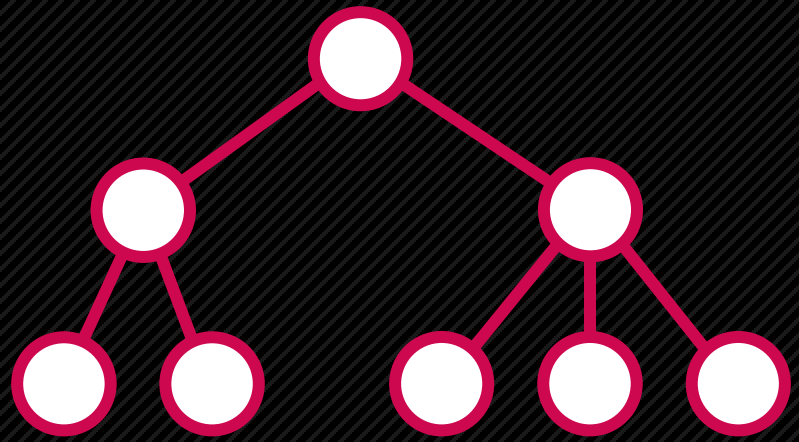
알고리즘 특강
힙/해시

데이터를 빠르게 삽입/삭제/탐색 할 수 있는 자료구조입니다.
각각의 특성을 잘 생각해서 다양한 문제에 적용해 봅시다.

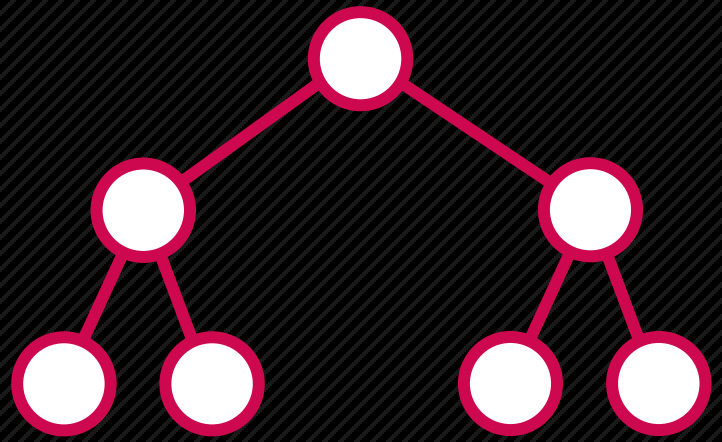
이진 트리



Tree

- 루트 노드와 자식 노드의 **연결관계가 반복적으로 구성된 자료구조.**
- 루트 노드의 **부모는 0개**이며, 자식 노드의 **부모는 1개**이다.
- 루트 노드는 **0개 이상의 자식 노드**를 갖고 있으며, 자식 노드 또한 **0개 이상의 자식 노드**를 가지고 있다.
- 사이클을 갖고 있지 않다.

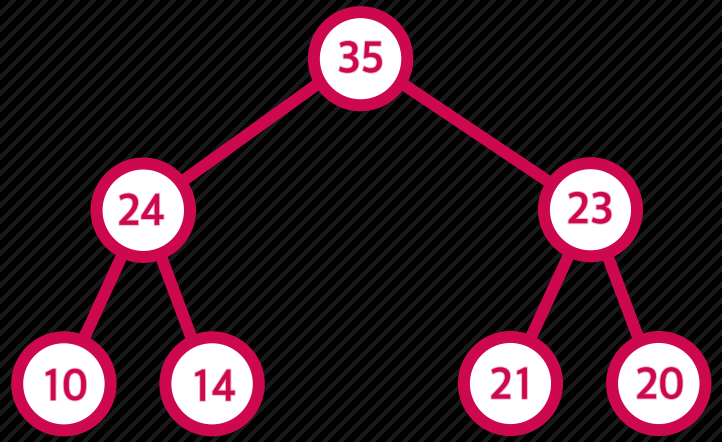
이진 트리



Binary Tree

- 트리에서, 자식 노드의 개수가 0개 이상 2개 이하로 제한된 트리.

힙



Heap

- 완전 이진 트리의 일종으로, 부모의 값이 항상 자식보다 크거나 작아야 함.
- 즉, 루트는 최댓값이거나, 최솟값임이 보장됨.
- 최댓값/최솟값을 O(1)만에 찾을 수 있는 자료구조.

Heap in Python

```python
import heapq

_list = [32, 16, 54, 94, 81, 31]
heapq.heapify(_list)

heapq.heappush(_list, 7)
print(heapq.heappop(_list))

print(heapq.heappushpop(_list, 100))

small_elements = heapq.nsmallest(4, _list)
print(small_elements)

large_elements = heapq.nlargest(4, _list)
print(large_elements)
```

- 사용 전 반드시 import!
- heapq는 리스트 기반 자료구조이며, 기존에 존재하던 list에 heapify를 사용해 배치
- heappush: 값을 heap에 넣음
- heappop: heap에 있는 값 중 최솟값을 뺌
- heappushpop: push하고, pop함.
- nsmallest: heap의 원소 중 최솟값 n개를 리턴
- nlargest: heap의 원소 중 최댓값 n개를 리턴

직접 짜보자!

Silver 1 - 최소 힙 (#1972)

요약

- 최소 힙을 만들어 다음과 같은 일련의 연산을 수행하는 프로그램을 작성하시오.
- 연산은 자연수를 넣거나, 최댓값을 출력하고, 제거하는 연산이다.

제약조건

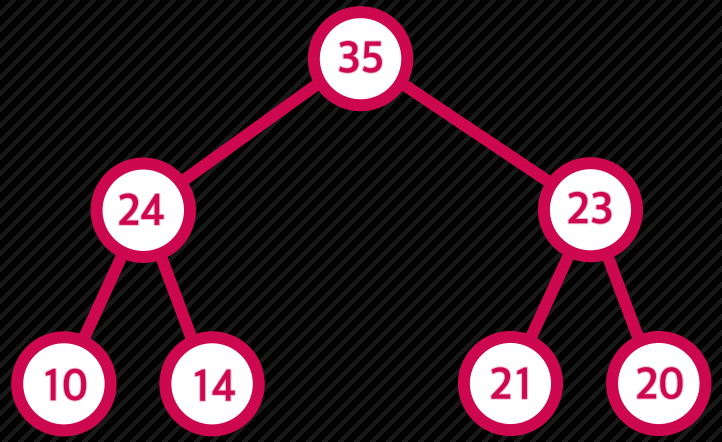
- 총 연산의 수 N은 $1 <= N <= 100,000$ 이다.
- 입력되는 자연수는 2^{31} 미만 이다.

그럼 최대 힙은?

```
_list = [32, 16, 54, 94, 81, 31]
_list = list(map(lambda x : x * -1, _list))
heapq.heapify(_list)
```

- -1을 곱하면 대소 관계가 뒤집힘! → 각 원소에 -1을 곱하자.

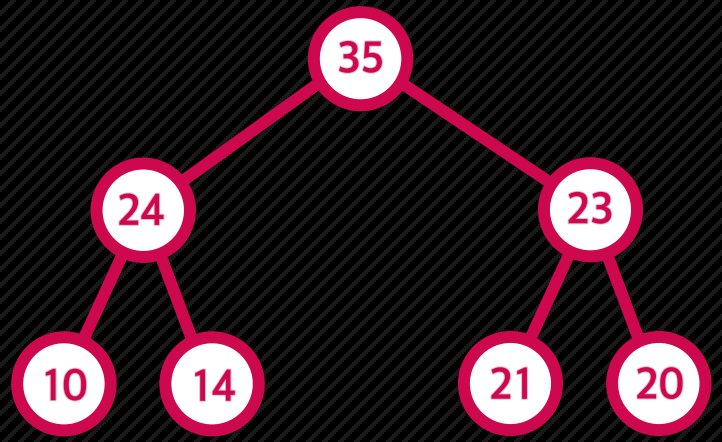
그래서 힙은?



Heap

- 완전 이진 트리의 일종으로, 부모의 값이 항상 자식보다 크거나 작아야 함.
- 즉, 루트는 최댓값이거나, 최솟값임이 보장됨.
- 최댓값/최솟값을 O(1)만에 찾을 수 있는 자료구조.

그래서 힙은?



Heap

- 최댓값/최솟값을 O(1)만에 찾을 수 있는 자료구조.

힙을 활용해보자!

Gold 4 - N번째 큰 수 (#2075)

요약

- N * N 개의 자연수가 있다.
- 자연수가 주어졌을 때, N번째 큰 수를 찾는 프로그램을 작성하시오.

제약조건

- N은 1 <= N <= 1,500 이다.
- 각각의 수의 범위는 $1 <= N_i <= 1,500$ 미만 이다.
- 사용되는 메모리는 12MB 이하여야 한다.

힙을 활용해보자!

Gold 4 - N번째 큰 수

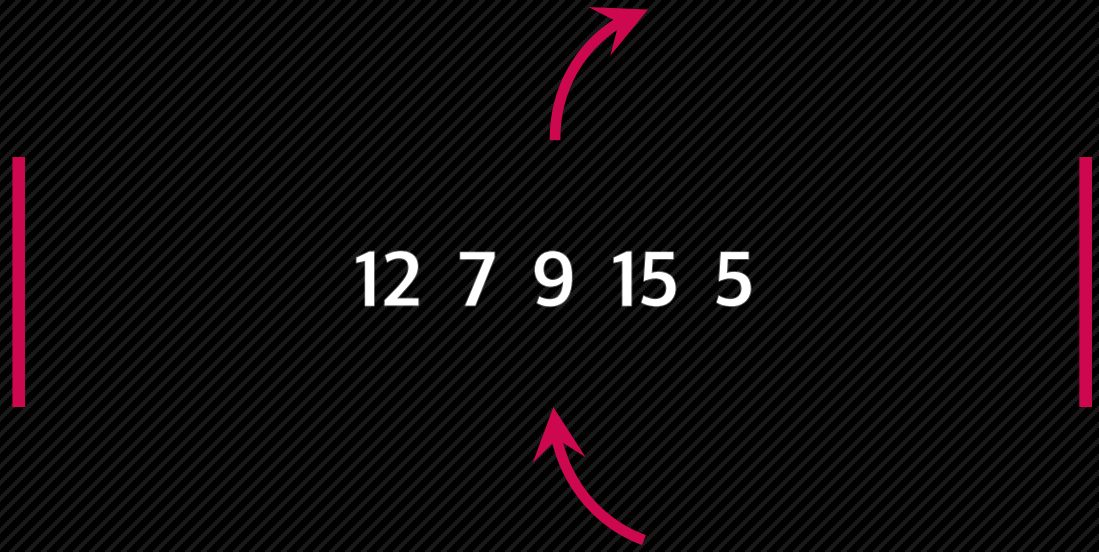
Heap

- 최댓값/최솟값을 O(1)만에 찾을 수 있는 자료구조.

힙을 활용해보자!

Gold 4 - N번째 큰 수

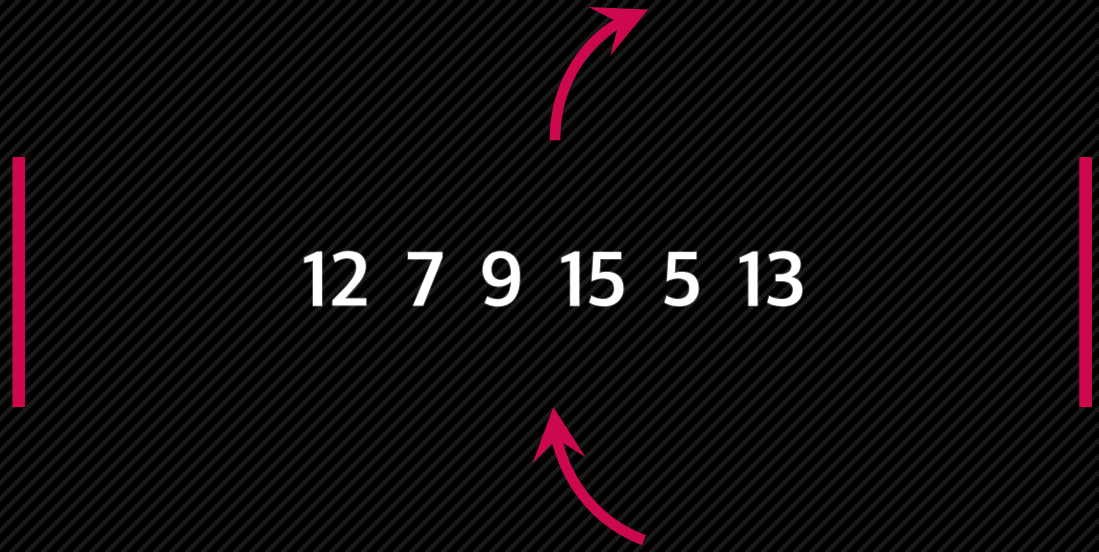
12 7 9 15 5
13 8 11 19 6
21 10 26 31 16
48 14 28 35 25
52 20 32 41 49



힙을 활용해보자!

Gold 4 - N번째 큰 수

12 7 9 15 5
13 8 11 19 6
21 10 26 31 16
48 14 28 35 25
52 20 32 41 49



힙을 활용해보자!

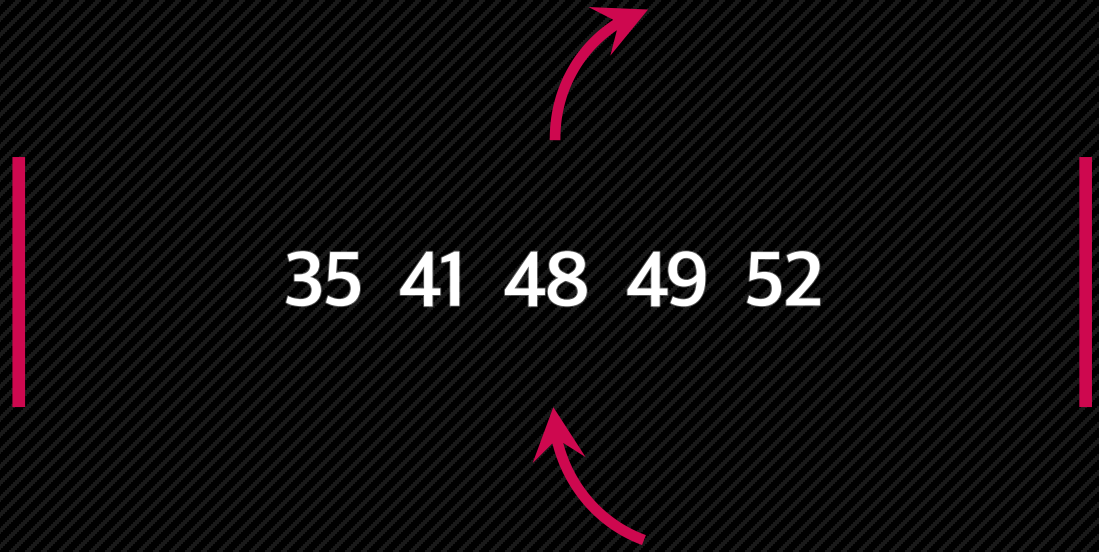
Gold 4 - N번째 큰 수

12 7 9 15 5
13 8 11 19 6
21 10 26 31 16
48 14 28 35 25
52 20 32 41 49

힙을 활용해보자!

Gold 4 - N번째 큰 수

12 7 9 15 5
13 8 11 19 6
21 10 26 31 16
48 14 28 35 25
52 20 32 41 49



최댓값과 최솟값을 활용하는 문제는 힙을 고려해보자!

Level 2 - 더 맵게

요약

- 모든 음식의 스코빌 지수를 K 이상으로 만드려고 한다.
- 스코빌 지수가 K 미만인 경우, 음식을 섞어 스코빌 지수를 올리려고 하는데, 이때 스코빌 지수는 (낮은 스코빌 지수) + (두번째로 낮은 스코빌 지수) * 2 가 된다.
- 모든 음식의 스코빌 지수가 K 이상이 되도록 하기 위해 섞어야 하는 최소 횟수를 구하는 프로그램을 작성하시오.

제약조건

- 음식의 수의 범위는 $2 <= N <= 1,000,000$ 이다.
- 음식의 스코빌 지수의 범위는 $0 <= N_i <= 1,000,000$ 이다.
- K의 범위는 $2 <= K <= 1,000,000,000$ 이다.

이런 건 어떨까?

Gold 3 - 중앙값 구하기 (#2696)

요약

- 홀수 번째 수를 읽을 때 마다, 지금까지 입력 받은 값의 중앙값을 출력하는 프로그램을 작성하시오.

제약조건

- 수열의 크기 M은 $1 <= M <= 9,999$ 이다.
- 각각의 수의 범위는 $1 <= M_i <= 2^{32}-1$ 미만 이다.

이런 건 어떨까?

Gold 3 - 중앙값 구하기

아니!!! 힙은 최솟값과 최댓값만 된다면서요!!!!!

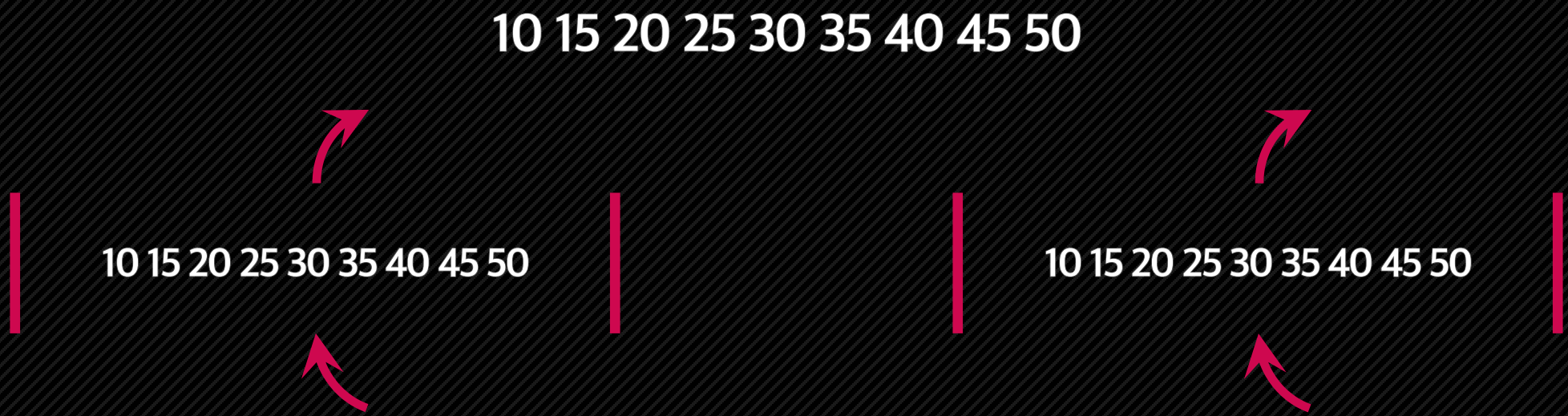
이런 건 어떨까?

Gold 3 - 중앙값 구하기

10 15 20 25 30 35 40 45 50

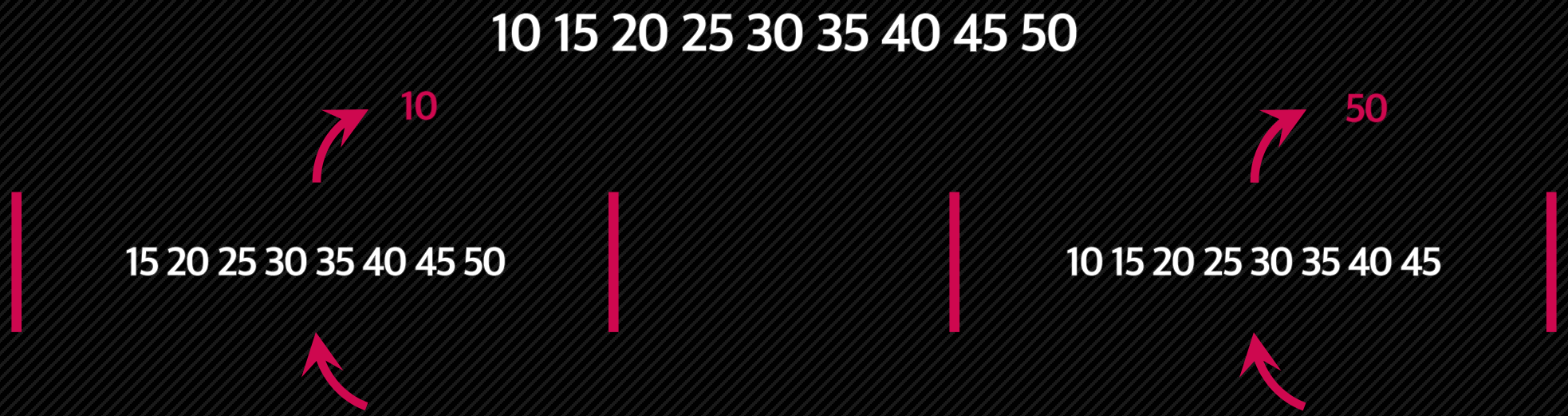
이런 건 어떨까?

Gold 3 - 중앙값 구하기



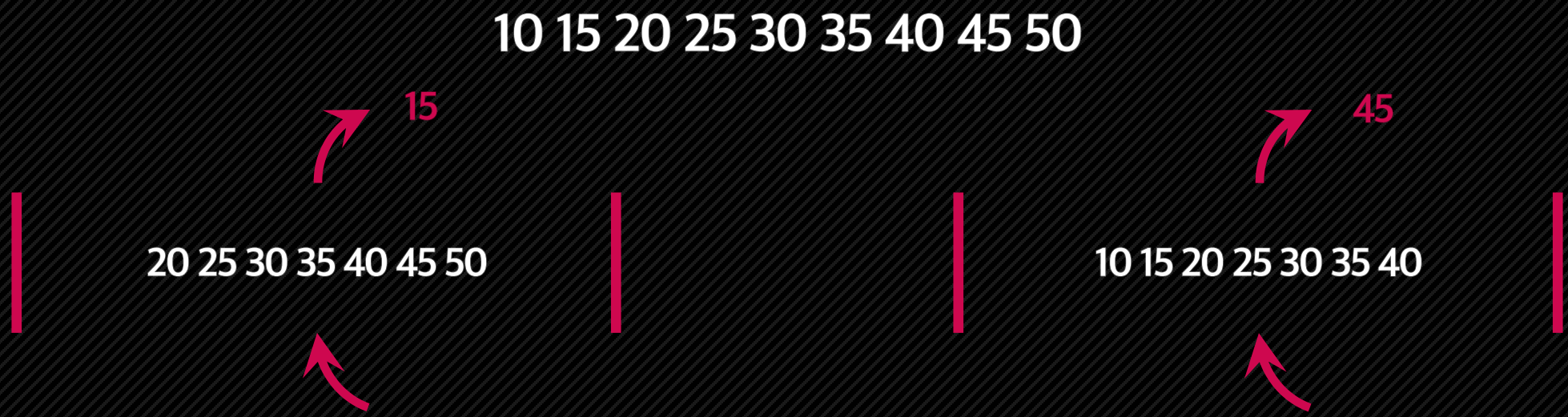
이런 건 어떨까?

Gold 3 - 중앙값 구하기



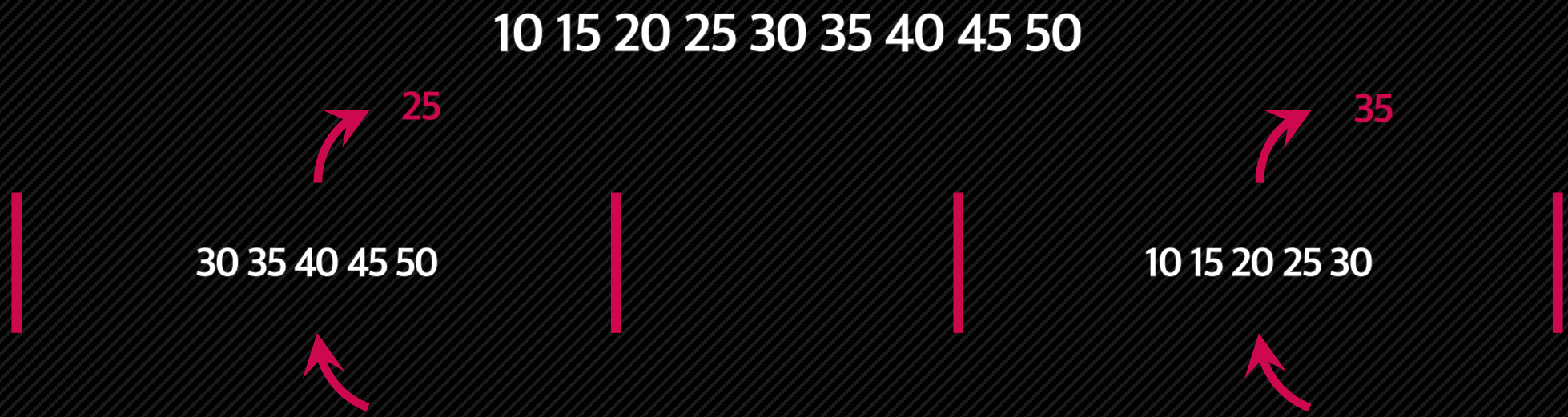
이런 건 어떨까?

Gold 3 - 중앙값 구하기



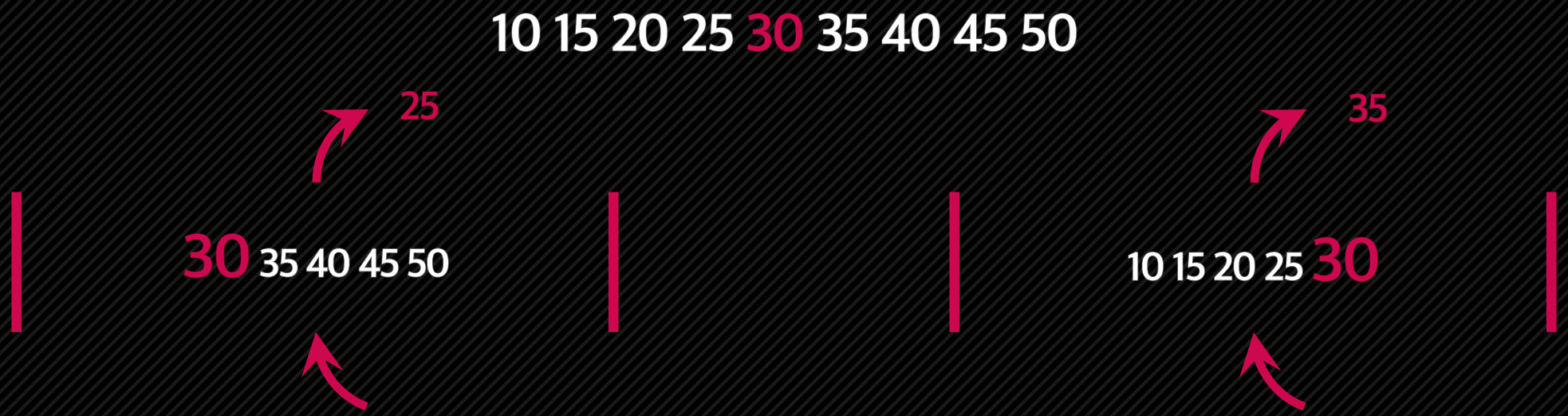
이런 건 어떨까?

Gold 3 - 중앙값 구하기

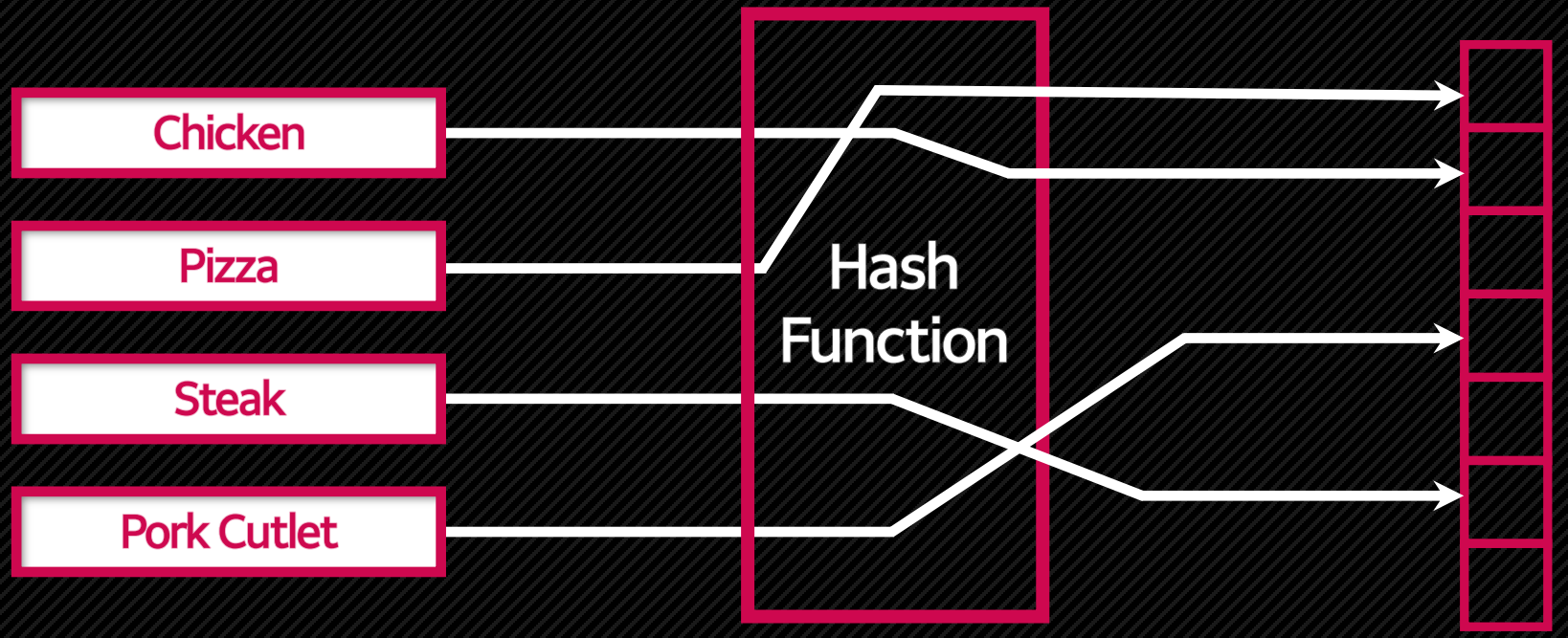


이런 건 어떨까?

Gold 3 - 중앙값 구하기



해시



Hash

- 임의의 데이터에 대해 고정된 길이의 데이터로 매핑하는 자료구조.
- 이론적으로 삭제 O(1), 삽입 O(1), 검색 O(1)의 시간복잡도를 갖고 있다.
- 그러나, 해시 테이블 내부의 값이 많아지면 해시충돌 현상이 발생해 기본 연산의 시간이 길어진다.

Dict

```
_dict = dict(one = 1, two = 2, three = 3)
_dict = {'one': 1, 'two': 2, 'three': 3}
```

일반적인 딕셔너리 만들기

Dict

```
_number = ['one', 'two', 'three', 'four']
_num = [1, 2, 3, 4]
_dict = dict(zip(_number, _num))

new_dict = {word : number for word, number in _dict.items() if number > 1}
```

이건 어때요?

Zip

2개 이상의 iterable한 데이터를 묶어주는 메소드

```
_number = ['one', 'two', 'three', 'four']
_num = [1, 2, 3, 4]
_zip = (zip(_number, _num))
for i in _zip:
    print(i)
```

```
('one', 1)
('two', 2)
('three', 3)
('four', 4)
```

Dict

딕셔너리 안에 키가 있으면 **값을 출력하고**, 없으면 **값을 넣자.**

```
if 5 in _dict:
    print(_dict[5])
else:
    _dict[5] = "Test"
    print("Test")
```

Dict

딕셔너리 안에 키가 있으면 **값을 출력하고**, 없으면 **값을 넣자.**

```
if 5 in _dict:
    print(_dict[5])
else:
    _dict[5] = "Test"
    print("Test")
```

```
print(_dict.setdefault(5, "Test"))
```

Sorted?

```
_number = ['one', 'two', 'three', 'four']
_num = [1, 2, 3, 4]
_dict = dict((zip(_number, _num)))

print(sorted(_dict))
```

출력 값은?

Sorted?

```
_number = ['one', 'two', 'three', 'four']
_num = [1, 2, 3, 4]
_dict = dict((zip(_number, _num)))

print(sorted(_dict))
```

출력 값은?

['four', 'one', 'three', 'two']

Sorted?

```
_number = ['one', 'two', 'three', 'four']
_num = [1, 2, 3, 4]
_dict = dict((zip(_number, _num)))

print(sorted(_dict.items()))
```

출력 값은?

[('four', 4), ('one', 1), ('three', 3), ('two', 2)]

Set

```
_set = set()
```

기본적인 set의 선언

```
_list = [1, 2, 3, 4, 5, 6, 1, 2, 3]
_set = set(_list)
```

당연히 list로도 선언이 가능하다.

```
print(_set & {1, 2, 3})
print(_set + {10, 11, 12})
```

set은 본래 집합을 위한 자료구조이기 때문에,
+ (합집합), - (차집합), & (교집합) 같은 연산을 지원

```
print(sorted(_set))
```

dict와 달리 값의 쌍이 없기 때문에 정렬해도 무방.

Set의 응용

중복된 데이터를 날려버리자!

```
_list = ["test", "chicken", "pizza", "chicken", "hamburger"]
print(list(set(_list)))

# 만약 대소문자를 신경 쓰지 않는다면?
_list = ["Test", "test", "tEst", "TEST"]
print(list(set(map(lambda x : x.lower(), _list))))
```

기억하세요?

Silver 4 - 수 찾기

요약

- N개의 정수 A[1], A[2], …, A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

제약조건

- N의 범위는 1 <= N <= 100,000 이다.
- 찾아야 하는 수의 개수 M의 범위는 1 <= M <= 100,000 이다.
- 모든 수의 범위는 $-2^{31} < N_i < 2^{31}$ 이다.

기억하세요?

Silver 4 - 수 찾기

```python
import sys
input = sys.stdin.readline

_ = input()
_set = set(map(int, input().split()))
q = input()
_list = list(map(int, input().split()))

print(*[1 if dt in _set else 0 for dt in _list], sep = '\n')
```

Silver 3 - 문자열 집합 (#14425)

요약

- 총 N개의 문자열로 이루어진 집합 S가 주어진다.
- 주어진 M개의 문자열 중 S에 포함된 문자열의 수를 출력하는 프로그램을 작성하시오.

제약조건

- N과 M은 1 <= N, M <= 10,000 이다.
- 문자의 길이는 최대 500이다.

중요한 구조니까 하나만 더!

Silver 4 - 나는야 포켓몬 마스터 이다솜 (#1620)

요약

- 포켓몬의 이름이 N개 주어지며, 이후 M개의 줄에 숫자 또는 포켓몬 이름이 주어진다.
- 숫자를 입력 받았을 시 포켓몬의 이름을, 포켓몬의 이름을 입력 받았을 시 번호를 출력하는 프로그램을 작성하시오.

제약조건

- N과 M은 1 <= N, M <= 100,000 이다.
- 포켓몬 이름의 길이는 최대 20이다.

여기까지 풀면 기능은 마스터!

Gold 4 - 생태학 (#4358)

요약

- 나무들이 중복을 포함해서 주어질 때, 각 종이 전체에서 몇 %를 차지하는지 출력하는 프로그램을 작성하시오.

제약조건

- 나무 종은 최대 10,000종 이다.
- 주어지는 나무의 수는 최대 1,000,000그루 이다.

Level 2 - 오픈채팅방 (2019 Kakao Blind Recruitment 1차 코딩테스트)

문제 설명이 기니 직접 읽어보고 풀어봅시다!

추가 추천 문제

Gold 4 - 카드 정렬하기 (#1715)

- 예제를 연구해보면 '이걸' 써야 합니다. (Hint: Greedy)

Gold 5 - 이중 우선순위 큐 (#7662)

- 가…운…데?

Level 2 - 위장

- 결국 변환만 제대로 하면 풀 수 있는 문제네요?

Level 3 - N으로 표현

- 동적계획법 문제 같지만, 배운 걸로도 충분히 풀 수 있는 문제입니다! 아이디어가 중요하니 잘 생각해보세요.