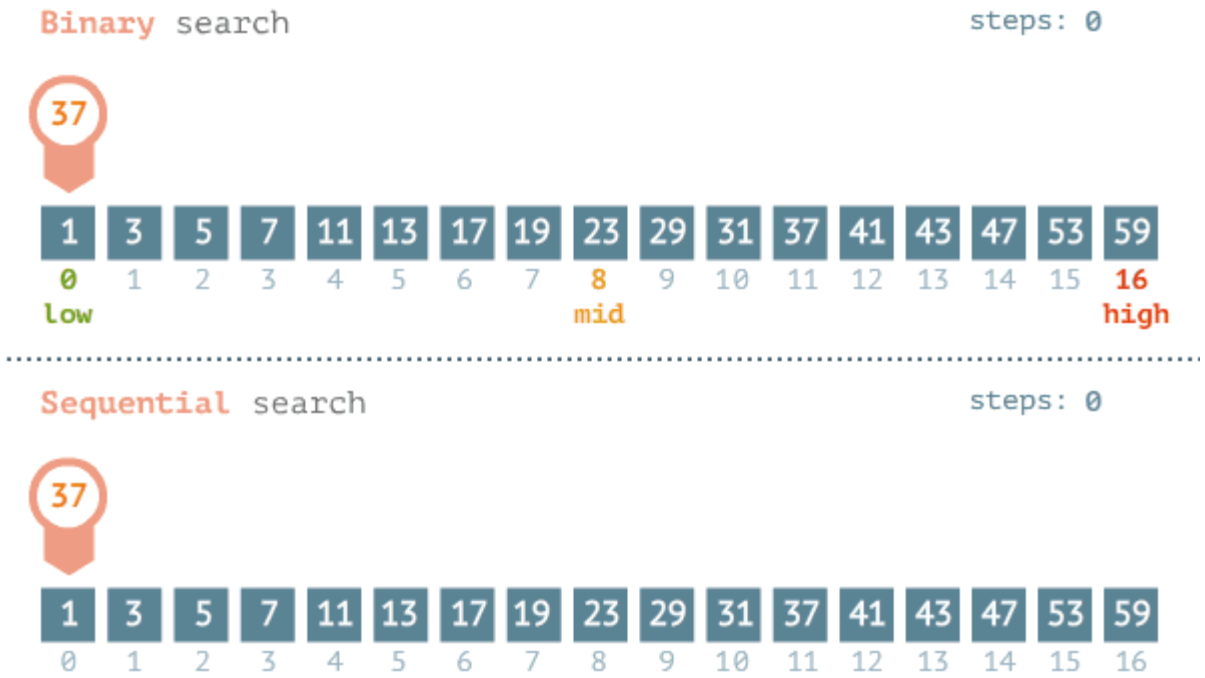
06 탐색 알고리즘

이분 탐색

이분 탐색 개념

정렬되어 있는 리스트에서 탐색 범위를 절반씩 좁혀가며 데이터를 탐색하는 방법이다.

이분 탐색은 배열 내부의 데이터가 **정렬되어 있어야만** 사용할 수 있는 알고리즘이다.



06 탐색 알고리즘

1. 모의고사

<문제 설명>

수포자는 수학을 포기한 사람의 준말입니다. 수포자 삼인방은 모의고사에 수학 문제를 전부 찍으려 합니다. 수포자는 1번 문제부터 마지막 문제까지 다음과 같이 찍습니다.

- 1번 수포자가 찍는 방식: 1, 2, 3, 4, 5, 1, 2, 3, 4, 5, ...
- 2번 수포자가 찍는 방식: 2, 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 3, 2, 4, 2, 5, ...
- 3번 수포자가 찍는 방식: 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, 3, 3, 1, 1, 2, 2, 4, 4, 5, 5, ...

1번 문제부터 마지막 문제까지의 정답이 순서대로 들은 배열 answers가 주어졌을 때, 가장 많은 문제를 맞힌 사람이 누구인지 배열에 담아 return 하도록 solution 함수를 작성해주세요.

<제한 조건>

- 시험은 최대 10,000 문제로 구성되어있습니다.
- 문제의 정답은 1, 2, 3, 4, 5중 하나입니다.
- 가장 높은 점수를 받은 사람이 여럿일 경우, return하는 값을 오름차순 정렬해주세요.

<입출력 예>

answers	return
[1,2,3,4,5]	[1]
[1,3,2,4,2]	[1,2,3]

입출력 예#1

수포자 1은 모든 문제를 맞혔습니다.
수포자 2는 모든 문제를 틀렸습니다.
수포자 3은 모든 문제를 틀렸습니다.
따라서 가장 문제를 많이 맞힌 사람은 수포자 1입니다.

입출력 예 #2

모든 사람이 2문제씩을 맞췄습니다.

06 탐색 알고리즘

4. 이분 탐색 예제

<문제 설명>

배열 arr과 정수 n이 주어졌을 때, 배열에서 n의 값을 이분 탐색 알고리즘으로 탐색하여 몇 번째 위치에서 찾았는지 확인하는 함수, solution을 완성해 보세요. 단, 찾는 값이 없을 때는 -1을 반환한다.

<입출력 예>

arr	n	return
[1 ,2, 3, 4, 5, 6]	4	3

<제한 조건>

- arr 은 길이 1 이상 50이하인 배열입니다.

06 탐색 알고리즘

5. 타겟 넘버

<문제 설명>

n개의 음이 아닌 정수들이 있습니다. 이 정수들을 순서를 바꾸지 않고 적절히 더하거나 빼서 타겟 넘버를 만들려고 합니다. 예를 들어 [1, 1, 1, 1, 1]로 숫자 3을 만들려면 다음 다섯 방법을 쓸 수 있습니다.
사용할 수 있는 숫자가 담긴 배열 numbers, 타겟 넘버 target이 매개변수로 주어질 때 숫자를 적절히 더하고 빼서 타겟 넘버를 만드는 방법의 수를 return 하도록 solution 함수를 작성해주세요.

```
-1+1+1+1+1 = 3
+1-1+1+1+1 = 3
+1+1-1+1+1 = 3
+1+1+1-1+1 = 3
+1+1+1+1-1 = 3
```

<제한 조건>

- 주어지는 숫자의 개수는 2개 이상 20개 이하
- 각 숫자는 1 이상 50 이하인 자연수입니다.
- 타겟 넘버는 1 이상 1000 이하인 자연수입니다.

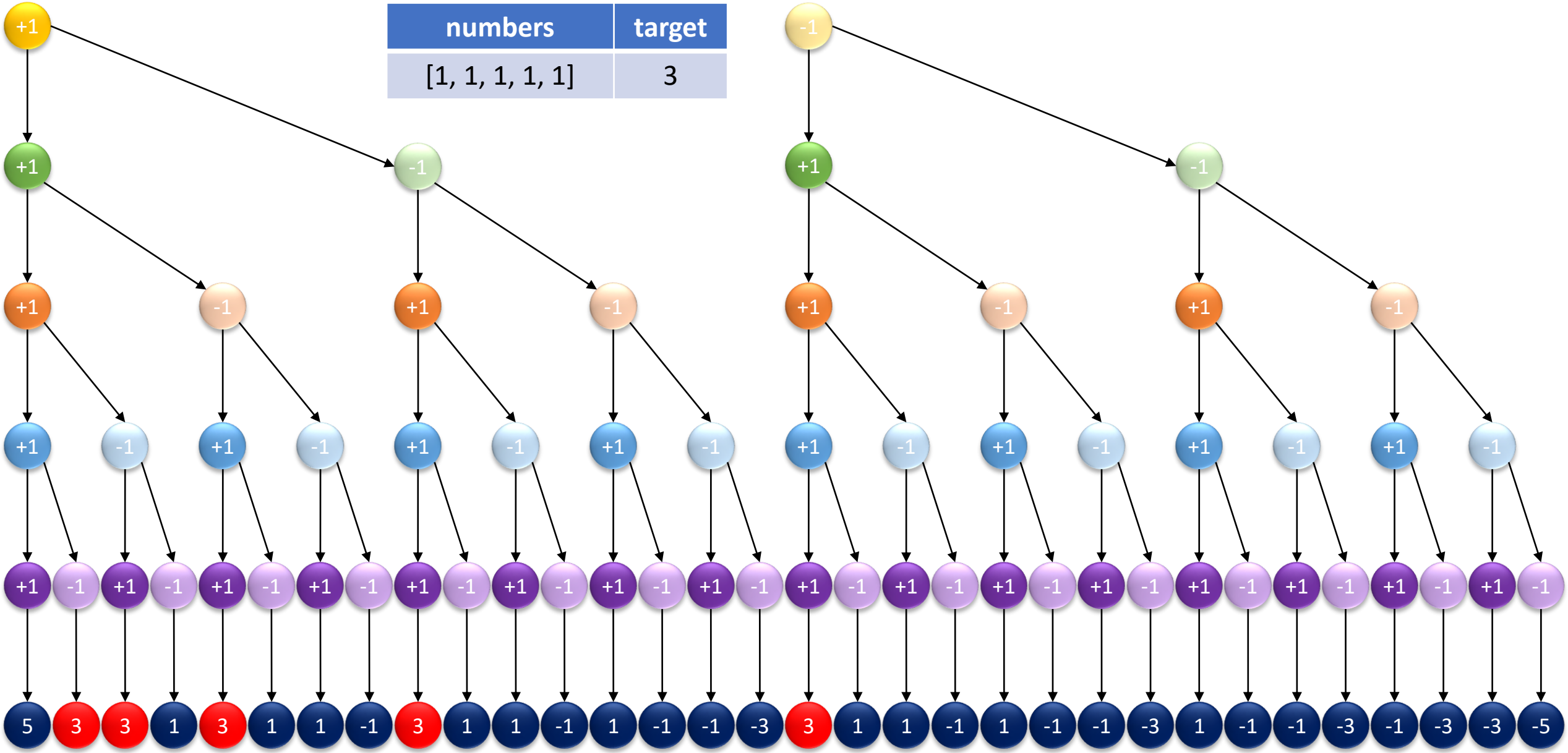
<입출력 예>

numbers	target	return
[1, 1, 1, 1, 1]	3	5
[4, 1, 2, 1]	4	2

입출력 예 #1
문제 예시와 같습니다.

입출력 예 #2
+4+1-2+1 = 4
+4-1+2-1 = 4
총 2가지 방법이 있으므로, 2를 return 합니다.

numbers	target
[1, 1, 1, 1, 1]	3



07

그리디 알고리즘

07-1 그리디 알고리즘 개요

07-2 동전 개수의 최소값

07-3 체육복

07 그리디 알고리즘

그리디 알고리즘 개요

그리디 알고리즘 개념

단어 뜻 그대로 '탐욕적' 문제해결 알고리즘으로 선택 단계마다 욕심을 부리며 가장 최고의 답을 선택하는 알고리즘이다.

즉, 현재 상태에서 보는 선택지 중 최선의 선택지가 전체 선택지 중 최선의 선택지라고 가정한다. 때문에 문제 전체를 보았을 때는 최적의 해결 방안이 아닐 수도 있다.

그리디 알고리즘 수행 과정

① 해 선택: 현재 상태에서 가장 최선이라고 생각되는 해를 선택한다.

② 적절성 검사: 현재 선택한 해가 전체 문제의 제약 조건에 벗어나지 않는지 검사한다.

③ 해 검사: 현재까지 선택한 해 집합이 전체 문제를 해결할 수 있는지 검사한다. 전체 문제를 해결하지 못한다면 ①로 돌아가 같은 과정을 반복한다.

07 그리디 알고리즘

1. 동전 개수의 최소값

<문제 설명>

동전은 총 6 종류이고, 동전을 적절히 사용하여 K 금액에 필요한 동전 개수의 최소값을 구하여 return 하는 solution 함수를 작성해주세요.

<제한 조건>

- 동전의 종류: 1, 5, 10, 50, 100, 500

<입출력 예>

K	return
100000	200
4200	10

입출력 예#1
500*200 이므로 200 return

입출력 예 #2
500*8 + 100*2 이므로 10 return

07 그리디 알고리즘

2. 체육복

<문제 설명>

점심시간에 도둑이 들어, 일부 학생이 체육복을 도난당했습니다. 다행히 여벌 체육복이 있는 학생이 이들에게 체육복을 빌려주려 합니다. 학생들의 번호는 체격 순으로 매겨져 있어, 바로 앞번호의 학생이나 바로 뒷번호의 학생에게만 체육복을 빌려줄 수 있습니다. 예를 들어, 4번 학생은 3번 학생이나 5번 학생에게만 체육복을 빌려줄 수 있습니다. 체육복이 없으면 수업을 들을 수 없기 때문에 체육복을 적절히 빌려 최대한 많은 학생이 체육수업을 들어야 합니다.

전체 학생의 수 n, 체육복을 도난당한 학생들의 번호가 담긴 배열 lost, 여벌의 체육복을 가져온 학생들의 번호가 담긴 배열 reserve가 매개변수로 주어질 때, 체육수업을 들을 수 있는 학생의 최댓값을 return 하도록 solution 함수를 작성해주세요.

<제한 조건>

- 전체 학생의 수는 2명 이상 30명 이하입니다.
- 체육복을 도난당한 학생의 수는 1명 이상 n명 이하이고 중복되는 번호는 없습니다.
- 여벌의 체육복을 가져온 학생의 수는 1명 이상 n명 이하이고 중복되는 번호는 없습니다.
- 여벌 체육복이 있는 학생만 다른 학생에게 체육복을 빌려줄 수 있습니다.
- 여벌 체육복을 가져온 학생이 체육복을 도난당했을 수 있습니다. 이때 이 학생은 체육복을 하나만 도난당했다고 가정하며, 남은 체육복이 하나이기에 다른 학생에게는 체육복을 빌려줄 수 없습니다.

07 그리디 알고리즘

2. 체육복

<입출력 예>

n	lost	reserve	return
5	[2, 4]	[1, 3, 5]	5
5	[2, 4]	[3]	4
3	[3]	[1]	2

입출력 예#1

1번 학생이 2번 학생에게 체육복을 빌려주고, 3번 학생이나 5번 학생이 4번 학생에게 체육복을 빌려주면 학생 5명이 체육수업을 들을 수 있습니다.

입출력 예 #2

3번 학생이 2번 학생이나 4번 학생에게 체육복을 빌려주면 학생 4명이 체육수업을 들을 수 있습니다.

08

구현

07-1 스택 활용

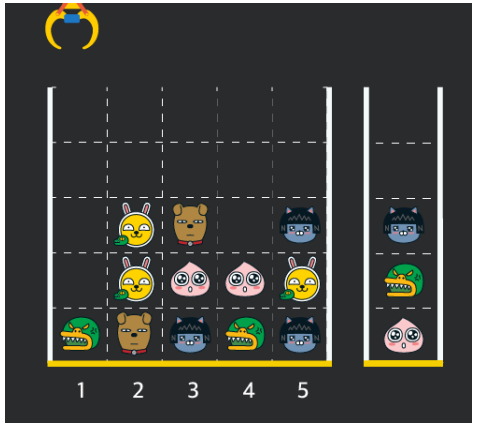
07-2 큐 활용

08 구현 – 스택 활용

1. 크레인 인형뽑기 게임

<문제 설명>

크레인 인형뽑기 기계를 모바일 게임으로 만들려고 합니다. 화면 구성과 규칙을 다음과 같다.
게임 화면은 "1 x 1" 크기의 칸들로 이루어진 "N x N" 크기의 정사각 격자이며 위쪽에는 크레인이 있고 오른쪽에는 바구니가 있습니다. (위 그림은 "5 x 5" 크기의 예시입니다). 각 격자 칸에는 다양한 인형이 들어 있으며 인형이 없는 칸은 빈칸입니다. 모든 인형은 "1 x 1" 크기의 격자 한 칸을 차지하며 격자의 가장 아래 칸부터 차곡차곡 쌓여 있습니다. 게임 사용자는 크레인을 좌우로 움직여서 멈춘 위치에서 가장 위에 있는 인형을 집어 올릴 수 있습니다. 집어 올린 인형은 바구니에 쌓이게 되는 데, 이때 바구니의 가장 아래 칸부터 인형이 순서대로 쌓이게 됩니다. 다음 그림은 [1번, 5번, 3번] 위치에서 순서대로 인형을 집어 올려 바구니에 담은 모습입니다.



만약 같은 모양의 인형 두 개가 바구니에 연속해서 쌓이게 되면 두 인형은 터뜨려지면서 바구니에서 사라지게 됩니다. 위 상태에서 이어서 [5번] 위치에서 인형을 집어 바구니에 쌓으면 같은 모양 인형 두 개가 없어집니다.

08 구현 – 스택 활용

1. 크레인 인형뽑기 게임

크레인 작동 시 인형이 집어지지 않는 경우는 없으나 만약 인형이 없는 곳에서 크레인을 작동시키는 경우에는 아무런 일도 일어나지 않습니다. 또한 바구니는 모든 인형이 들어갈 수 있을 만큼 충분히 크다고 가정합니다. (그림에서는 화면표시 제약으로 5칸만으로 표현하였음)

게임 화면의 격자의 상태가 담긴 2차원 배열 board와 인형을 집기 위해 크레인을 작동시킨 위치가 담긴 배열 moves가 매개변수로 주어질 때, 크레인을 모두 작동시킨 후 터트려져 사라진 인형의 개수를 return 하도록 solution 함수를 완성해주세요.

<제한 조건>

- board 배열은 2차원 배열로 크기는 "5 x 5" 이상 "30 x 30" 이하입니다.
- board의 각 칸에는 0 이상 100 이하인 정수가 담겨있습니다.
  - 0은 빈 칸을 나타냅니다.
  - 1 ~ 100의 각 숫자는 각기 다른 인형의 모양을 의미하며 같은 숫자는 같은 모양의 인형을 나타냅니다.
- moves 배열의 크기는 1 이상 1,000 이하입니다.
- moves 배열 각 원소들의 값은 1 이상이며 board 배열의 가로 크기 이하인 자연수입니다.

08 구현 – 스택 활용

1. 크레인 인형뽑기 게임

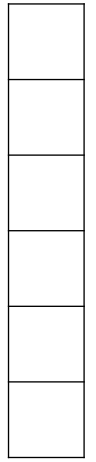
<입출력 예>

board	moves	result
[[0,0,0,0,0],[0,0,1,0,3], [0,2,5,0,1],[4,2,4,4,2],[3,5,1,3,1]]	[1,5,3,5,1,2,1,4]	4

입출력 예#1
인형의 처음 상태는 문제에 주어진 예시와 같습니다.
크레인이 [1, 5, 3, 5, 1, 2, 1, 4] 번 위치에서 차례대로 인형을 집어서 바구니에 옮겨 담은 후, 상태는 아래 그림과 같으며
바구니에 담는 과정에서 터트려져 사라진 인형은 4개 입니다.

[1, 5, 3, 5, 1, 2, 1, 4]

0	0	0	0	0
0	0	1	0	3
0	2	5	0	1
4	2	4	4	2
3	5	1	3	1
1	2	3	4	5



스택

08 구현 – 큐 활용

2. 프린터

<문제 설명>

일반적인 프린터는 인쇄 요청이 들어온 순서대로 인쇄합니다. 그렇기 때문에 중요한 문서가 나중에 인쇄될 수 있습니다. 이런 문제를 보완하기 위해 중요도가 높은 문서를 먼저 인쇄하는 프린터를 개발했습니다. 이 새롭게 개발한 프린터는 아래와 같은 방식으로 인쇄 작업을 수행합니다.

- 인쇄 대기목록의 가장 앞에 있는 문서(J)를 대기목록에서 꺼냅니다.
- 나머지 인쇄 대기목록에서 J보다 중요도가 높은 문서가 한 개라도 존재하면 J를 대기목록의 가장 마지막에 넣습니다.
- 그렇지 않으면 J를 인쇄합니다.

예를 들어, 4개의 문서(A, B, C, D)가 순서대로 인쇄 대기목록에 있고 중요도가 2 1 3 2 라면 C D A B 순으로 인쇄하게 됩니다.

내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 알고 싶습니다. 위의 예에서 C는 1번째로, A는 3번째로 인쇄됩니다.

현재 대기목록에 있는 문서의 중요도가 순서대로 담긴 배열 priorities와 내가 인쇄를 요청한 문서가 현재 대기목록의 어떤 위치에 있는지를 알려주는 location이 매개변수로 주어질 때, 내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 return 하도록 solution 함수를 작성해주세요.

08 구현 – 큐 활용

2. 프린터

<제한 조건>

현재 대기목록에는 1개 이상 100개 이하의 문서가 있습니다.

인쇄 작업의 중요도는 1~9로 표현하며 숫자가 클수록 중요하다는 뜻입니다.

location은 0 이상 (현재 대기목록에 있는 작업 수 - 1) 이하의 값을 가지며 대기목록의 가장 앞에 있으면 0, 두 번째에 있으면 1로 표현합니다.

<입출력 예>

priorities	location	return
[2, 1, 3, 2]	2	1
[1, 1, 9, 1, 1, 1]	0	5

입출력 예#1

문제에 나온 예와 같습니다.

입출력 예 #2

6개의 문서(A, B, C, D, E, F)가 인쇄 대기목록에 있고 중요도가 1 1 9 1 1 1 이므로 C D E F A B 순으로 인쇄합니다.

잠시 후에 만나요.