



# 1. 스택, 큐, 데크, 그래프

자료구조의 표현과 활용예시

# 기초 알고리즘 강의 구성 목록

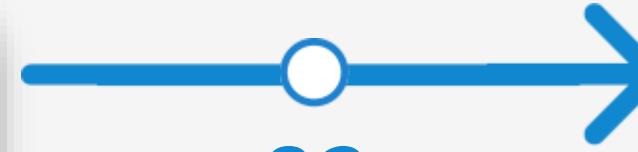


01

스택, 큐, 데크, 그래프

알고리즘 학습을 위한  
기초 도구인 자료구조를  
학습  
자료 구조의 동작 방식을  
중점적으로 이해하고,  
활용처를 알아본다.

05. 07

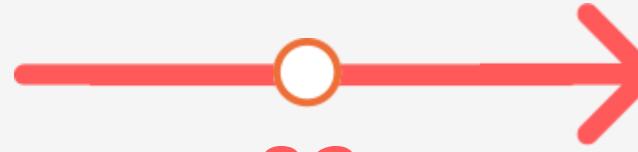


02

BFS/DFS

그래프 구조를 체계적으  
로 탐색하기 위한 기본 도  
구  
다양한 알고리즘(최단경  
로, 백트래킹, 네트워크  
등)의 핵심 기반을 배운  
다

05.14



03

DP와 그리디

동적 계획법과  
탐욕알고리즘의 정의와  
동작 원리를 면밀히  
살펴보고,  
활용하는 다양한 방법을  
배운다.

05.21

# 학습 목표

---

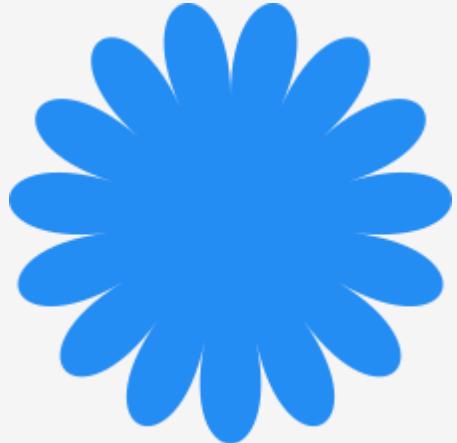
- 자료구조와 알고리즘이 무엇인지 알아본다.
- 스택, 큐, 덱을 소개하고, 동작 원리를 알아본다.
- 스택, 큐, 덱 기반 문제 해결 과정을 알아본다.
- 그래프와 트리가 무엇인지 학습한다.
- 그래프와 트리의 저장 방법을 학습한다.

기초가 되는 다양한 자료구조와 알고리즘의 동작 원리를 정확하게 학습하여, 다양한 사고방식을 습득하자!

실습과 구현은 배운 사고를 토대로 스스로 해보자!

# 목차

---



## 1. 자료구조와 알고리즘이란?

-소개

## 2. 스택

- 스택의 연산
  - 스택 그려보기
  - 스택구현체 사용하기
- 

## 3. 큐

- 큐의 연산
  - 큐 그려보기
  - 큐구현체 사용하기
- 

## 4. 그래프

- 그래프란
  - 그래프의 종류
  - 그래프를 저장하는 방법
- 

## 5. 트리

- 트리란
- 트리의 종류
- 트리를 저장하는 방법

## 6. 마무리 및 질문



# 자료구조와 알고리즘이란?



자료 구조는 데이터를 저장하는 구조입니다.  
데이터를 문제에 맞는 적절한 자료 구조를  
선택해 저장하면 문제를 효율적으로 해결할 수  
있습니다.

알고리즘은 문제를 정확히 정의하고,  
어떻게 풀어나갈지 진행사항을 정리하는  
것입니다.  
이때, 자료구조와 데이터를 사용합니다.

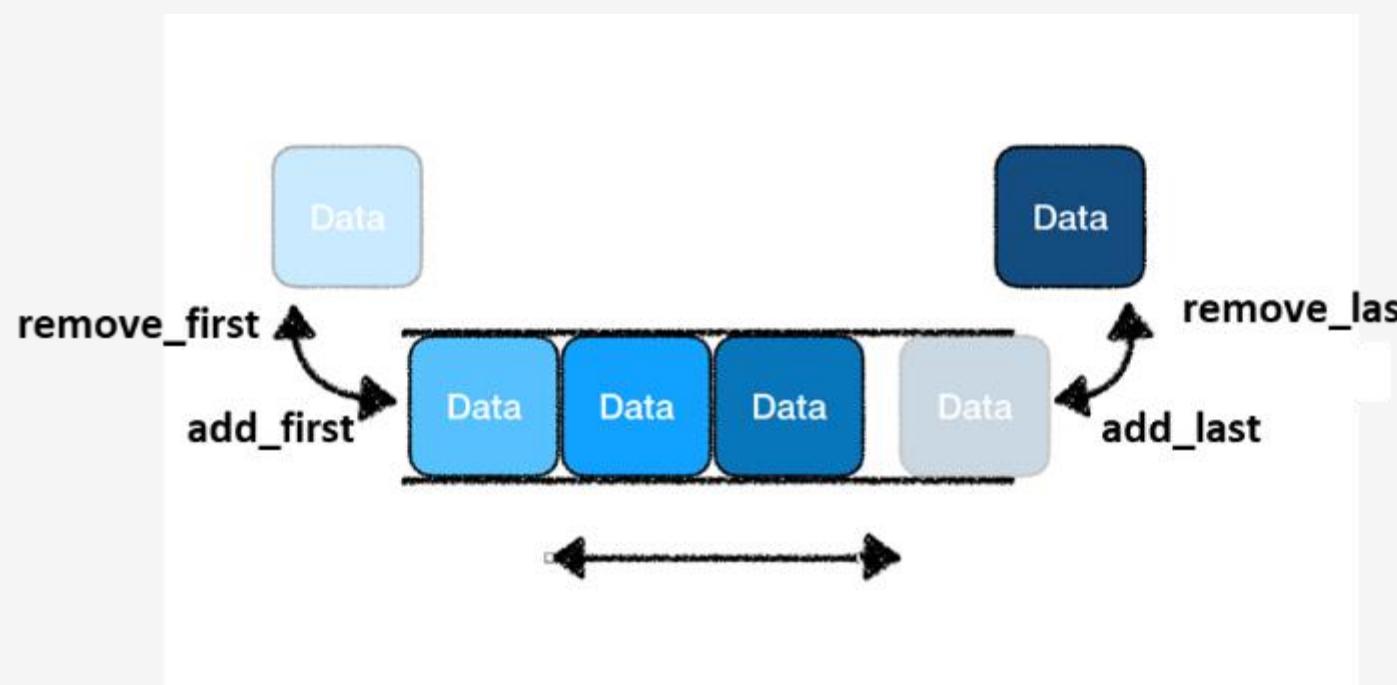
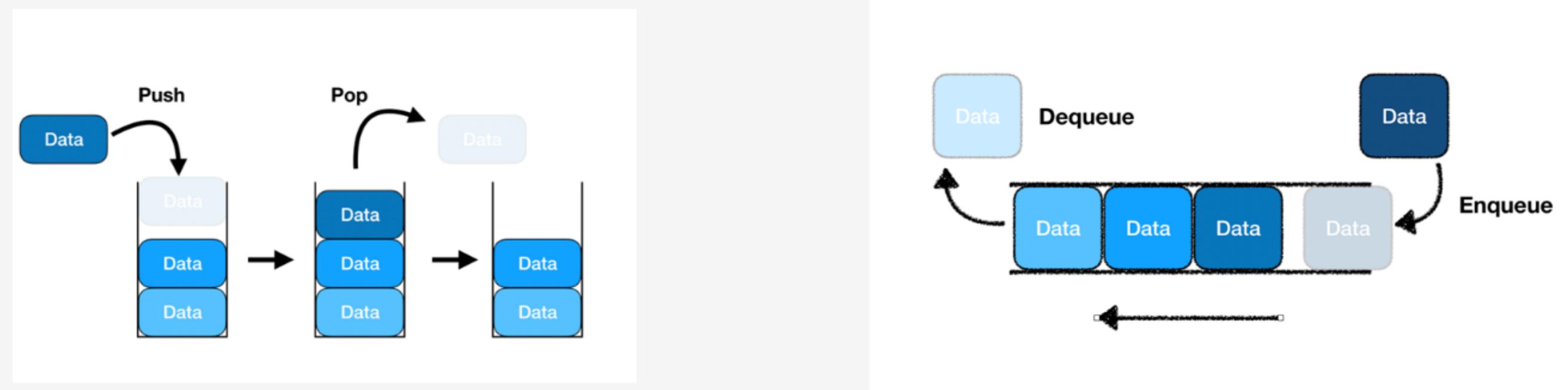
알고리즘들을 모아서 다양한 문제를 해결하는  
소프트웨어로 만들수 있습니다



스택, 큐, 데크

# 스택, 큐, 덱

자료를 저장할 수 있고, 도구로 활용할 수 있는 선형 자료 구조의 일종

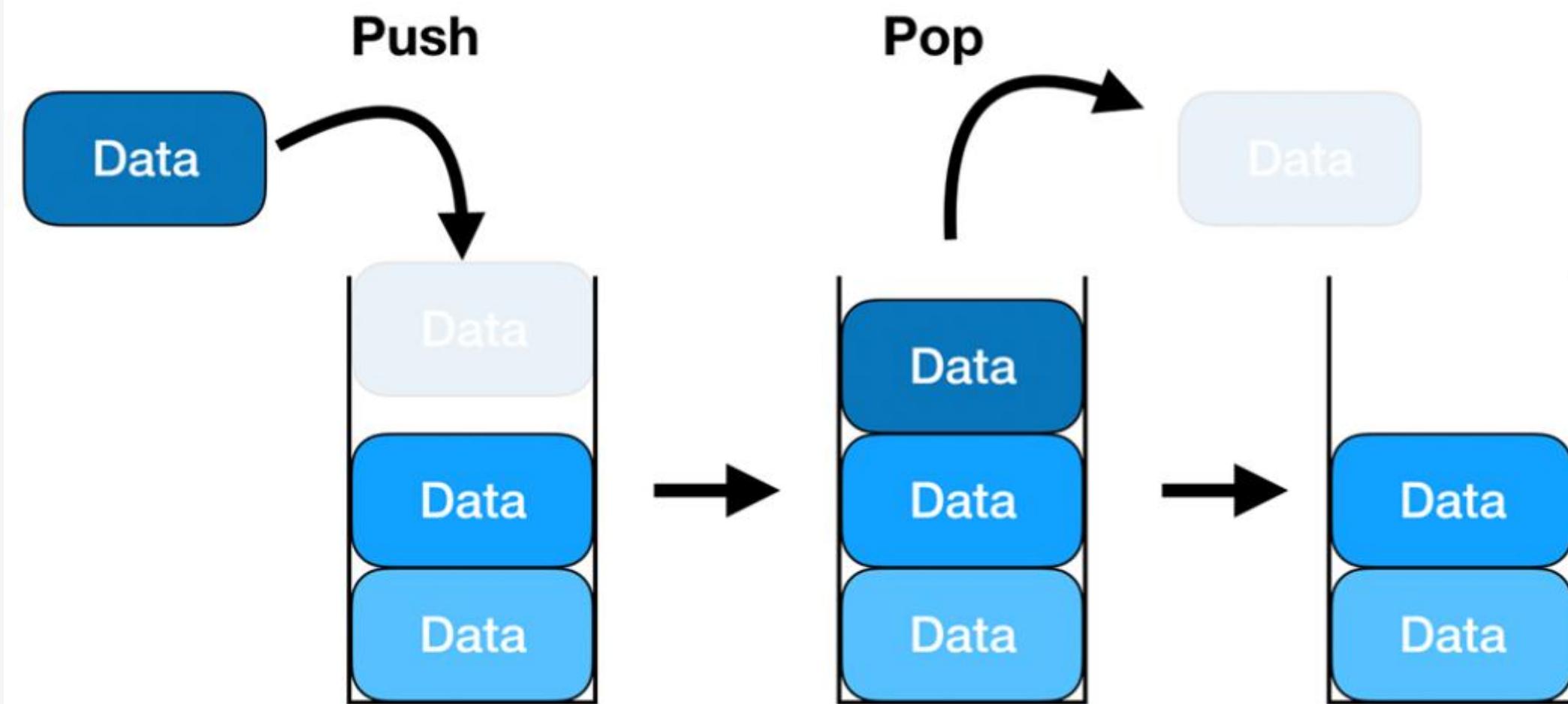




스택

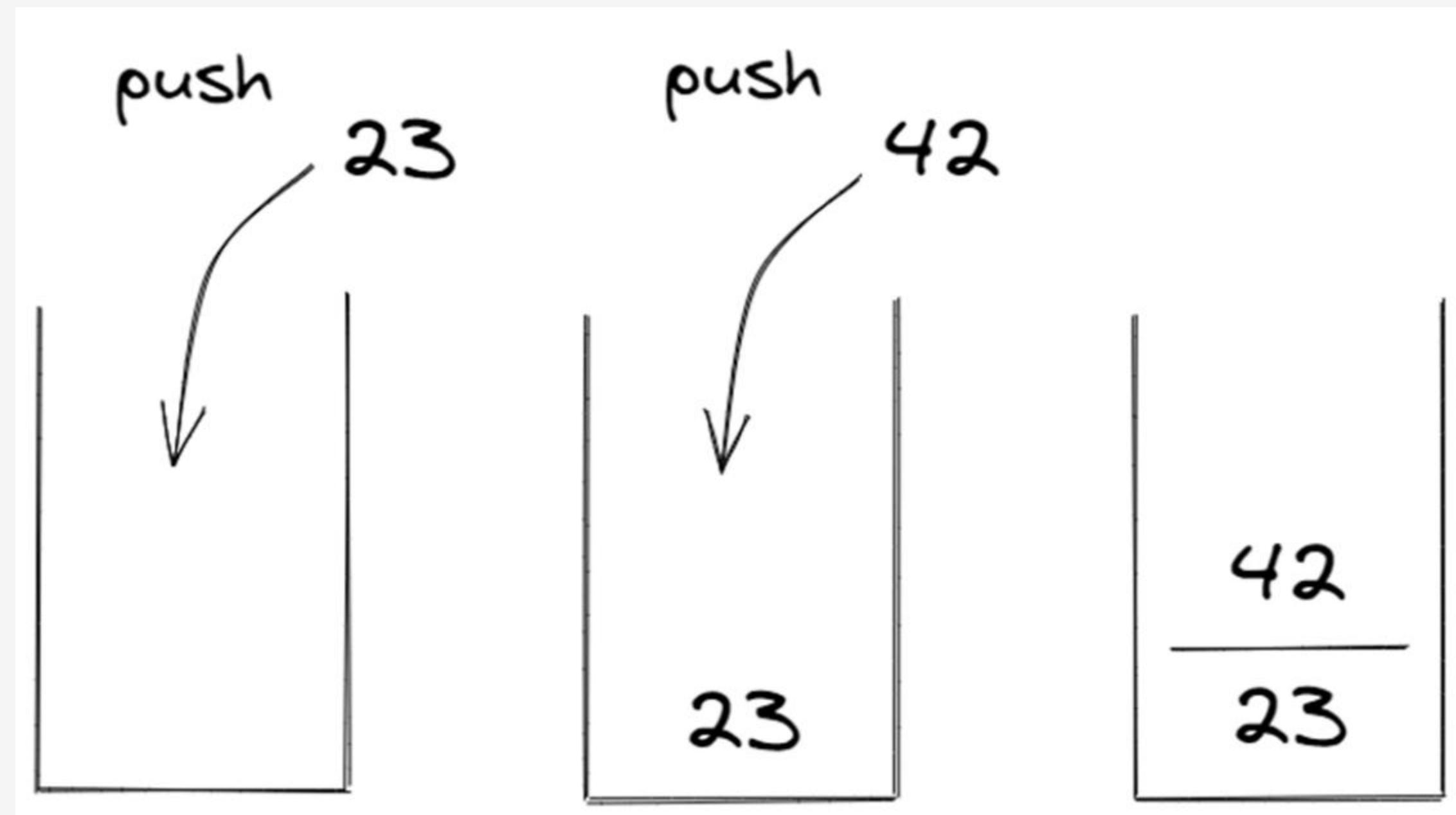
# 스택

한 쪽 끝에서만 자료를 넣거나 뺄 수 있는 자료 구조.



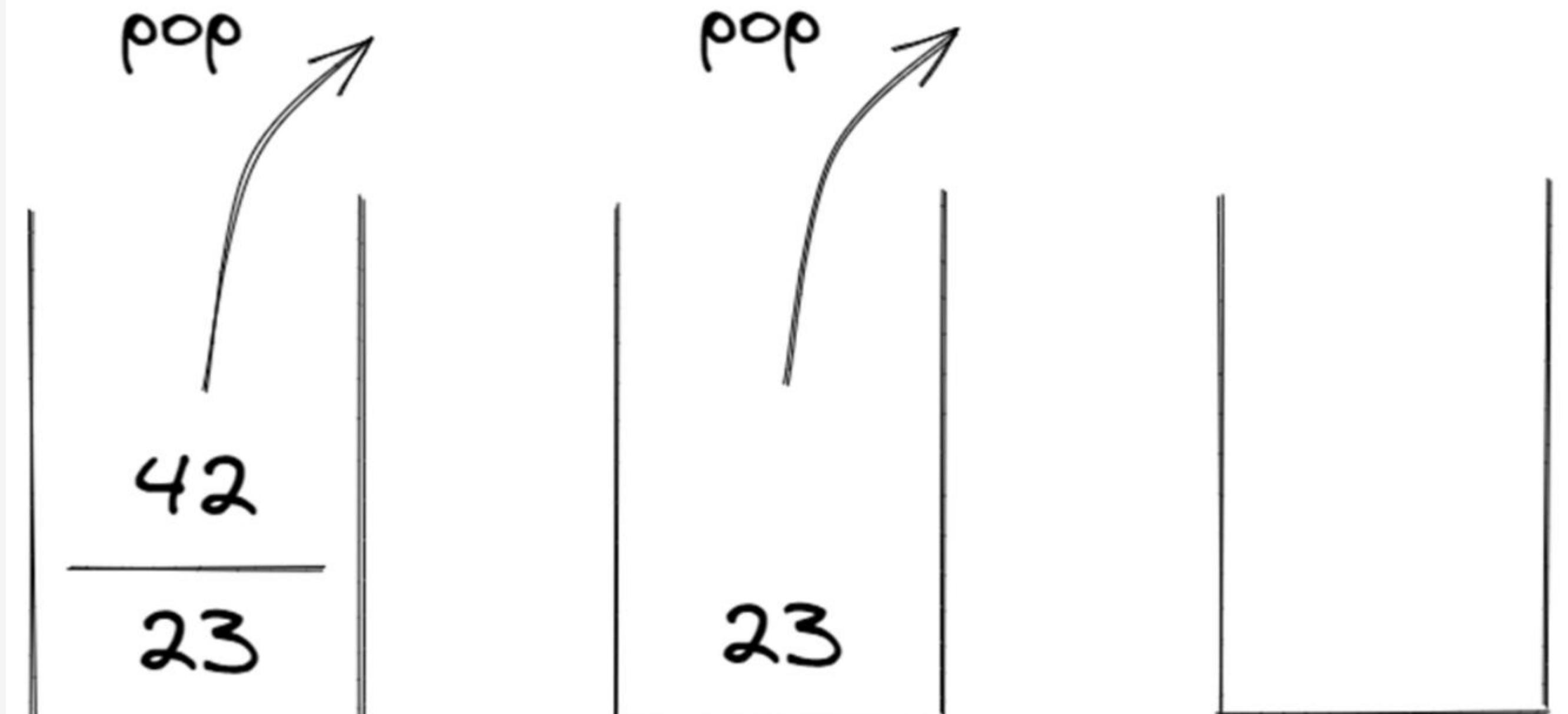
# 스택

1. Push 연산: 자료를 넣는다.



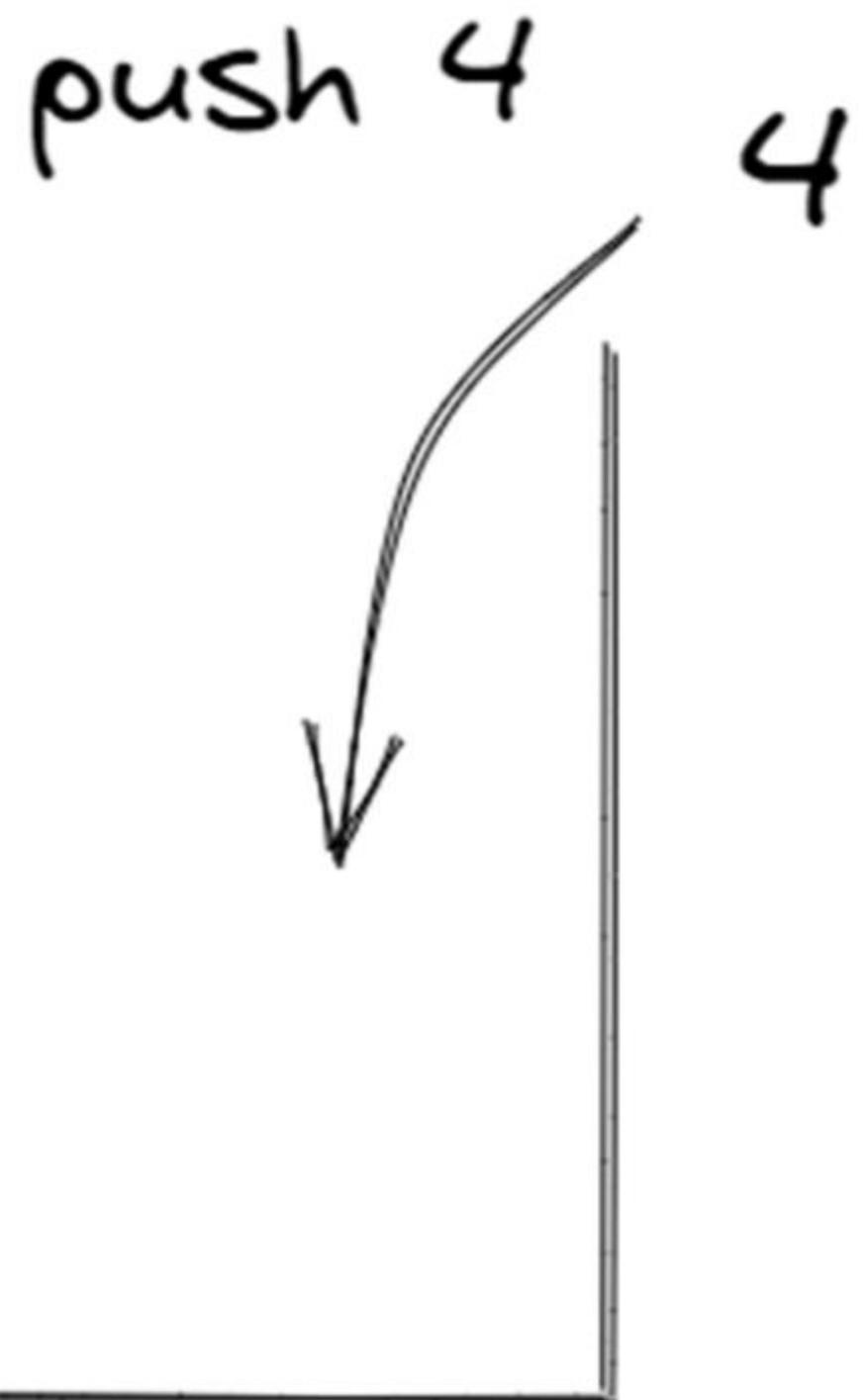
# 스택

2. Pop 연산: 가장 최근에 넣은 자료를 뺀다.



# 스택 그려보기

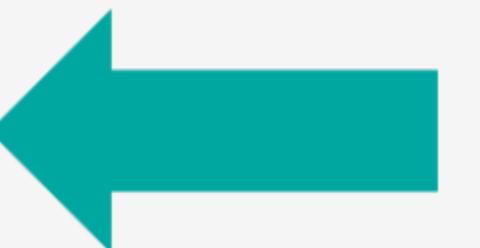
index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



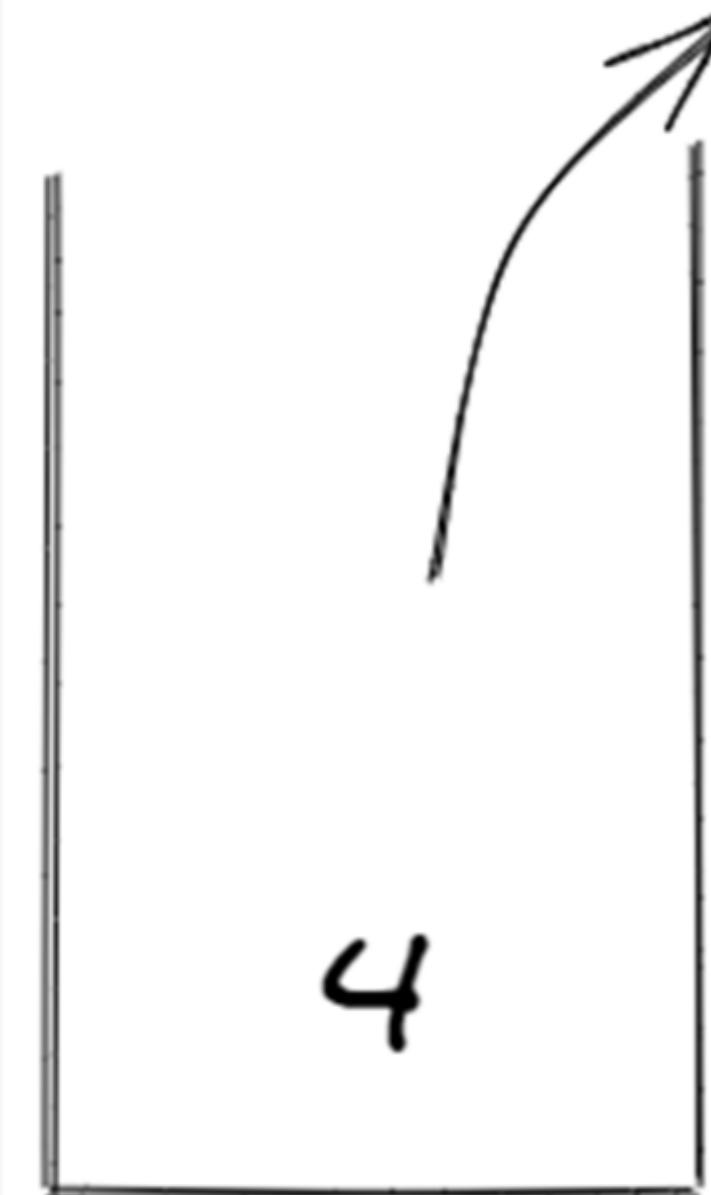
# 스택 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop

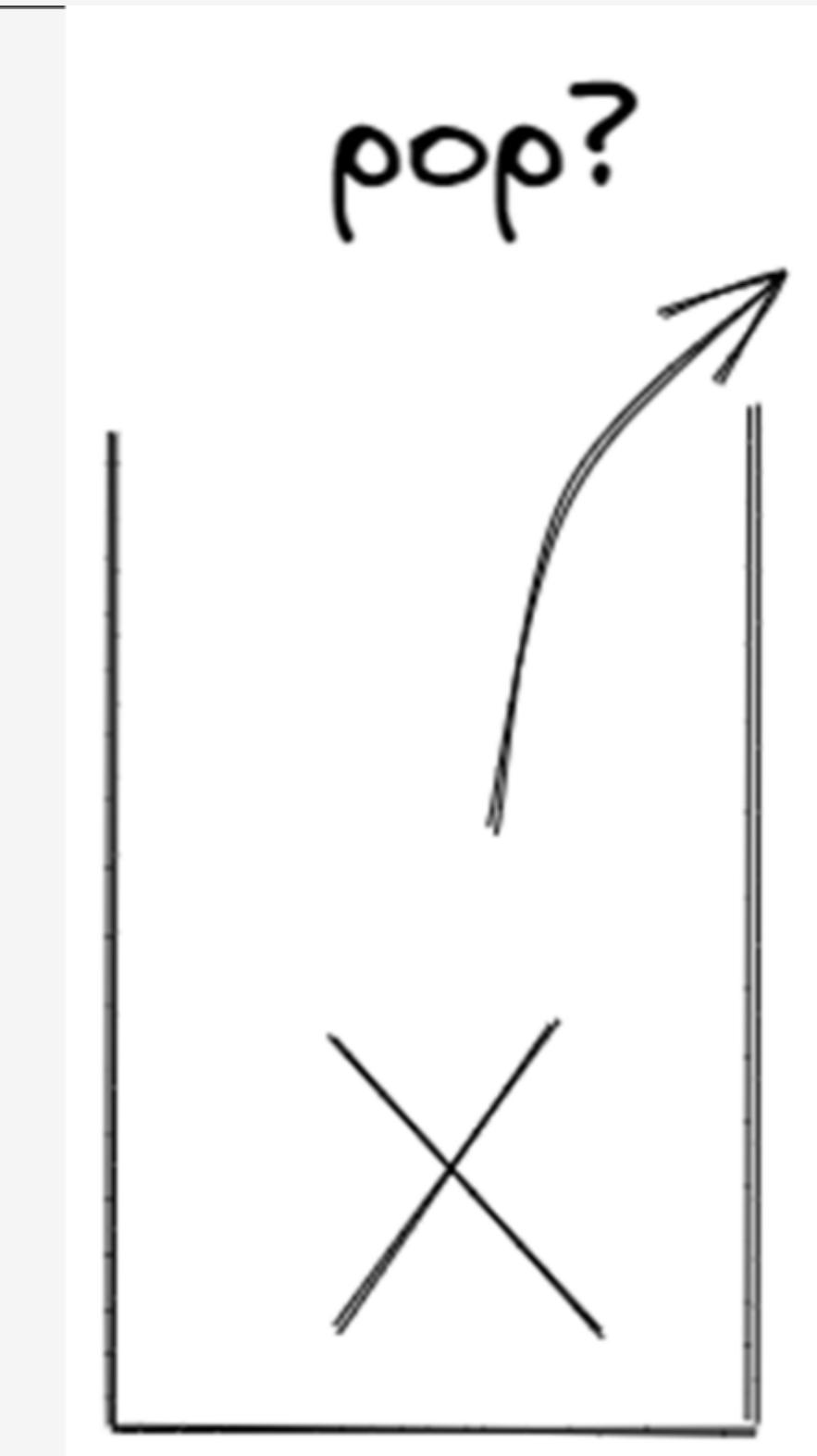


POP



# 스택 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 스택 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 스택 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop

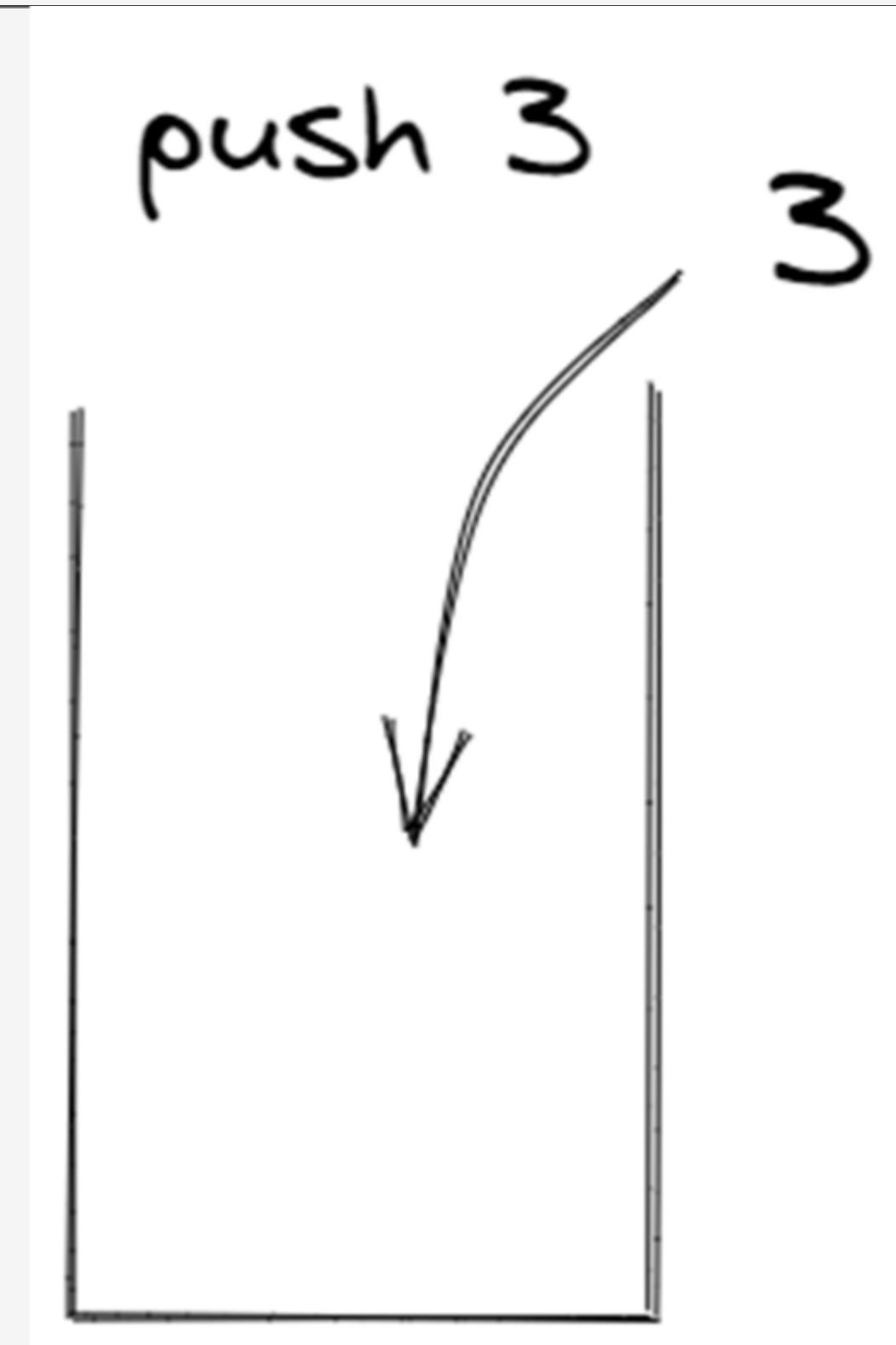
pop



# 스택 그려보기



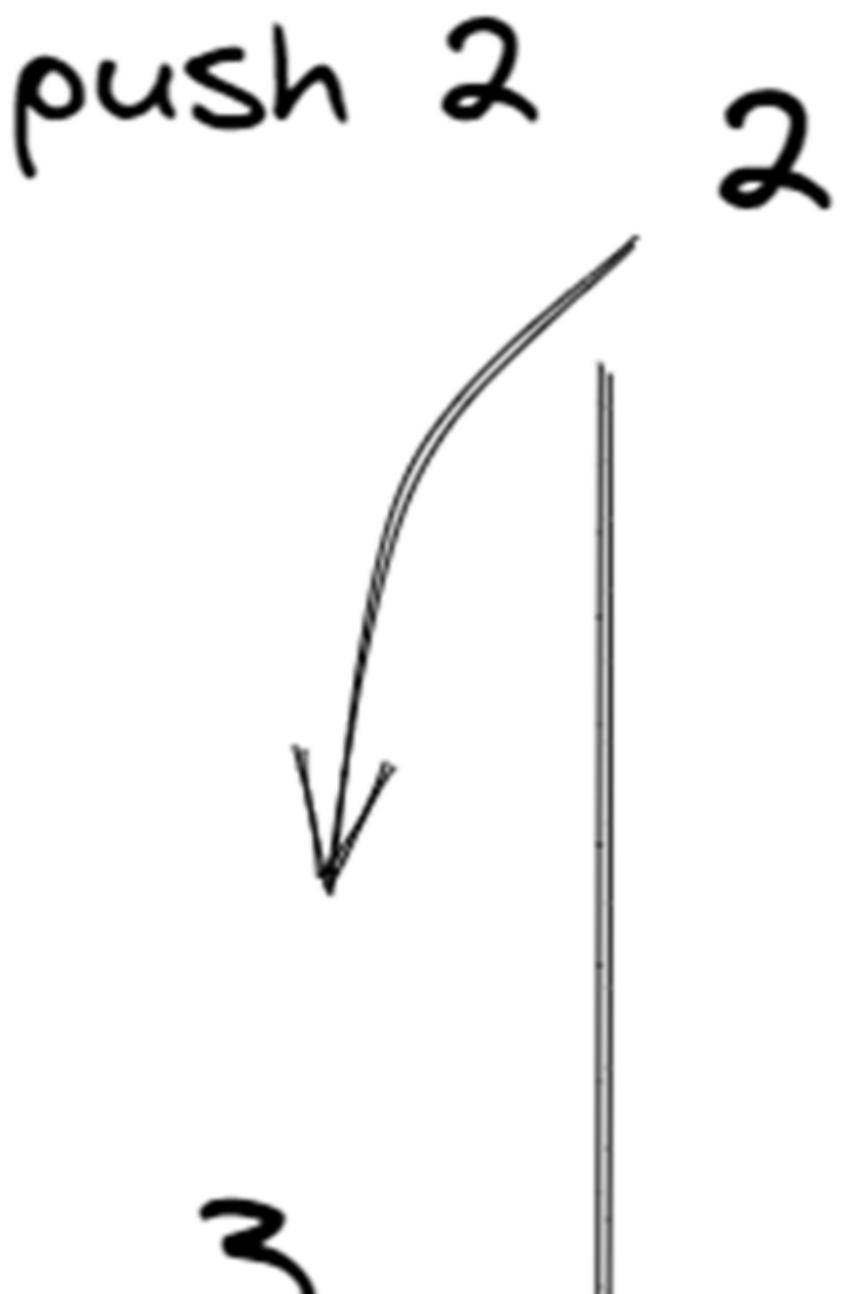
index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 스택 그려보기



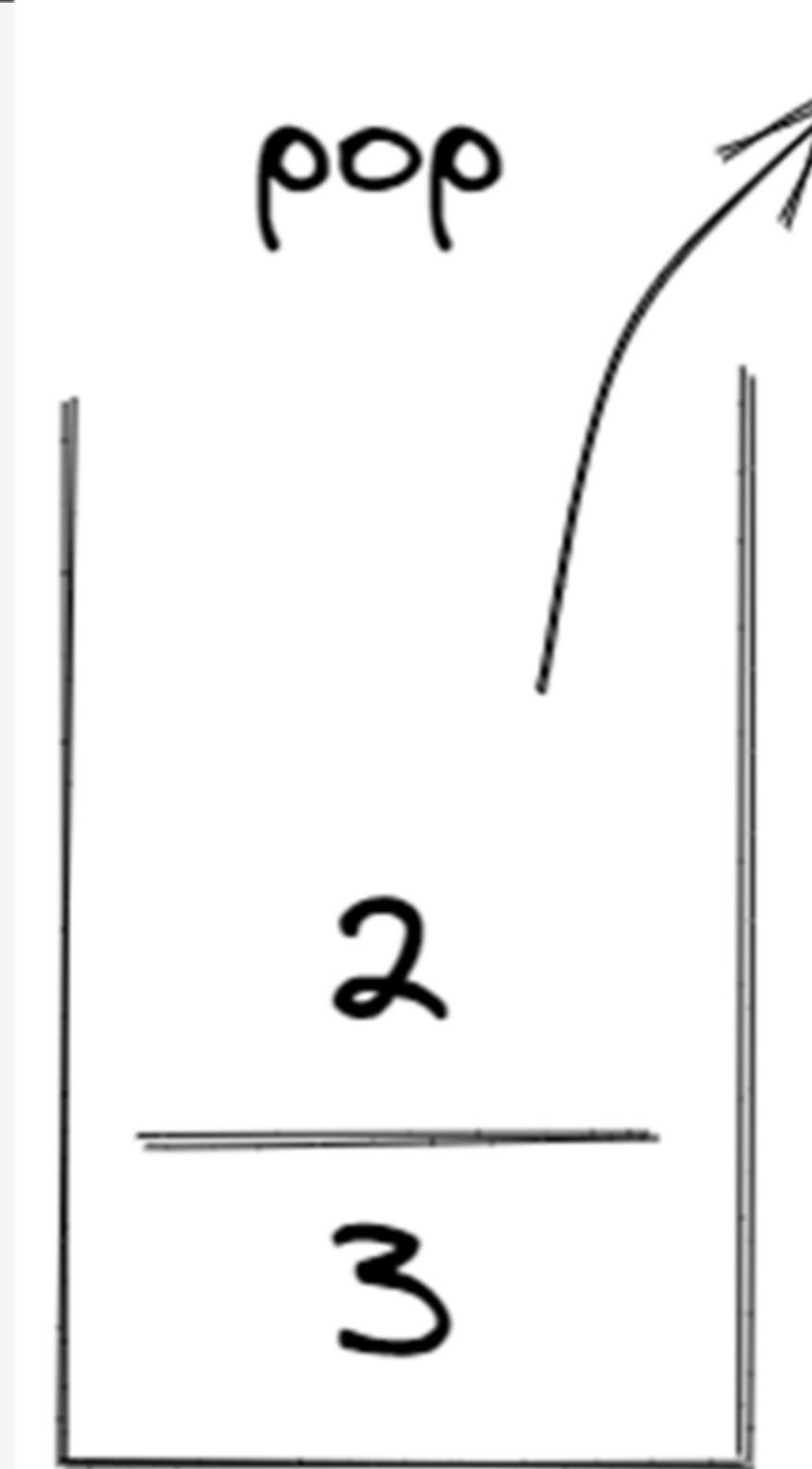
index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 스택 그려보기



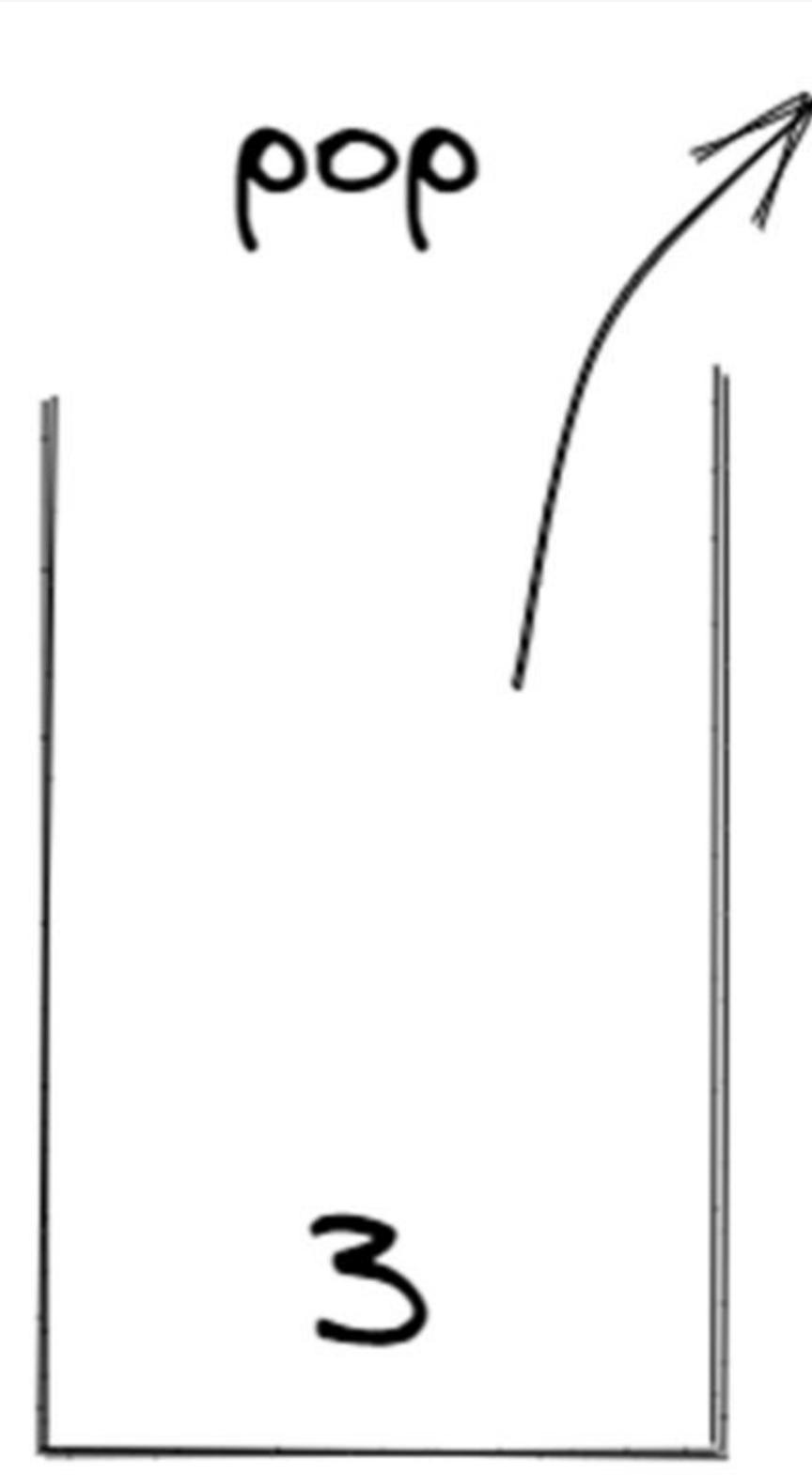
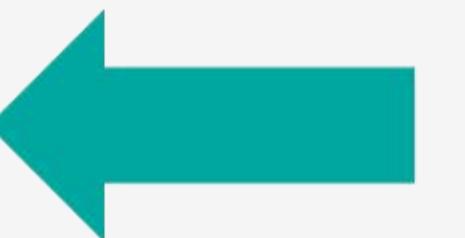
index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 스택 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



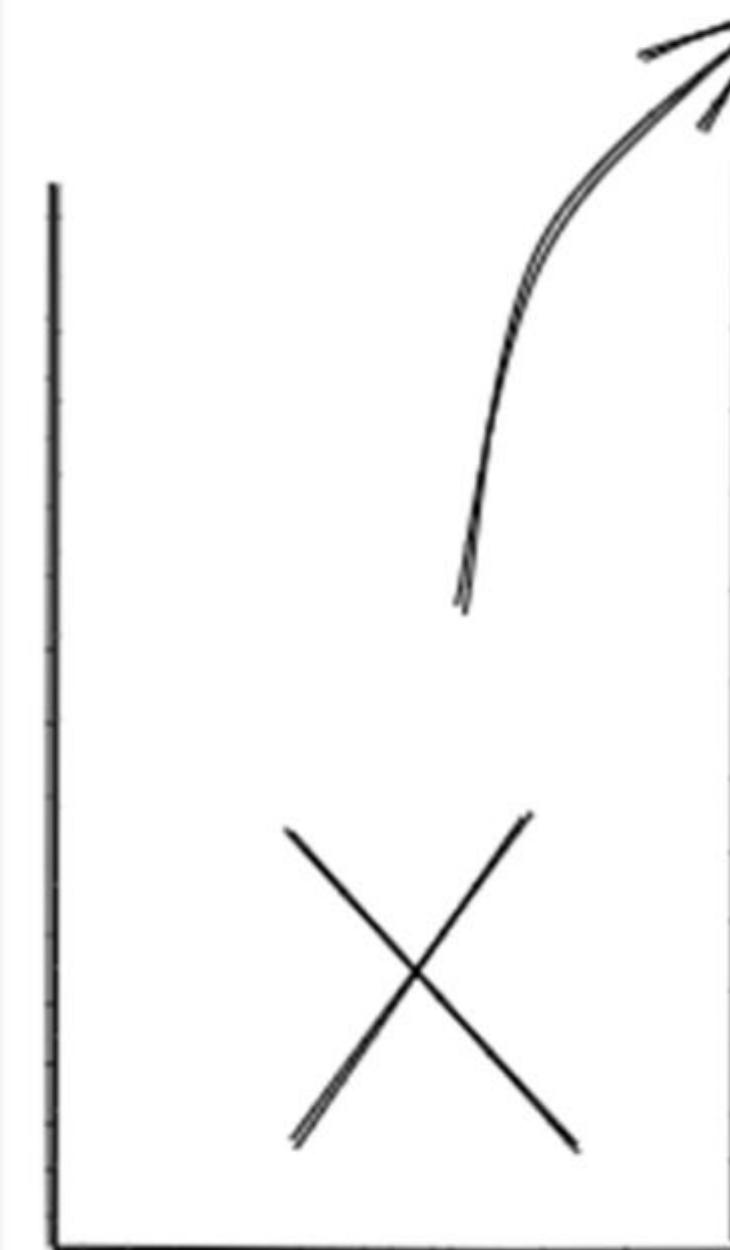
# 스택 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



pop?



# 스택 구현체 사용하기

---



Q. 스택을 어떻게 구현하나요?

A. push와 pop만 있다면 스택입니다.

# 스택 구현체 사용하기

---



## 각 언어 별 스택 구현체

Java: Stack 클래스

```
Stack<Integer> st = new Stack<>();
```

```
// push 42
st.push(42);
```

```
// pop
st.pop();
```

# 스택 구현체 사용하기



## 각 언어 별 스택 구현체

Python: 리스트

```
# push 42  
stack.append(42)
```

```
# pop  
stack.pop()
```

```
# top  
stack[-1]
```

```
# isEmpty  
if not stack:  
    pass
```

# 스택 구현체 사용하기

---

스택을 사용해봅시다!



# 스택 구현체 사용하기



균형잡힌 세상

성공  
다국어

☆ 한국어 ▾

4 실버 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	128 MB	169441	58353	44889	33.107%

## 문제

세계는 균형이 잘 잡혀있어야 한다. 양과 음, 빛과 어둠 그리고 왼쪽 괄호와 오른쪽 괄호처럼 말이다.

정민이의 임무는 어떤 문자열이 주어졌을 때, 괄호들의 균형이 잘 맞춰져 있는지 판단하는 프로그램을 짜는 것이다.

문자열에 포함되는 괄호는 소괄호("(") 와 대괄호("["")로 2종류이고, 문자열이 균형을 이루는 조건은 아래와 같다.

- 모든 왼쪽 소괄호("(")는 오른쪽 소괄호(")")와만 짹을 이뤄야 한다.
- 모든 왼쪽 대괄호("[")는 오른쪽 대괄호("]")와만 짹을 이뤄야 한다.
- 모든 오른쪽 괄호들은 자신과 짹을 이룰 수 있는 왼쪽 괄호가 존재한다.
- 모든 괄호들의 짹은 1:1 매칭만 가능하다. 즉, 괄호 하나가 둘 이상의 괄호와 짹지어지지 않는다.
- 짬을 이루는 두 괄호가 있을 때, 그 사이에 있는 문자열도 균형이 잡혀야 한다.

정민이를 도와 문자열이 주어졌을 때 균형잡힌 문자열인지 아닌지를 판단해보자.



# 스택 구현체 사용하기

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Main {
5     public static void main(String[] args) throws IOException {
6         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
7         BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out));
8         String[] line = br.readLine().split("");
9
10        while (!line[0].equals(".")){
11            Stack<String> stack = new Stack<>();
12            boolean check = true;
13            for(String s : line){
14                if(s.equals("(")) stack.push("(");
15                if(s.equals("[")) stack.push("[");
16                if(s.equals(")")){
17                    if(stack.isEmpty()||!stack.pop().equals("(")){
18                        bw.write("no\n");
19                        check =false;
20                        break;
21                    }
22                }
23            }
24        }
25    }
26}
```

```
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44 }
```



# 스택 구현체 사용하기

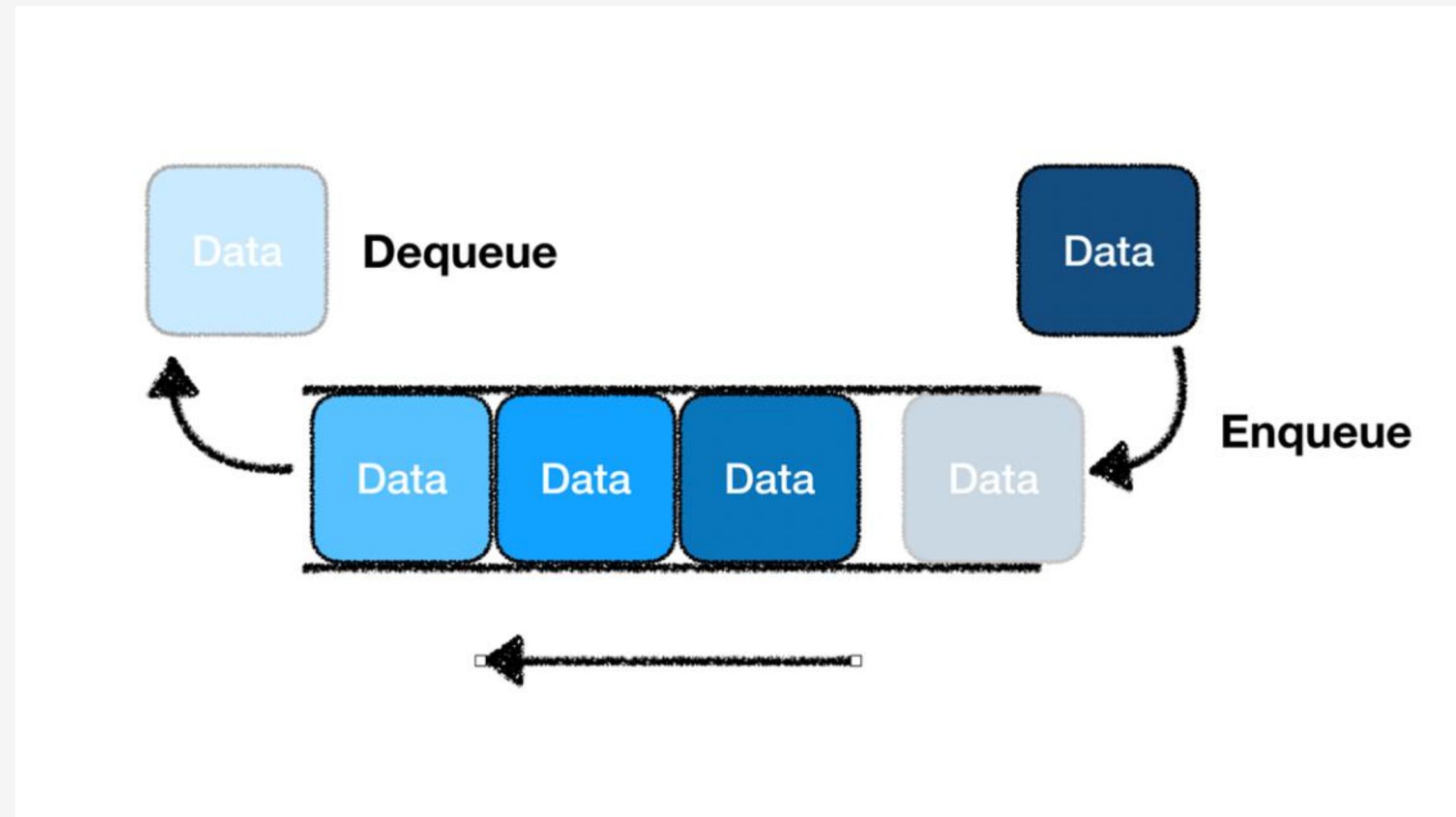
```
1 import sys
2
3 stack = []
4
5 while True:
6     line = sys.stdin.readline().rstrip()
7     if line == ".":
8         break
9
10    stack.clear()
11    check = True
12
13    for char in line:
14        if char == "(":
15            stack.append("(")
16        elif char == "[":
17            stack.append("[")
18        elif char == ")":
19            if not stack or stack.pop() != "(":
20                print("no")
21                check = False
22                break
23            elif char == "]":
24                if not stack or stack.pop() != "[":
25                    print("no")
26                    check = False
27                    break
28
29    if check:
30        if stack:
31            print("no")
32        else:
33            print("yes")
34
```



큐

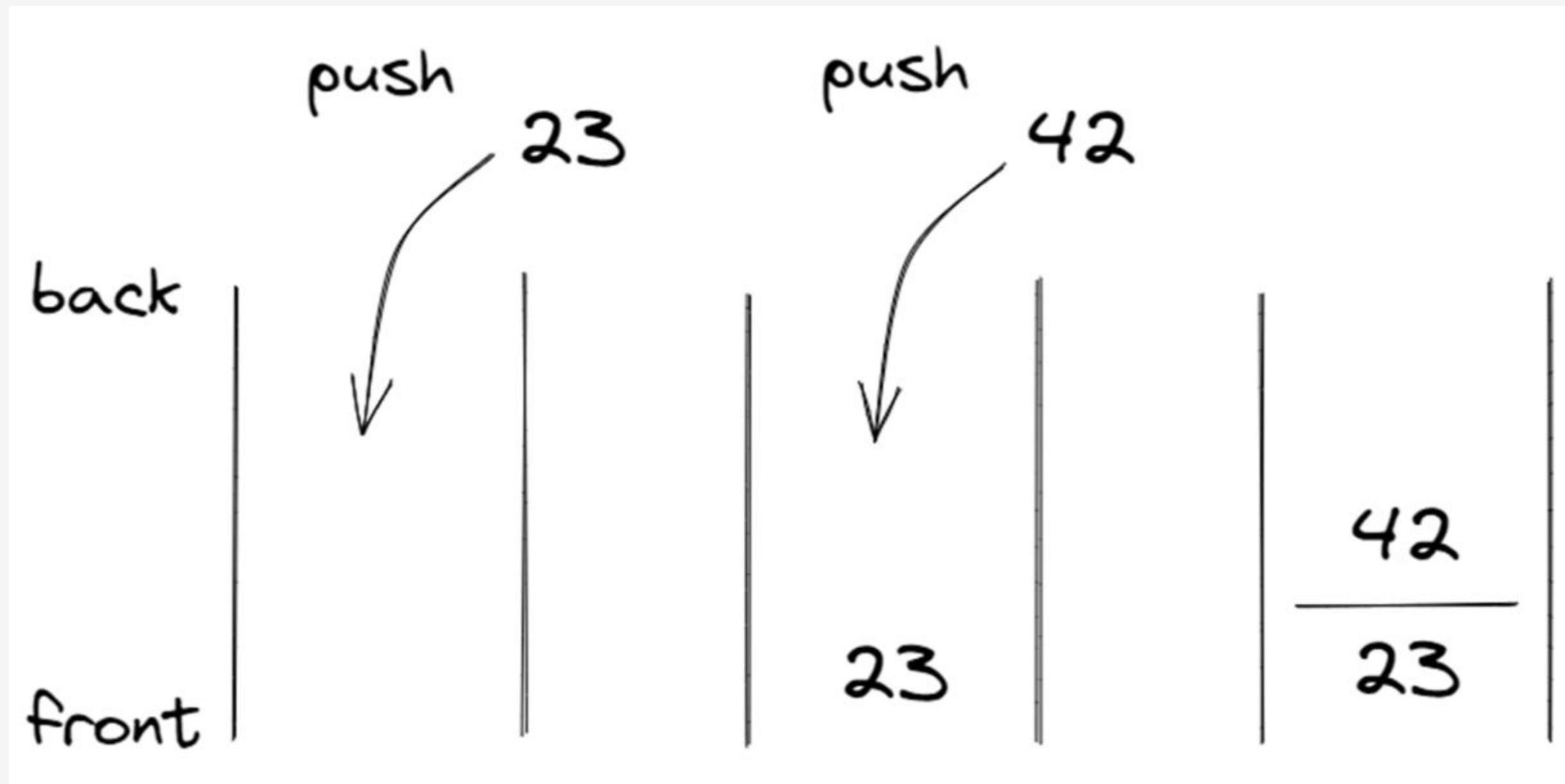
# 큐

한 쪽 끝에서만 자료를 넣을 수 있고, 다른 한쪽 끝에서만 뺄 수 있는 자료 구조.



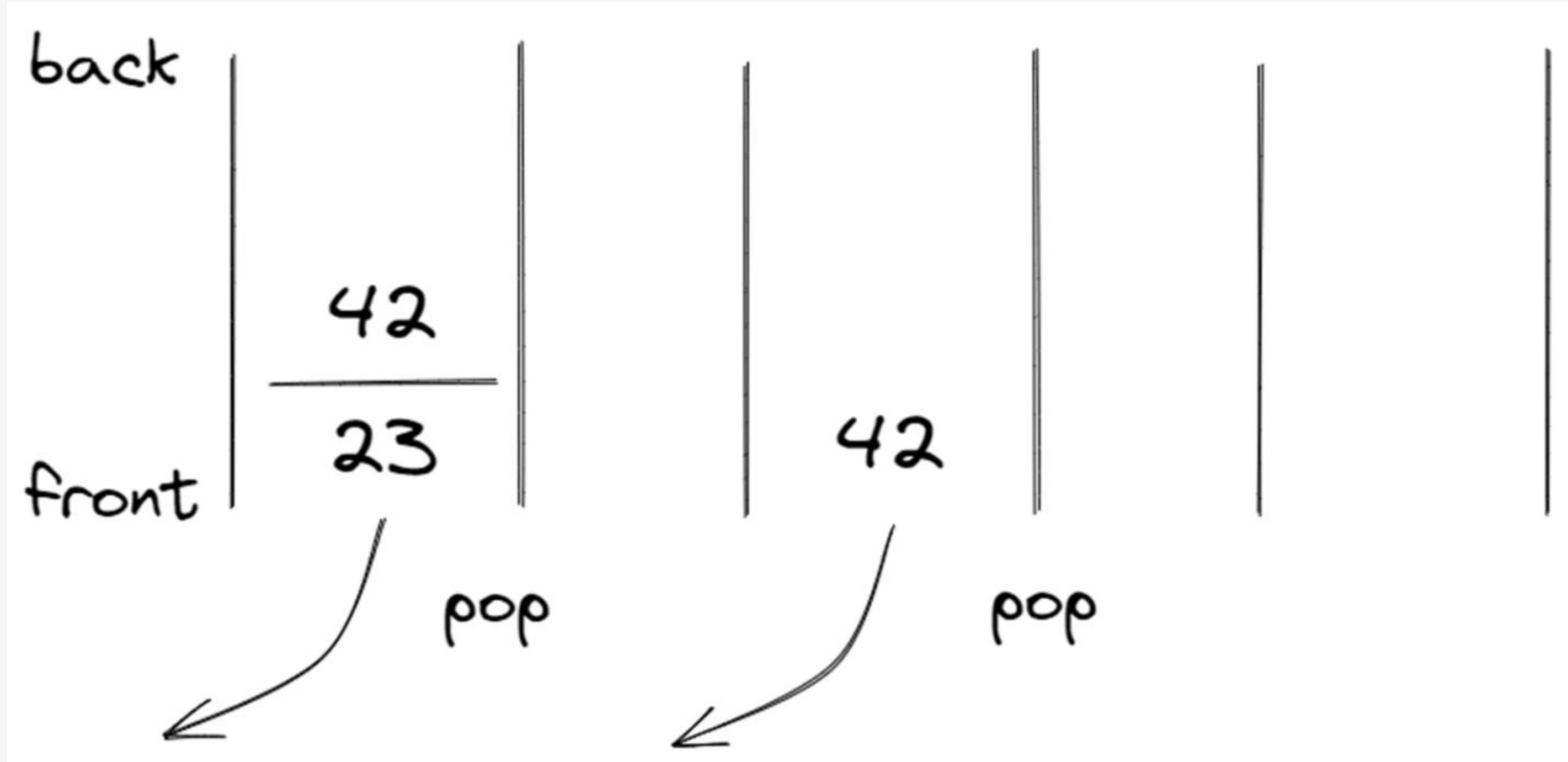
# 큐

1. Push 연산: 자료를 넣는다.



# 큐

2. Pop 연산: 가장 먼저 넣은 자료를 뺀다.



# 큐 그려보기

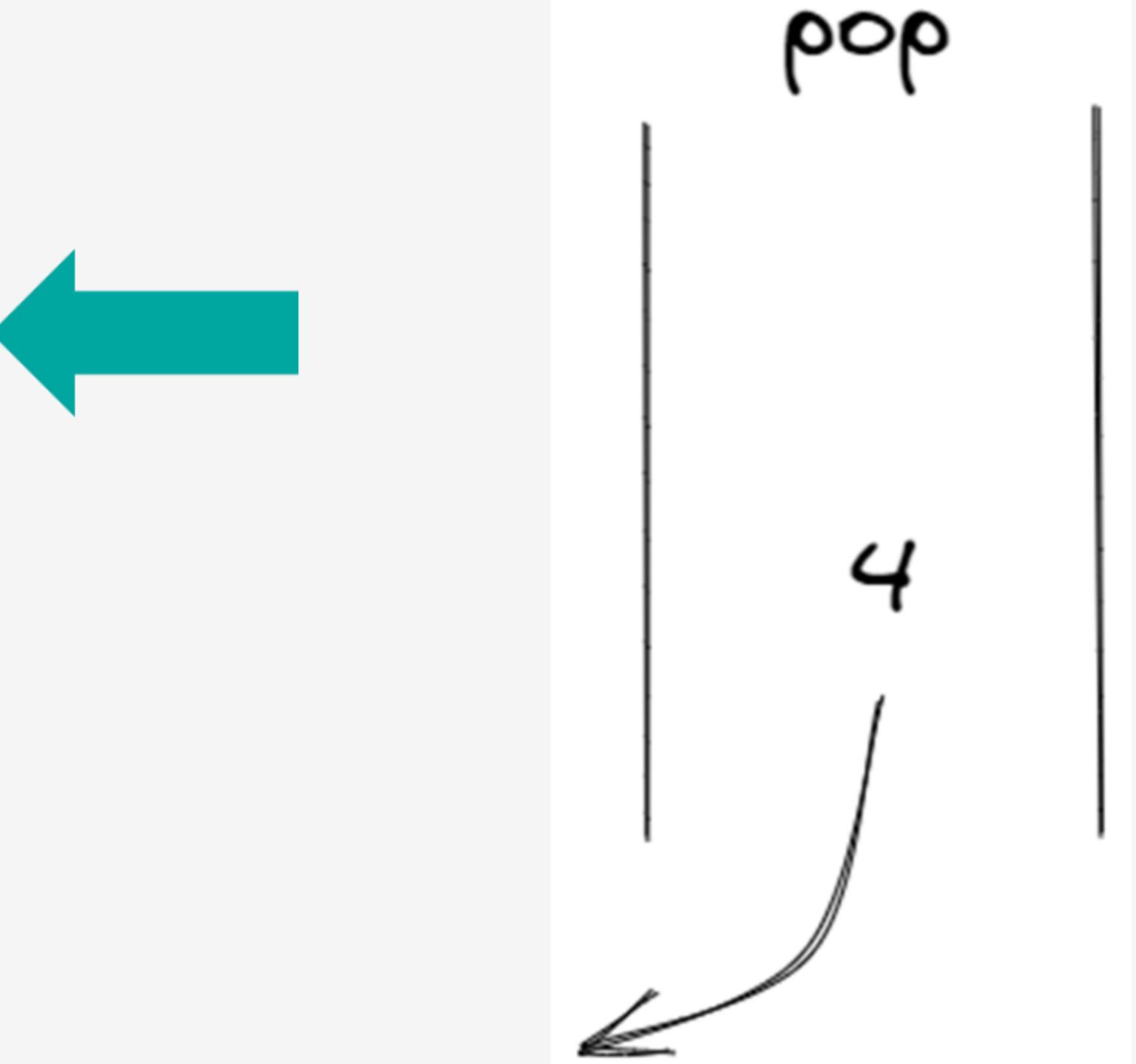
index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 큐 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



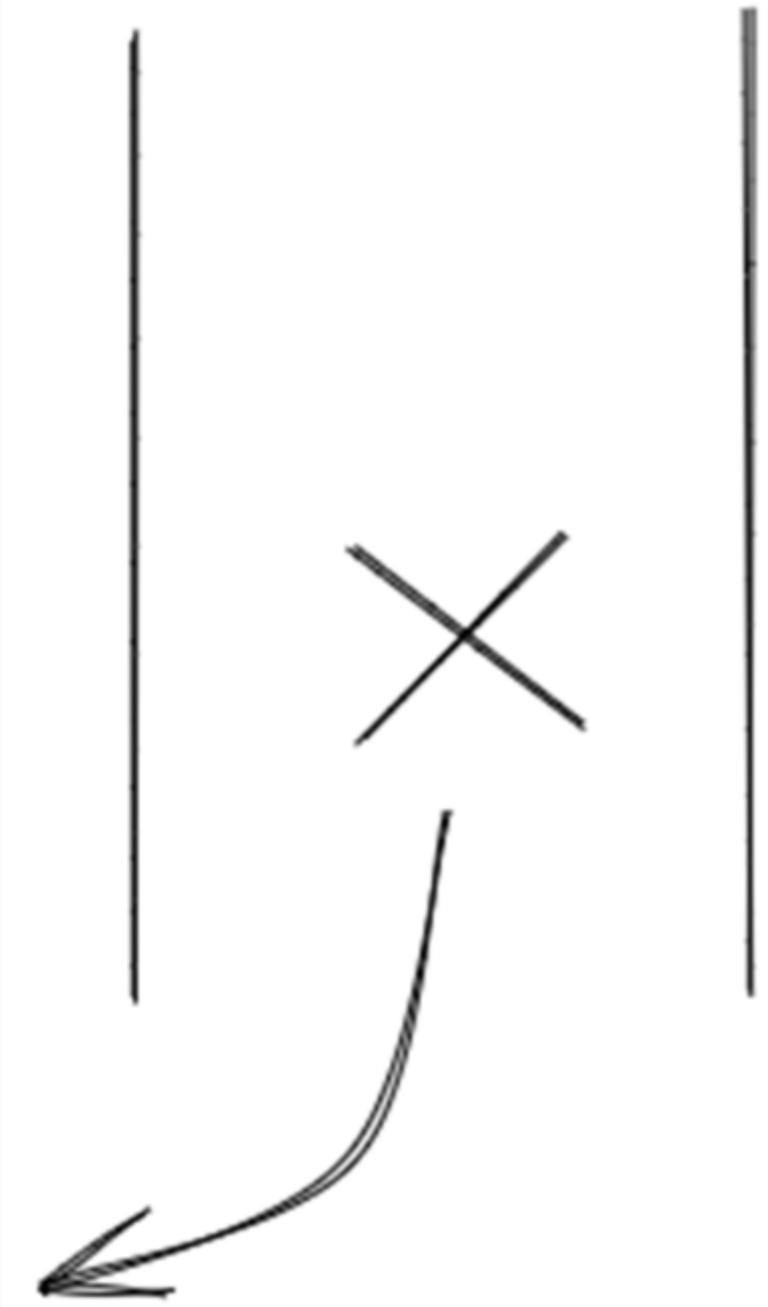
# 큐 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



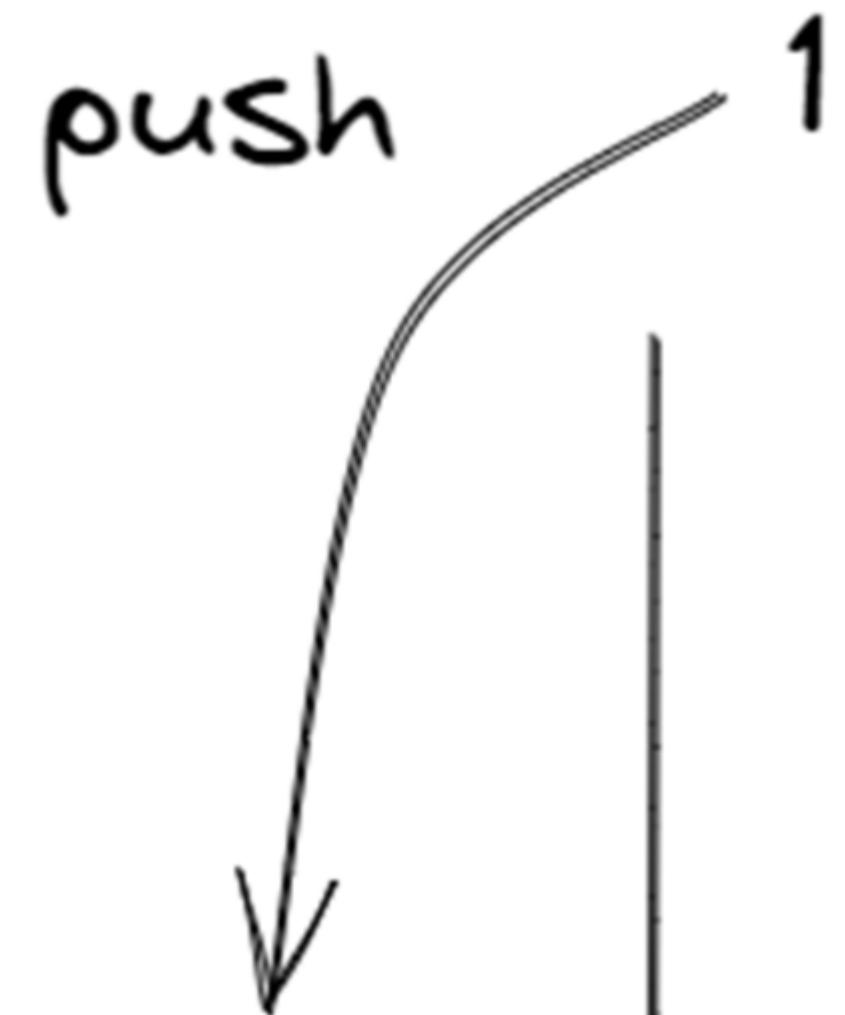
pop?



# 큐 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



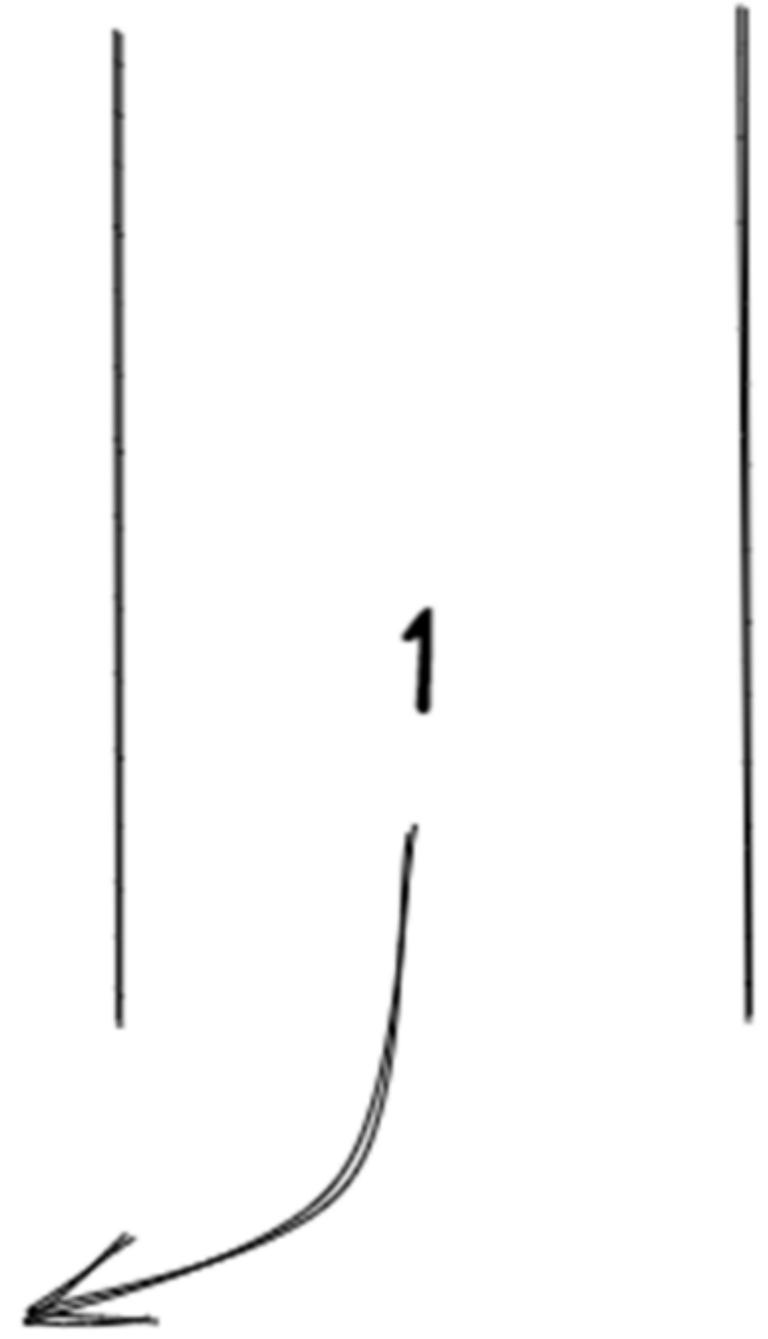
# 큐 그려보기



index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



pop



# 큐 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop

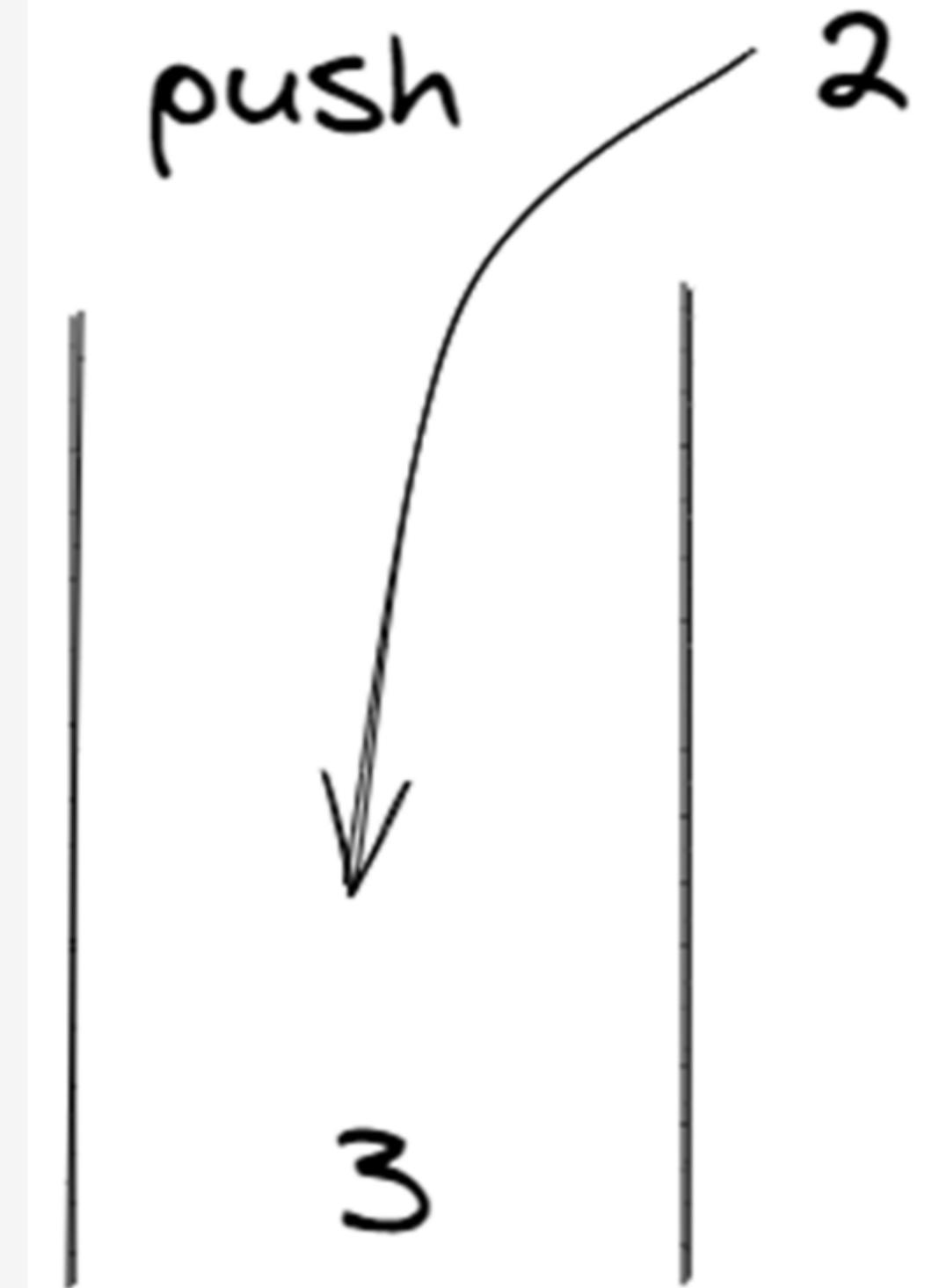


push 3



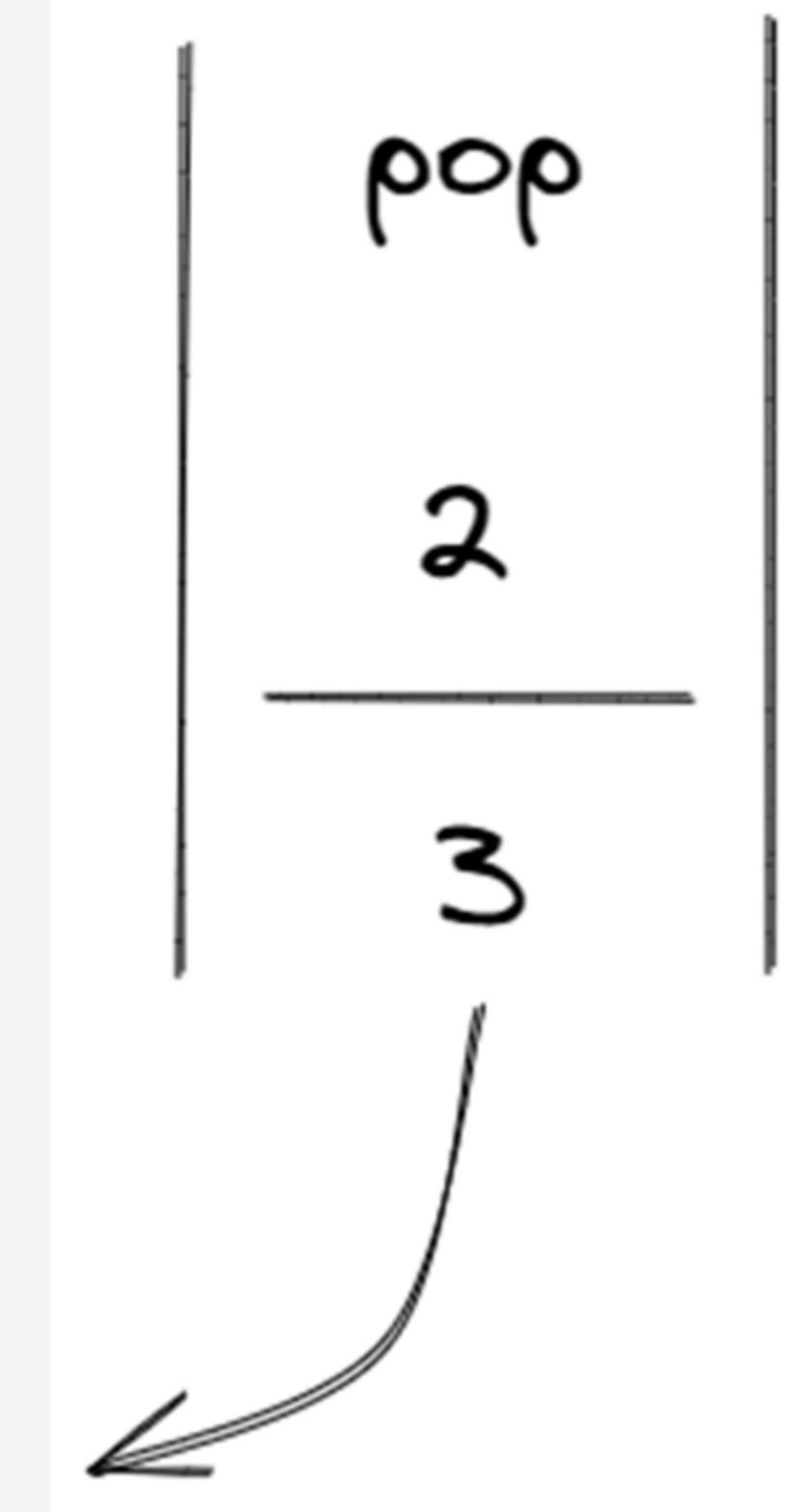
# 큐 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



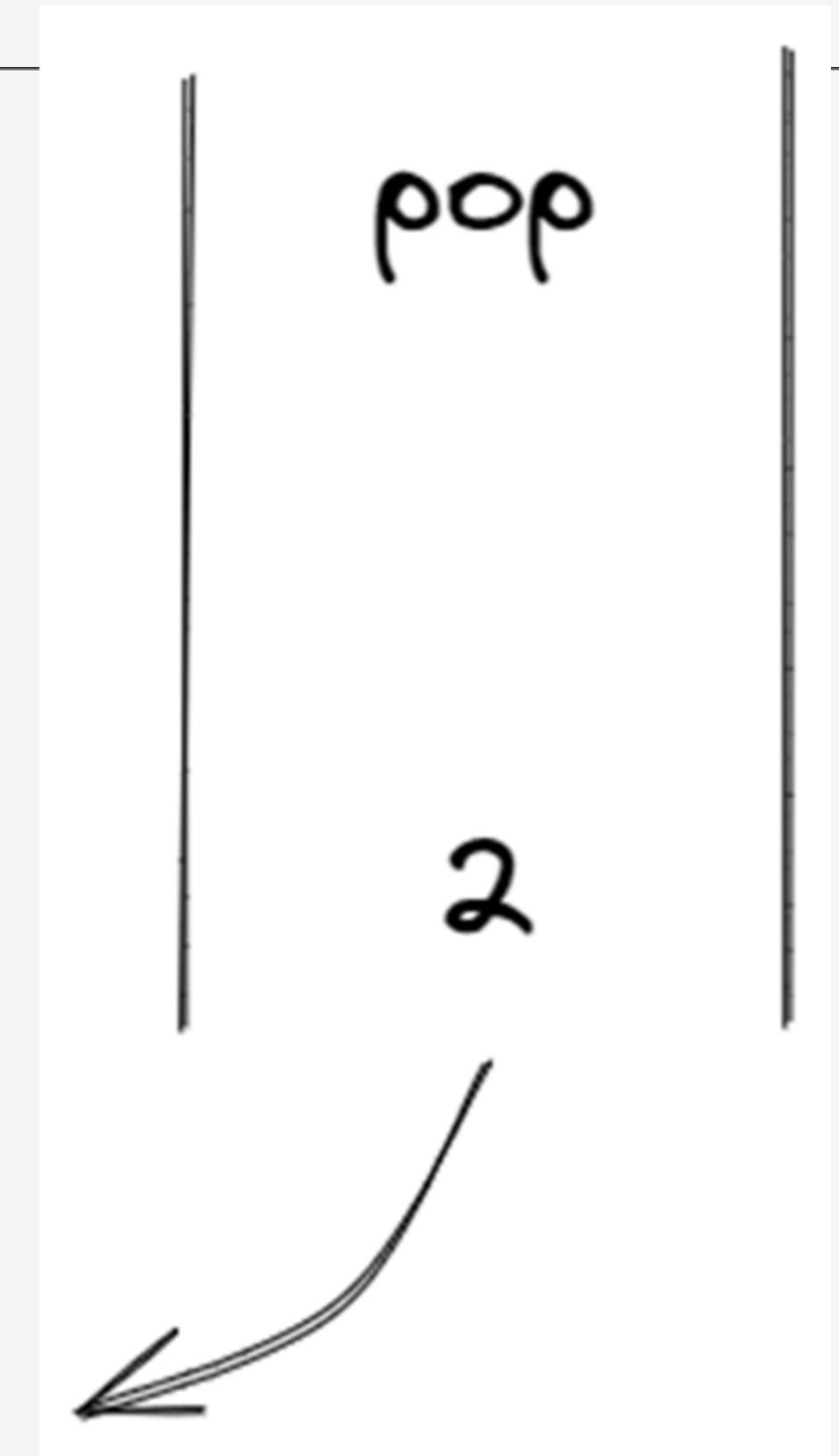
# 큐 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



# 큐 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop

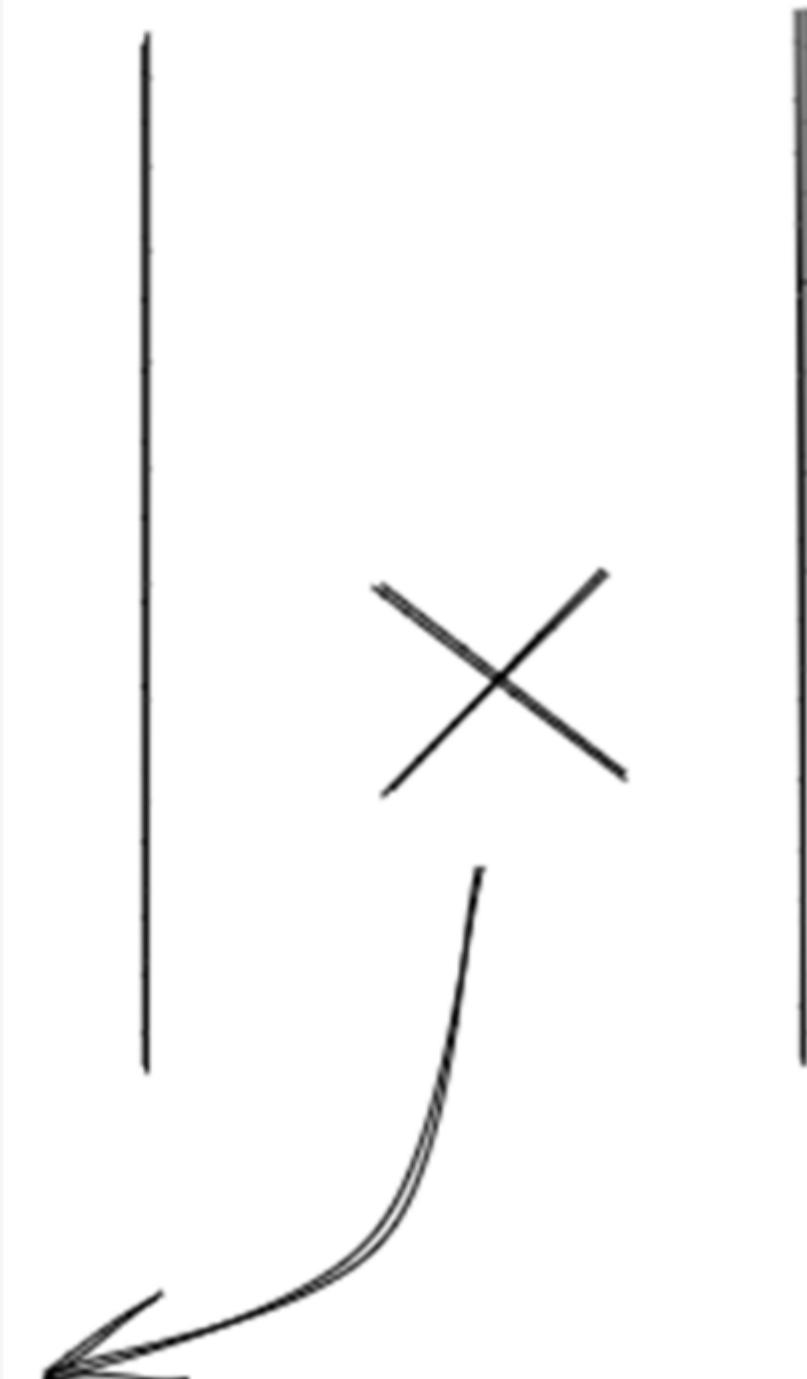


# 큐 그려보기

index	operation
1	push 4
2	pop
3	pop
4	push 1
5	pop
6	push 3
7	push 2
8	pop
9	pop
10	pop



pop?



# 큐 구현체 사용하기

---



Q. 큐를 어떻게 구현하나요?

A. push와 pop만 있다면 큐입니다.

# 큐 구현체 사용하기

---



## 각 언어 별 큐 구현체

Java: Queue 클래스

```
Queue<Integer> queue = new LinkedList<>();  
// push 42  
queue.offer(42);
```

```
// pop  
queue.poll();
```

# 큐 구현체 사용하기



## 각 언어 별 큐 구현체

Python: deque 클래스

```
from collections import deque
queue = deque()                                # front
# push 42
queue.append(42)                                # isEmpty
# pop
queue.popleft()                                 if not queue:
                                                pass
```

# 큐 구현체 사용하기

---

큐를 사용해봅시다!



# 큐 구현체 사용하기



카드2

성공



4 실버 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초 (추가 시간 없음)	128 MB	153940	80358	62210	50.993%

## 문제

N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 순서대로 카드가 놓여 있다.

이제 다음과 같은 동작을 카드가 한 장 남을 때까지 반복하게 된다. 우선, 제일 위에 있는 카드를 바닥에 버린다. 그 다음, 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮긴다.

예를 들어  $N=4$ 인 경우를 생각해 보자. 카드는 제일 위에서부터 1234의 순서로 놓여있다. 1을 버리면 234가 남는다. 여기서 2를 제일 아래로 옮기면 342가 된다. 3을 버리면 42가 되고, 4를 밑으로 옮기면 24가 된다. 마지막으로 2를 버리고 나면, 남는 카드는 4가 된다.

$N$ 이 주어졌을 때, 제일 마지막에 남게 되는 카드를 구하는 프로그램을 작성하시오.



# 큐 구현체 사용하기

```
1 import java.util.*;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner sc = new Scanner(System.in);
6         int n = sc.nextInt();
7         Queue<Integer> q = new LinkedList<>();
8         for (int i = 0; i < n; i++) {
9             q.offer(i+1);
10        }
11        while(q.size() !=1){
12            q.poll();
13            q.offer(q.poll());
14        }
15        System.out.println(q.poll());
16    }
17 }
```



# 큐 구현체 사용하기

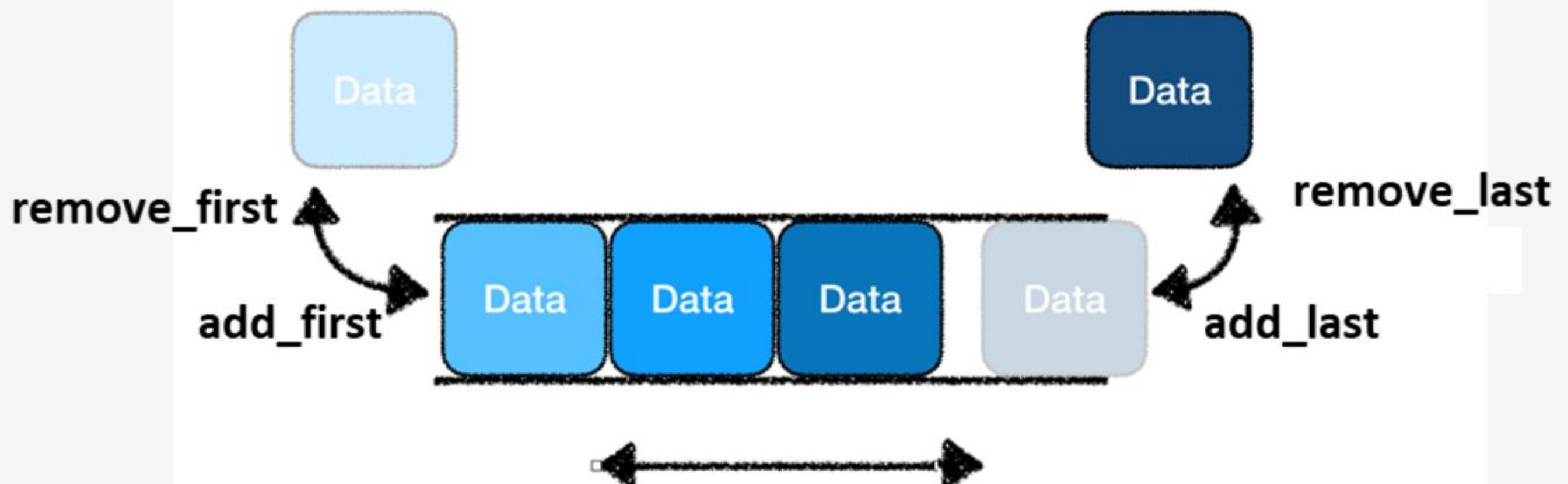
```
1 from collections import deque  
2  
3 n = int(input())  
4 q = deque()  
5 for i in range(1, n + 1):  
6     q.append(i)  
7  
8 while len(q) != 1:  
9     q.popleft()  
10    q.append(q.popleft())  
11  
12 print(q[0])
```



렉

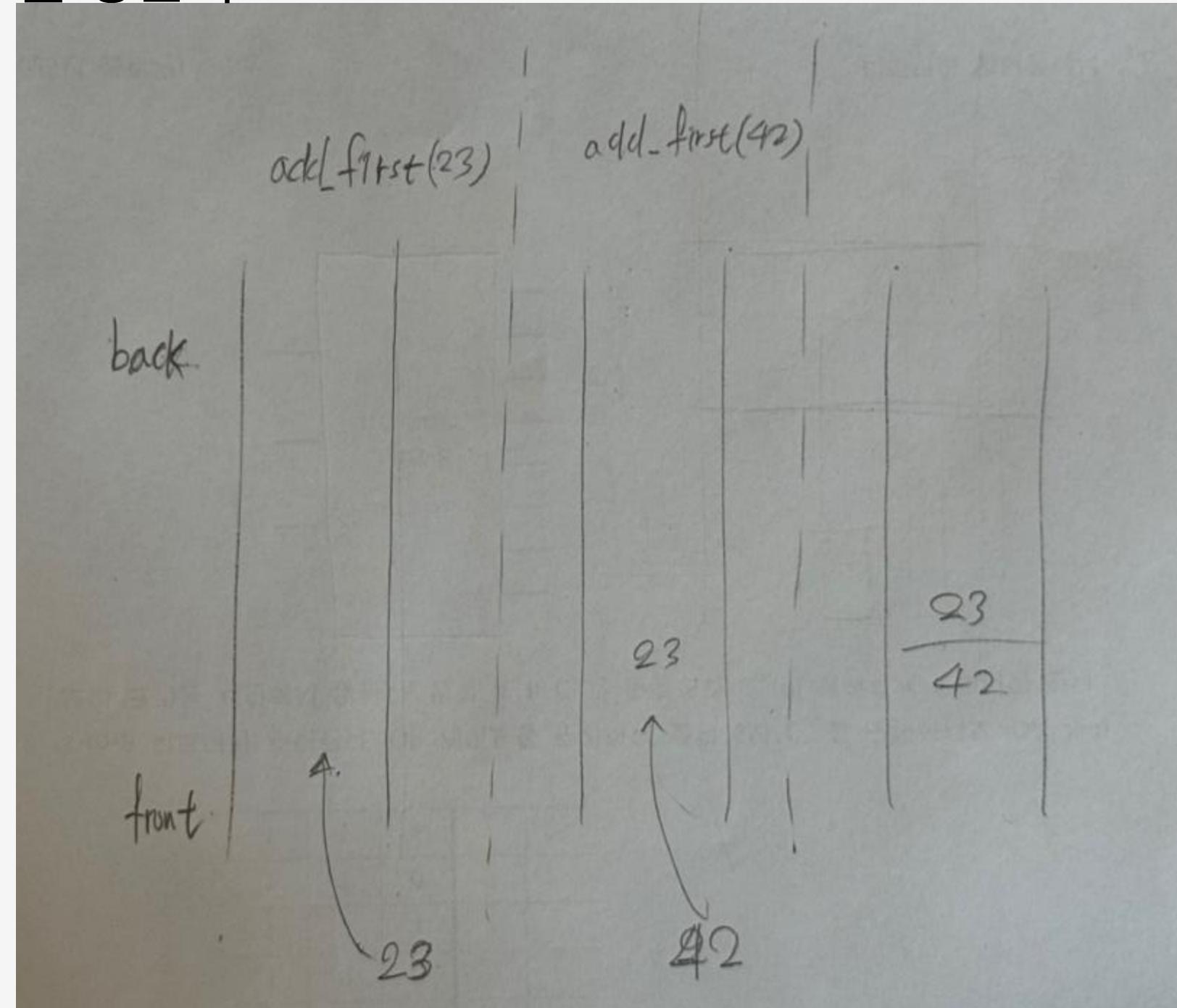


양 쪽 끝에서 자료를 넣을 수 있고, 양 쪽 끝에서 자료를 뺄 수 있는 자료 구조.



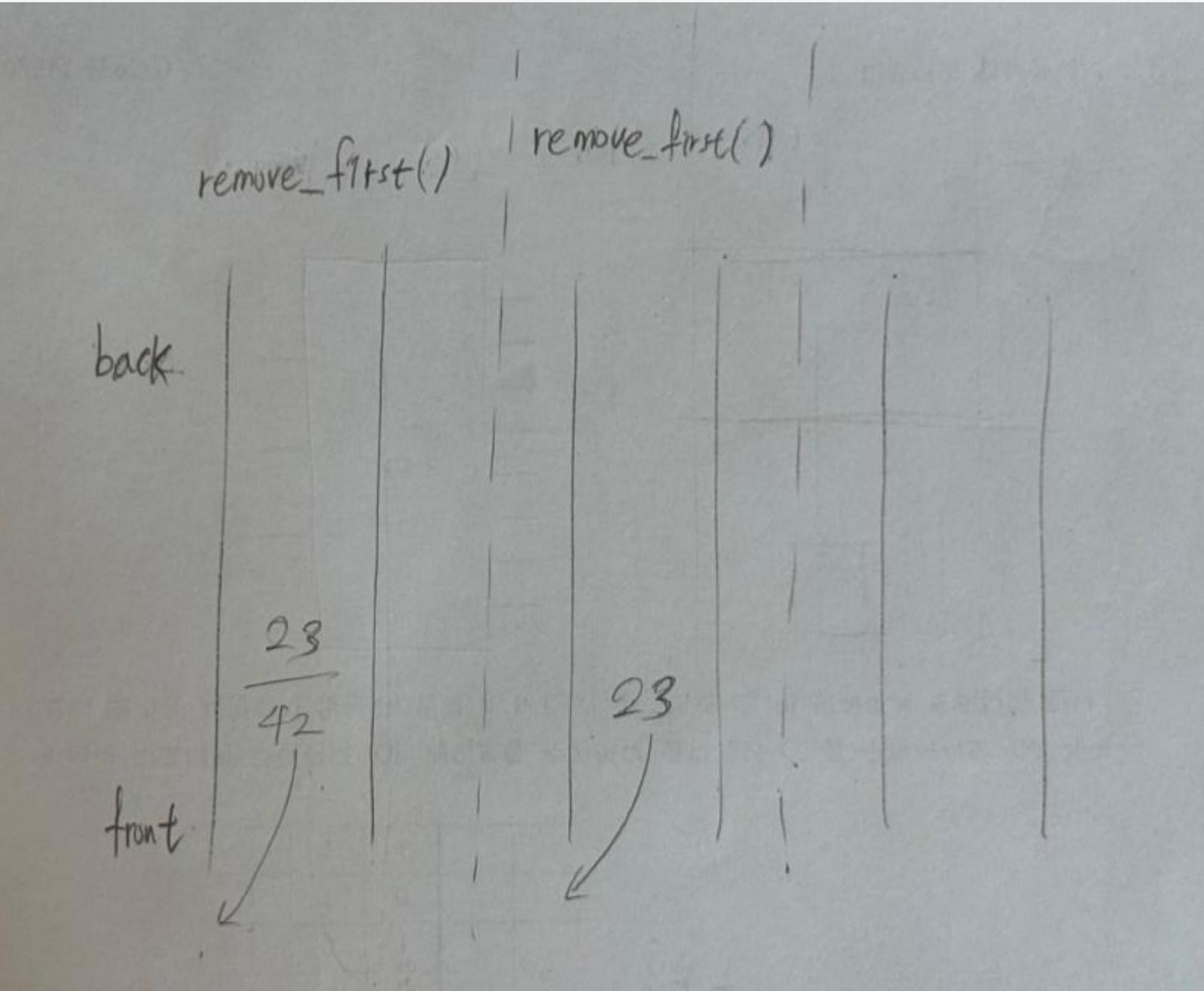
# 덱

1. add\_first 연산 : 앞에 자료를 넣는다.



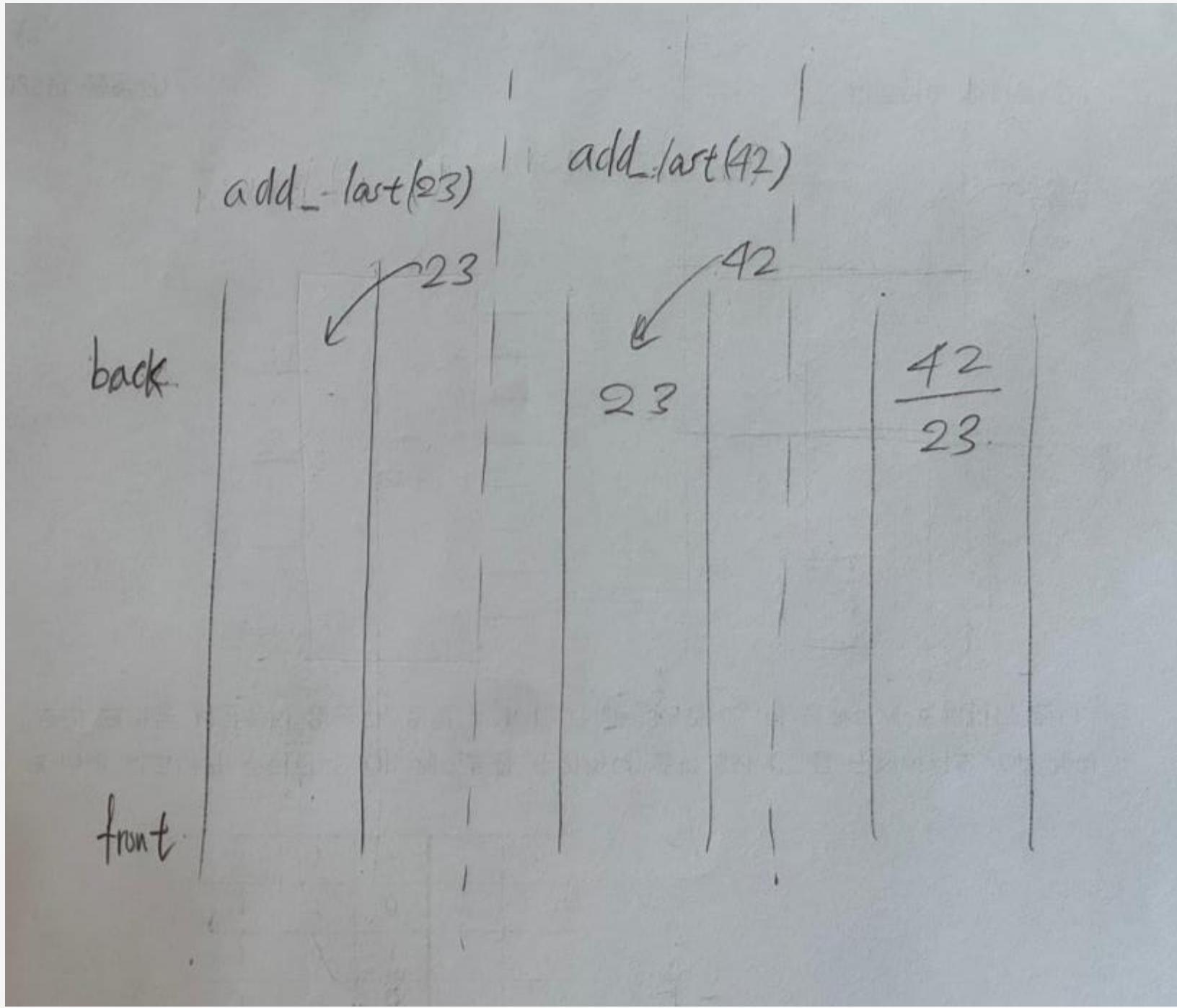
# 덱

2. remove\_first 연산: 앞의 자료를 뺀다.

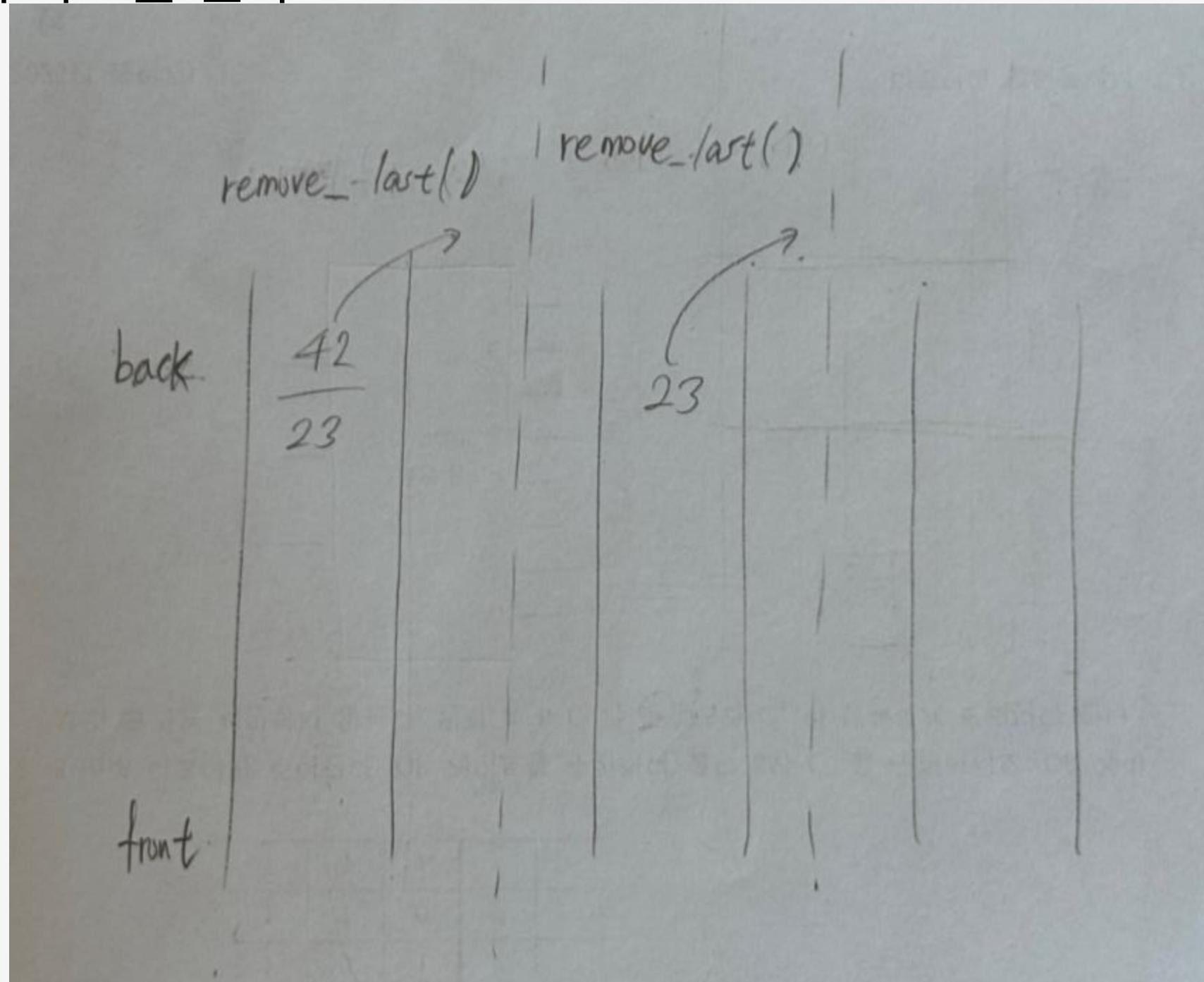


# 덱

3. add\_last 연산: 뒤에 자료를 넣는다.



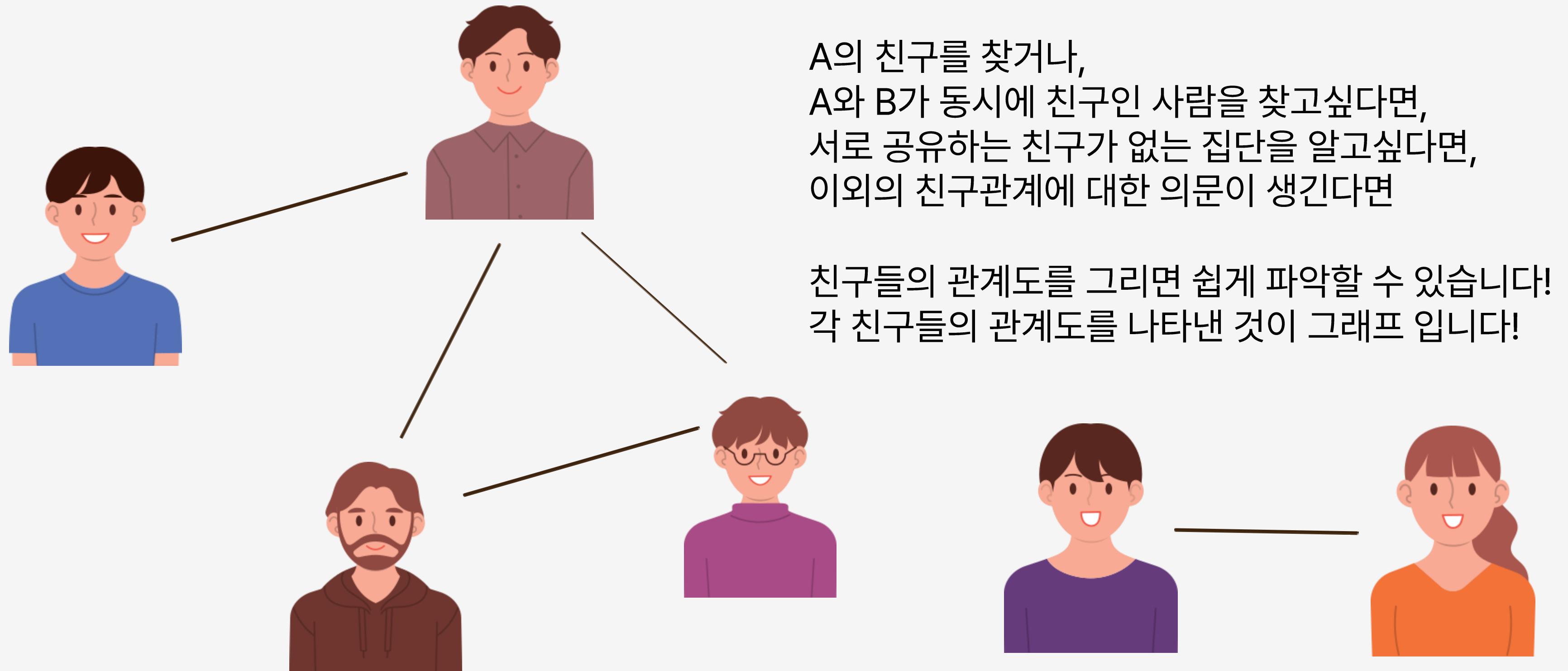
#### 4. remove\_last 연산: 뒤의 자료를 뺀다.





# 그래프

# 그래프란?



# 그래프란?

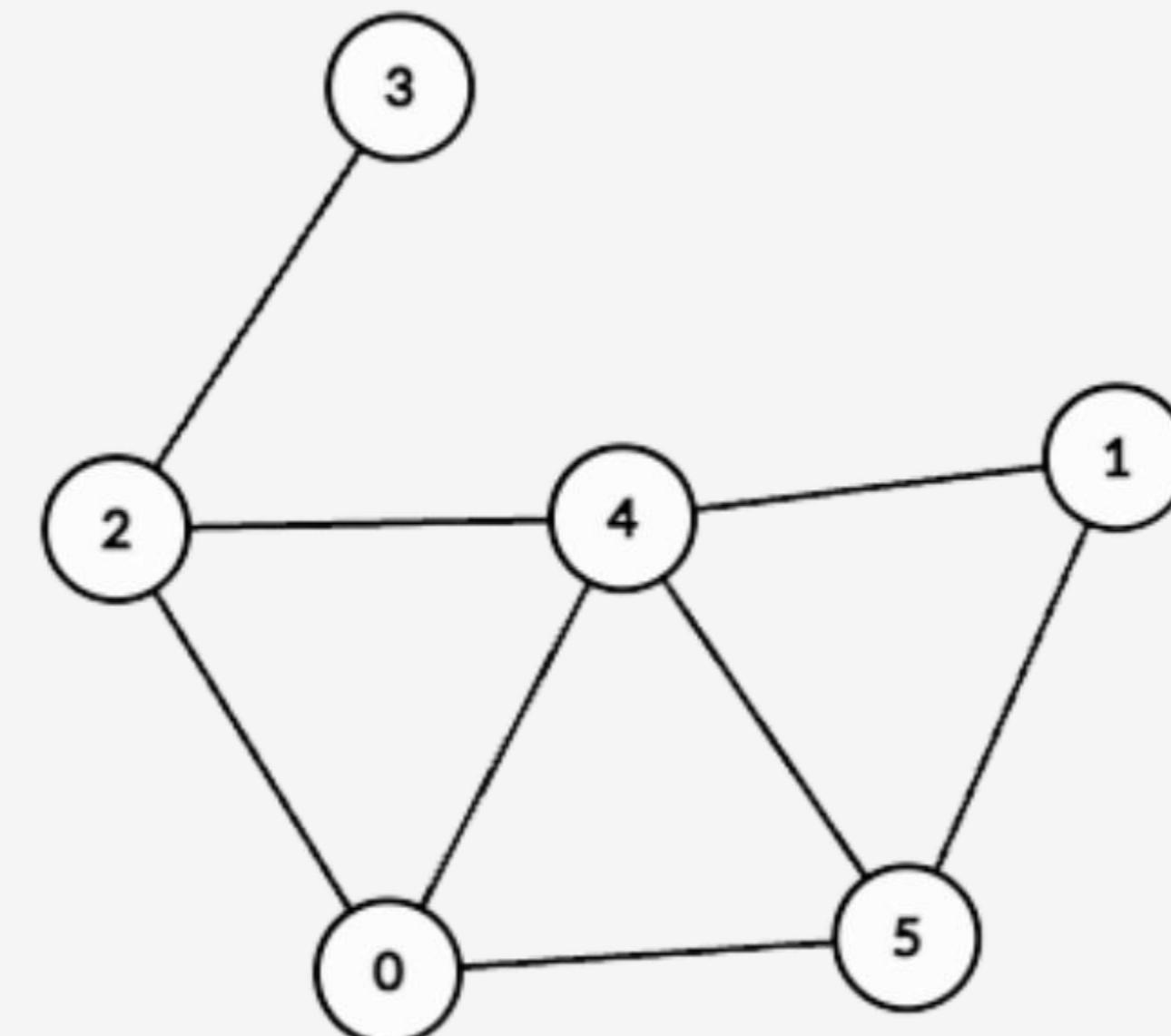
그래프(Graph)란 정점(Vertex)과 간선(Edge)의 집합으로 이루어진 자료구조로, 자료의 연결 관계를 표현하기 위한 수학적 구조입니다.

정점 (Vertex): 그래프에서 데이터를 담는 기본 단위

간선 (Edge): 두 정점 간의 연결선



친구 B



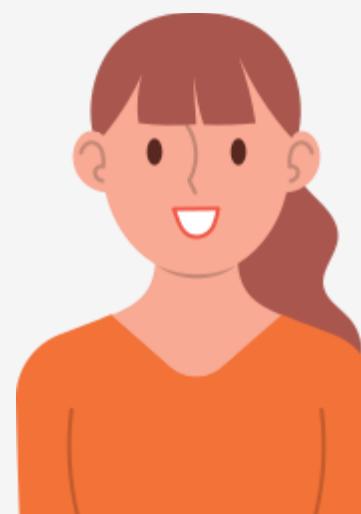
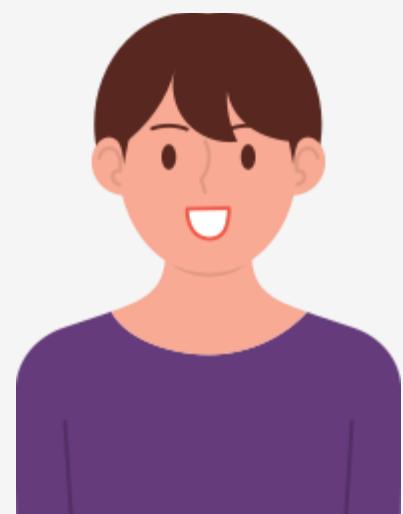
# 그래프의 종류

방향 그래프 (Directed Graph): 간선에 방향이 존재

무방향 그래프 (Undirected Graph): 간선에 방향이 없다!

가중치 그래프 (Weighted Graph): 간선마다 비용이 존재

등등.. 이 존재합니다!



친구 A

친구 B

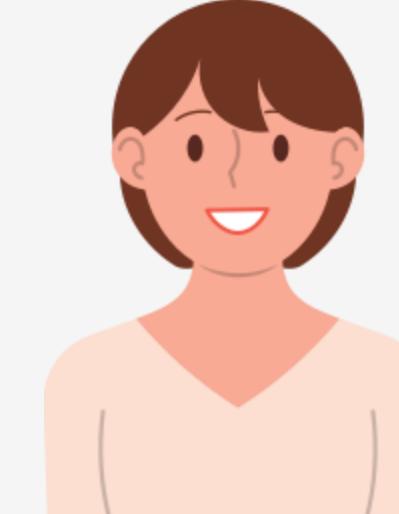


친구 A

친구사이



친구 B



친구 A

친구비 100만원



친구 B

# 그래프를 저장하는 방법

36

## 인접행렬 (Adjacency Matrix)

정점 간의 연결 관계를 2차원 배열로 저장합니다!

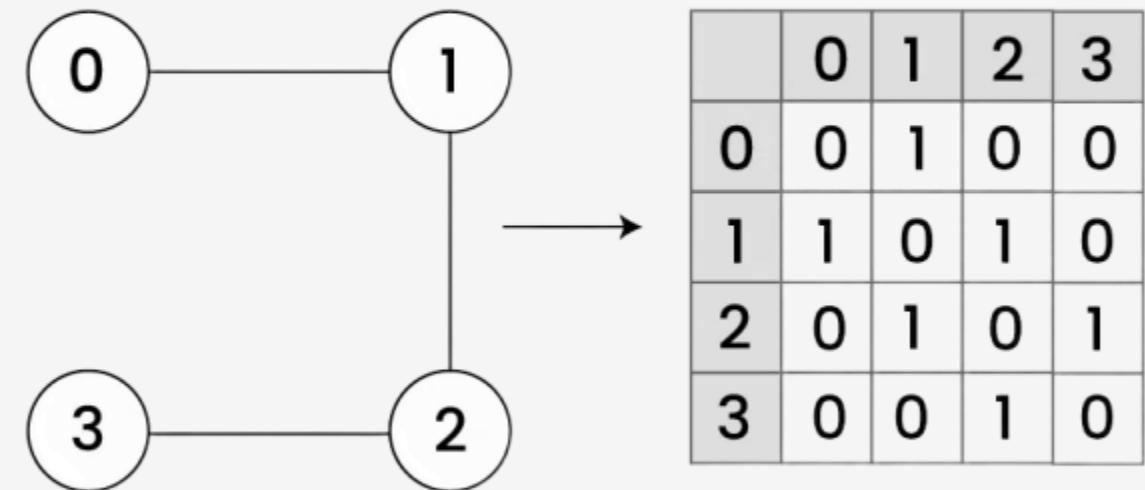
ex) A와 B가 친구면 A,B에 1을 저장 아니면 0을 저장.

간선의 존재 여부를 한번에 알 수 있습니다!

사람의 수가 N명일때, 관계도를 그래프로 표시하려면  $N^2$  의 공간이 필요합니다!

==> 이는 일반적으로 큰 수치

Adjacency Matrix for  
Undirected and Unweighted graph



Adjacency Matrix A[]

<https://www.geeksforgeeks.org/adjacency-matrix/>

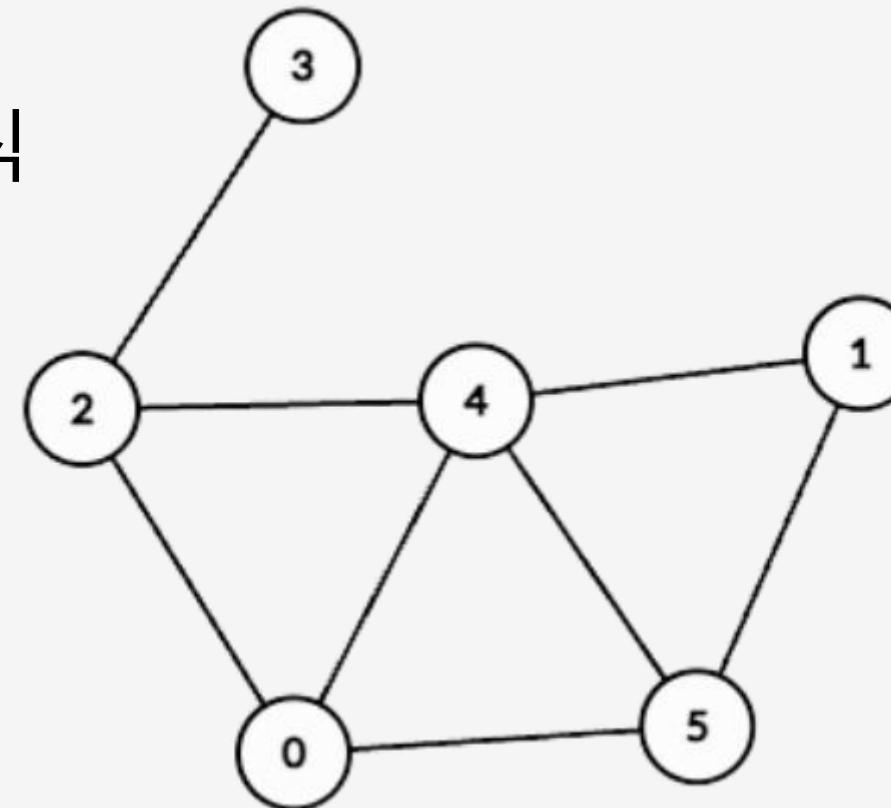


# 그래프를 저장하는 방법

## 인접리스트 (Adjacency List)

각 정점에 연결된 정점 목록을 리스트로 저장하는 방식  
ex) A의 친구는 누구누구.. B의 친구는 누구누구..

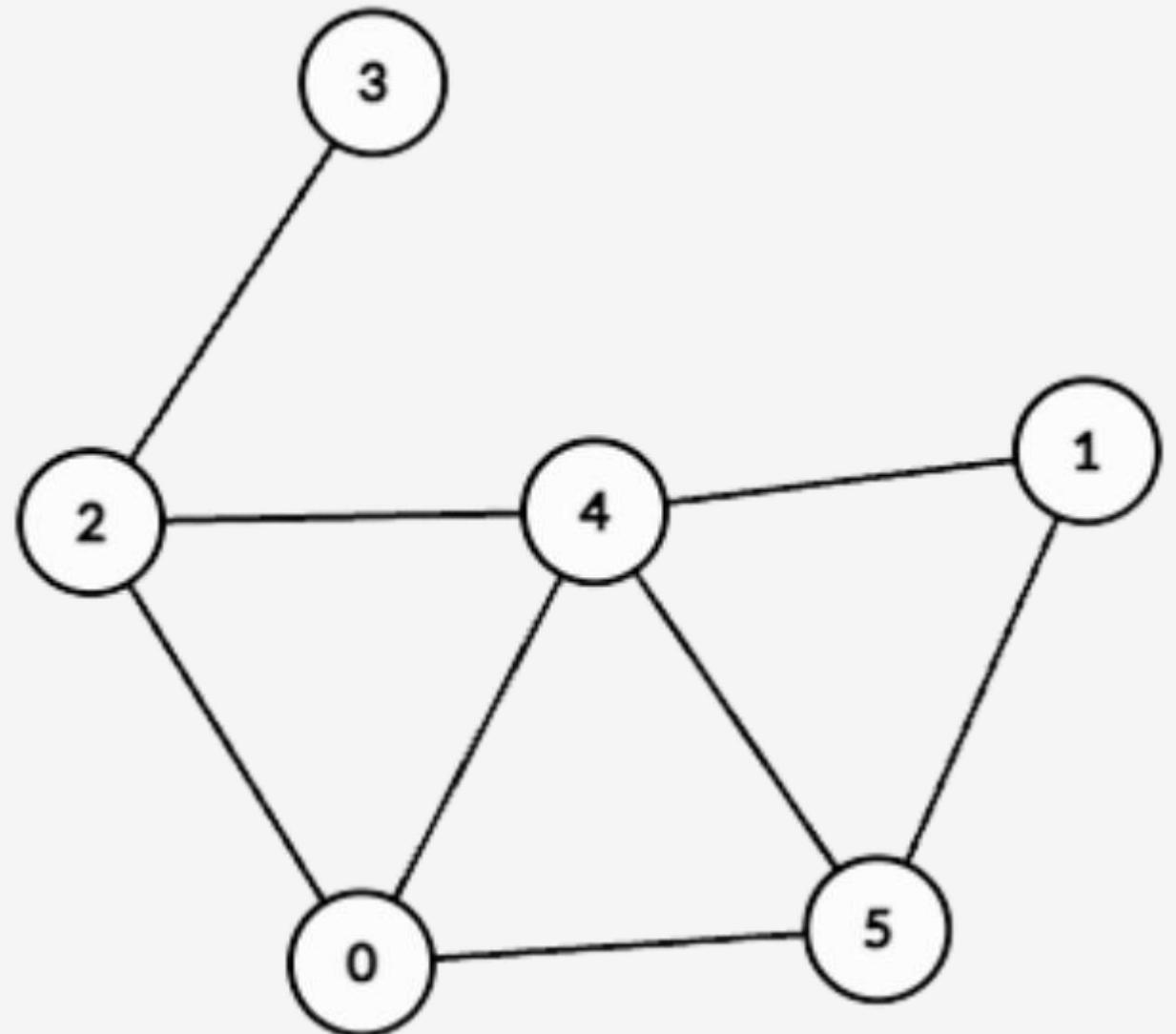
간선 존재 확인 속도 = 정점에 연결된 간선 수  
ex) A랑 B가 친구야? A의 친구목록에서 찾아보자



간선 존재 확인속도가 느린대신, 저장공간을 적게 차지한다는 장점이 있다!

$\text{adj}[0] = [2, 4, 5]$   
 $\text{adj}[1] = [4, 5]$   
 $\text{adj}[2] = [0, 3, 4]$   
 $\text{adj}[3] = [2]$   
 $\text{adj}[4] = [0, 1, 2, 5]$   
 $\text{adj}[5] = [0, 1, 4]$

# 그래프를 저장하는 방법 - PYTHON



## 인접행렬

```
adj = [[0, 0, 1, 0, 1, 1],  
       [0, 0, 0, 0, 1, 1],  
       [1, 0, 0, 1, 1, 0],  
       [0, 0, 1, 0, 0, 0],  
       [1, 1, 1, 0, 0, 1],  
       [1, 1, 0, 0, 1, 0]]
```

## 인접리스트

```
adj = [0, 0, 0, 0, 0, 0]  
adj[0] = [2, 4, 5]  
adj[1] = [4, 5]  
adj[2] = [0, 3, 4]  
adj[3] = [2]  
adj[4] = [0, 1, 2, 5]  
adj[5] = [0, 1, 4]
```

# 그래프를 저장하는 방법 - JAVA



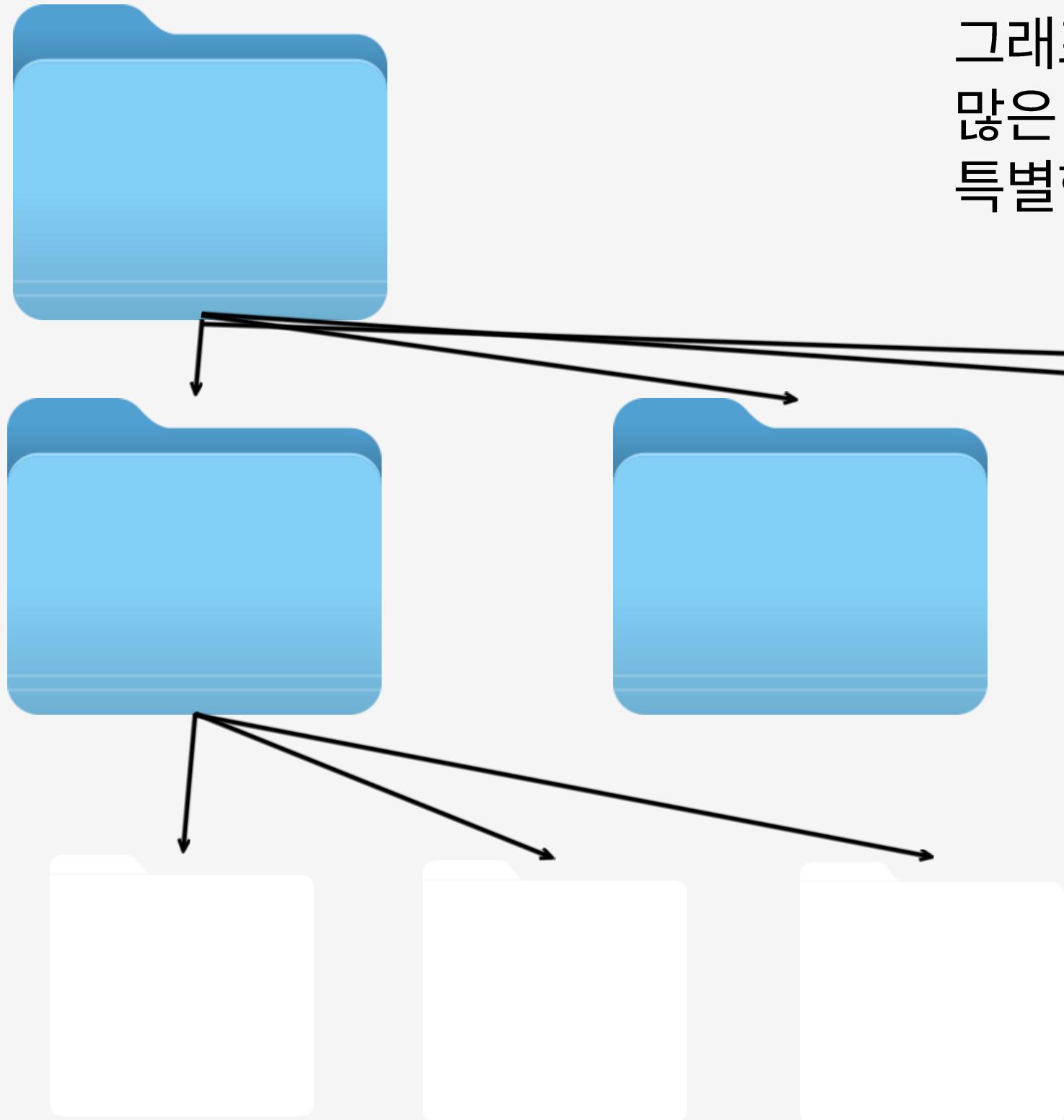
## 인접행렬

```
public class GraphMatrixExample {  
    public static void main(String[] args) {  
        int[][] adjMatrix = {  
            {0, 0, 1, 0, 1, 1}, // 0번 노드  
            {0, 0, 0, 0, 1, 1}, // 1번 노드  
            {1, 0, 0, 1, 1, 0}, // 2번 노드  
            {0, 0, 1, 0, 0, 0}, // 3번 노드  
            {1, 1, 1, 0, 0, 1}, // 4번 노드  
            {1, 1, 0, 0, 1, 0} // 5번 노드  
        };  
    }  
}
```

## 인접리스트

```
import java.util.*;  
public class GraphListExample {  
    public static void main(String[] args) {  
        List<Integer>[] adj = new List[6];  
        adj[0] = Arrays.asList(2, 4, 5);  
        adj[1] = Arrays.asList(4, 5);  
        adj[2] = Arrays.asList(0, 3, 4);  
        adj[3] = Arrays.asList(2);  
        adj[4] = Arrays.asList(0, 1, 2, 5);  
        adj[5] = Arrays.asList(0, 1, 4);  
    }  
}
```

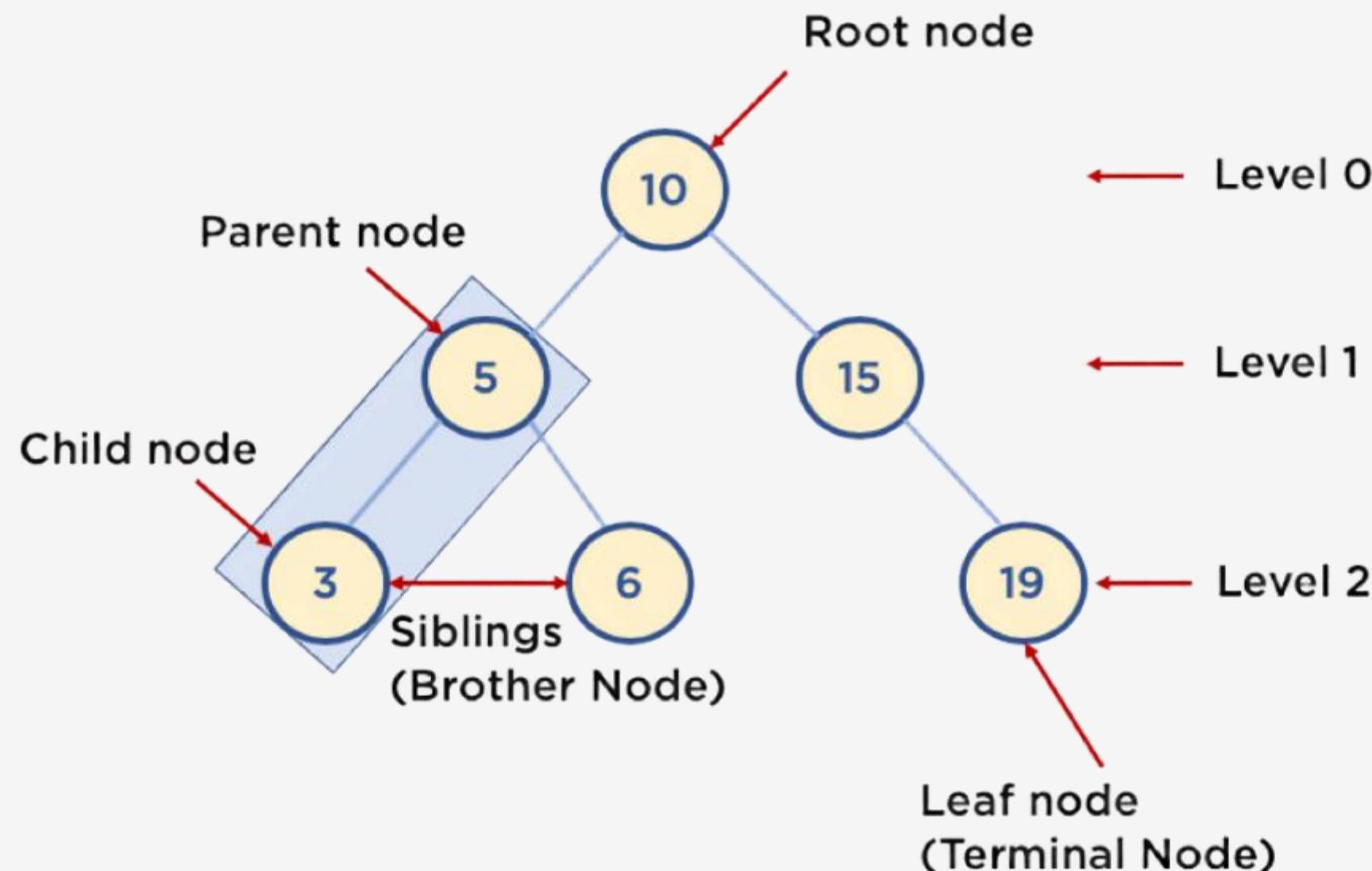
# 트리란?



그래프는 관계도를 표현하는 방법입니다!  
많은 관계중, 계층적 관계가 존재하는 관계를 표현하는 방법은  
특별한 그래프 구조중 하나인, 트리 구조를 사용하여 표현합니다!

# 트리란?

트리는 계층적(hierarchical) 구조를 가진 비순환 그래프(사이클이 없는 그래프)입니다!



루트(root) : 트리의 맨 위 노드 (시작점)

부모(parent) : 다른 노드를 가리키는 노드

자식(child) : 부모로부터 연결된 노드

리프(leaf) : 자식이 없는 노드 (끝 노드)

형제(sibling) : 같은 부모를 가진 노드

깊이(depth) : 루트에서 특정 노드까지의 간선 수

높이(height) : 트리의 최대 깊이

서브트리(subtree) : 트리의 일부 (자식 포함 작은 트리)

# 트리의 종류?

트리의 종류에는

일반 트리 : 자식 노드를 몇 개든 가질 수 있음.

이진 트리 : 자식 노드를 최대 2개만 가질 수 있음.

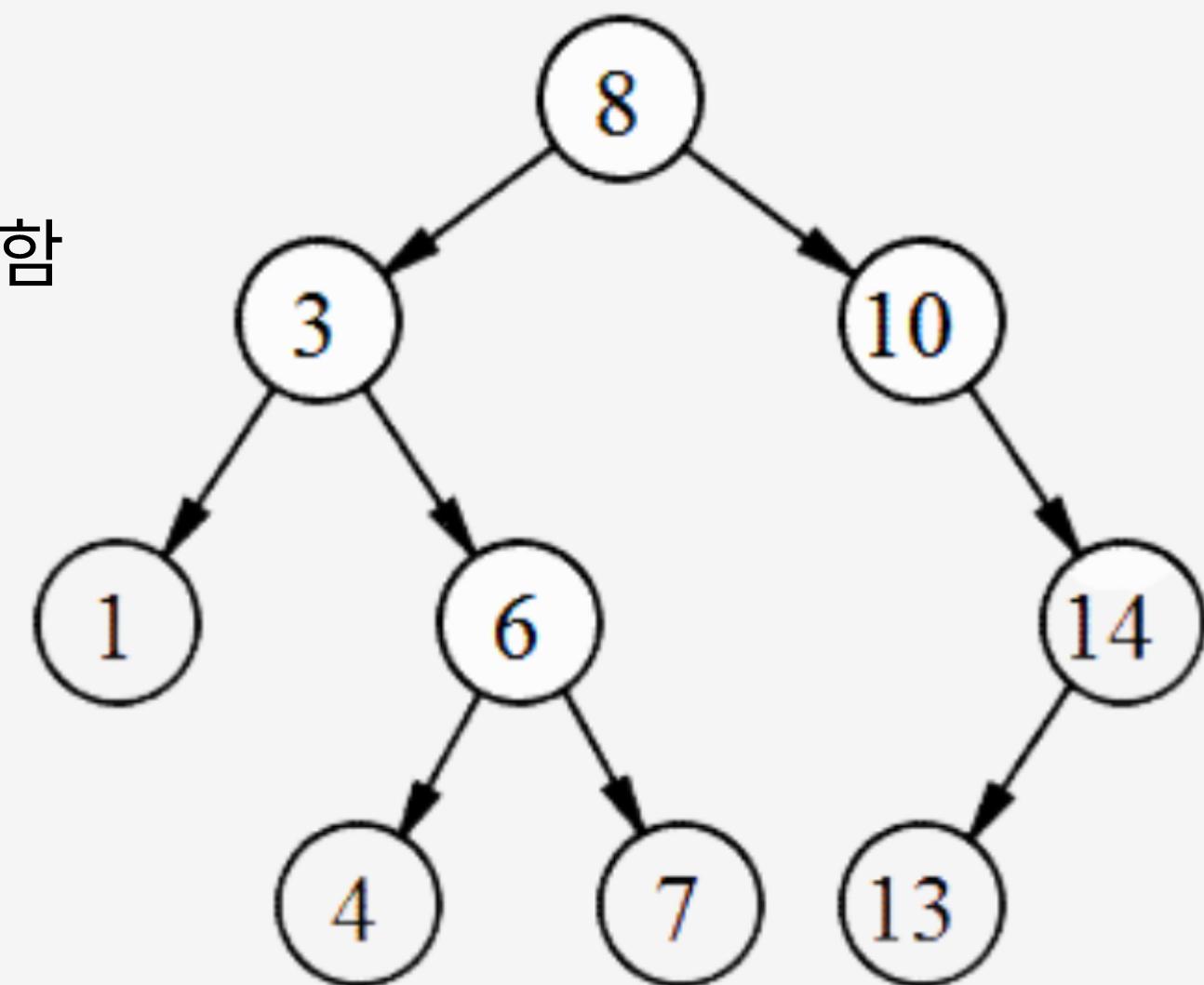
이진 탐색 트리 (BST) : 왼쪽에는 작은 값, 오른쪽에는 큰 값을 저장함

이외에도

이진탐색트리, 완전이진트리, 포화이진트리, 힙, 트라이, AVL..

정말 많은 트리가 있습니다!

이중에서, 일반트리와 이진트리의 구현에 대해 알아봅시다!



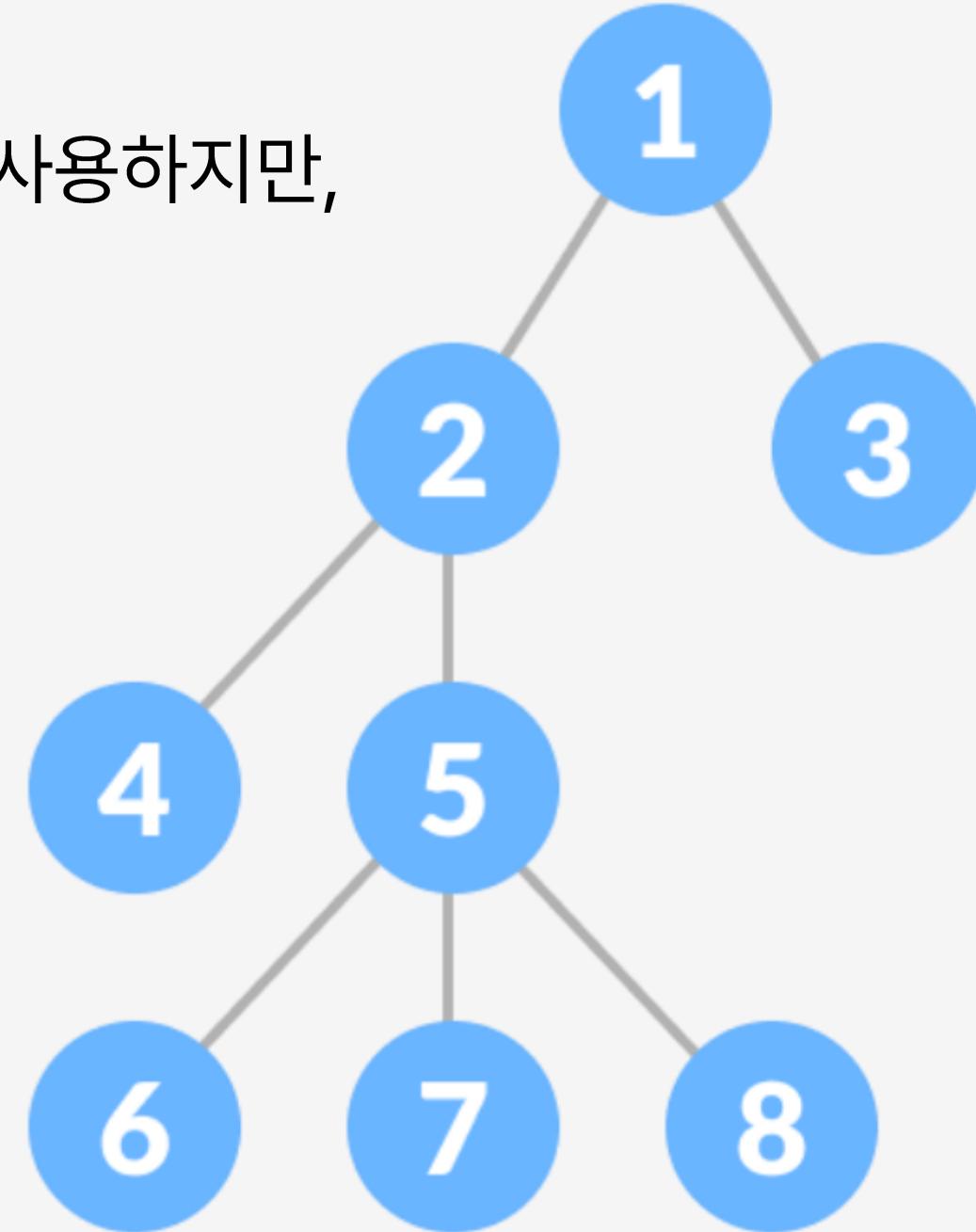
# 트리를 저장하는 방법 - 일반트리

트리를 구현하는 법에는 여러가지가 있습니다!

일반적으로 복잡한 트리구조를 만들기 위해선, 새로운 클래스를 정의해 사용하지만,  
여기선 인접리스트를 통한 트리의 구현에 대해 알아봅시다!

부모노드 = [자식노드1, 자식노드2 ..]

adj[1] = [2, 3]  
adj[2] = [4, 5]  
adj[4] = []  
adj[5] = [6, 7, 8]  
adj[6] = []  
adj[7] = []  
adj[8] = []  
adj[3] = []



# 트리를 저장하는 방법 - 이진트리



이진트리는 자식노드의 개수가 최대 2개로 고정된 특별한 트리입니다!

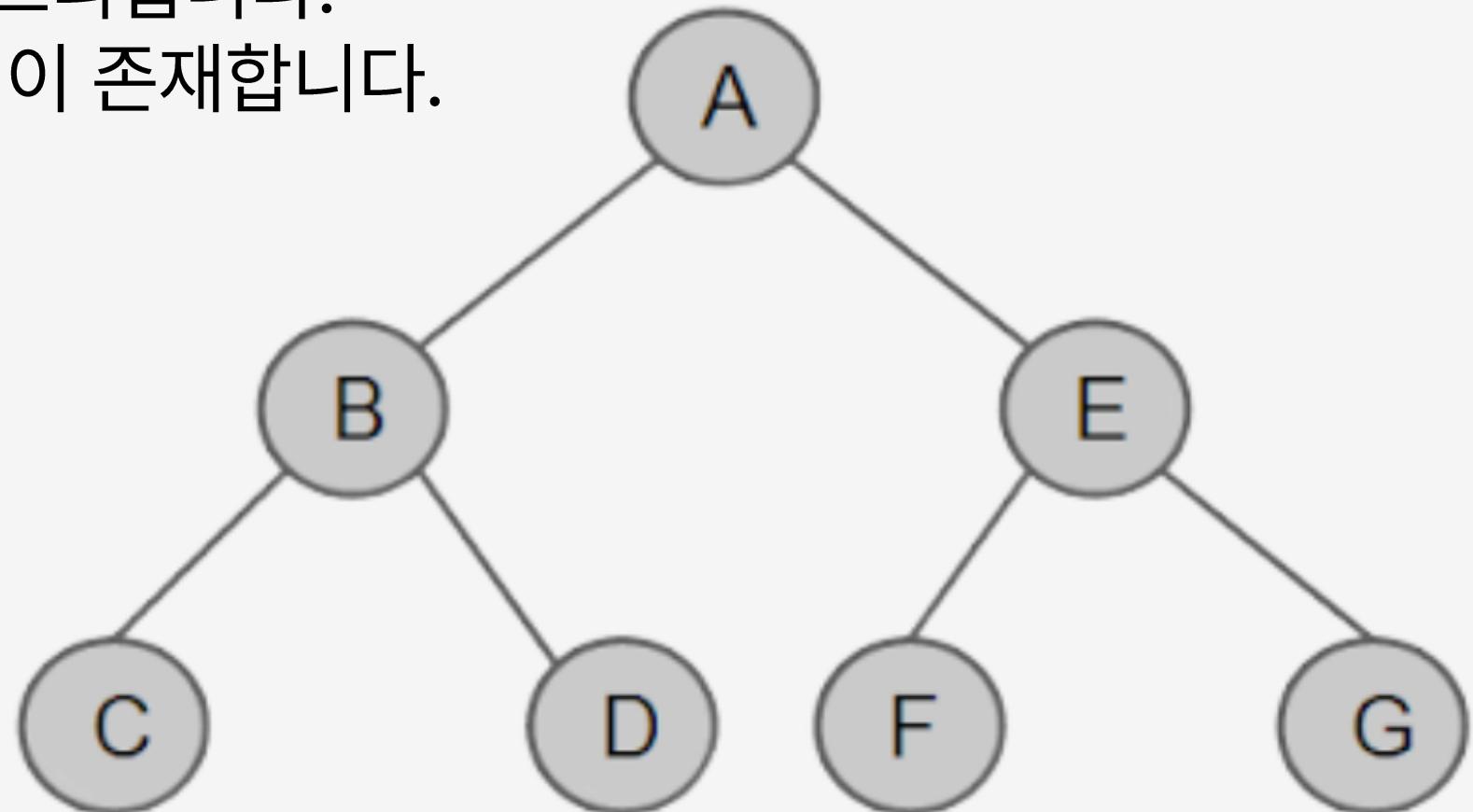
이러한 특성 덕분에, 이진트리를 효과적으로 저장하는 방법이 존재합니다.

배열의 인덱스로 부모 - 자식 관계를 표현하는 방법을 통해  
트리를 1차원 배열에 저장할 수 있습니다

부모노드의 인덱스가  $i$  일때,

왼쪽 자식노드의 인덱스 :  $2*i + 1$

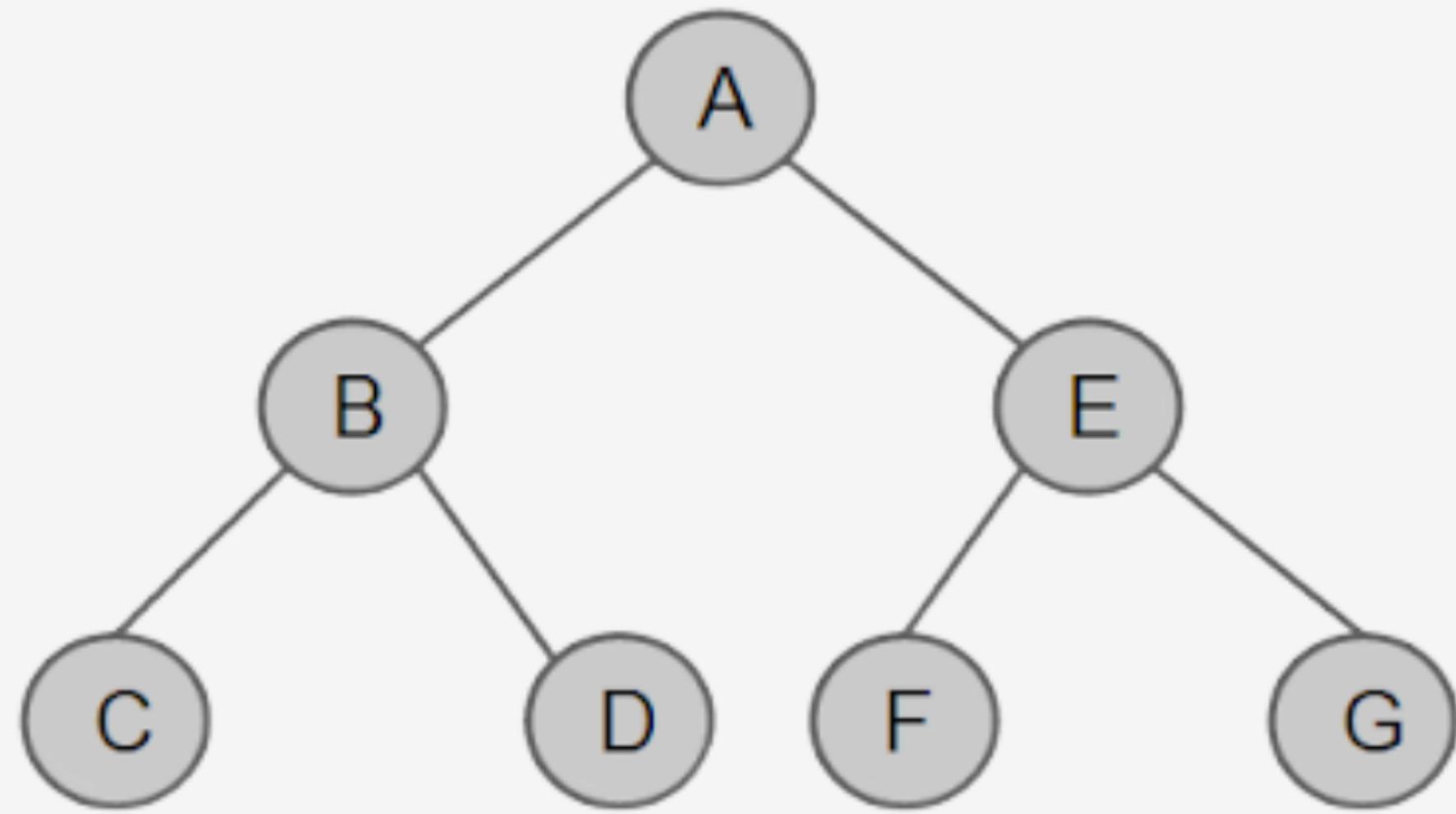
오른쪽 자식노드의 인덱스 :  $2*i + 2$



1      2      3      4      5      6      7

	A	B	E	C	D	F	G
--	---	---	---	---	---	---	---

# 트리를 저장하는 방법 - PYTHON



인접리스트 (+ 딕셔너리)

```
tree = dict()
tree['A'] = ['B', 'E']
tree['B'] = ['C', 'D']
tree['C'] = []
tree['D'] = []
tree['E'] = ['F', 'G']
tree['F'] = []
tree['G'] = []
```

이진 트리

```
tree = [0, 'A', 'B', 'E', 'C', 'D', 'F', 'G']
```

# 트리를 저장하는 방법 - JAVA

## 이진 트리

```
public class BinaryTreeNode {  
    public static void main(String[] args) {  
        String[] tree = String[] tree  
        = {"", "A", "B", "E", "C", "D", "F", "G"};  
    }  
}
```

## 인접리스트 (+ 해시 맵)

```
import java.util.*;  
public class TreeMapExample {  
    public static void main(String[] args) {  
        Map<String, List<String>> tree = new HashMap<>();  
        tree.put("A", Arrays.asList("B", "E"));  
        tree.put("B", Arrays.asList("C", "D"));  
        tree.put("C", new ArrayList<>());  
        tree.put("D", new ArrayList<>());  
        tree.put("E", Arrays.asList("F", "G"));  
        tree.put("F", new ArrayList<>());  
        tree.put("G", new ArrayList<>());  
    }  
}
```



# More.. more..

<https://www.acmicpc.net/problem/10845>  
<https://www.acmicpc.net/problem/24511>  
<https://www.acmicpc.net/problem/1966>  
<https://www.acmicpc.net/problem/10866>  
<https://www.acmicpc.net/problem/14244>  
<https://www.acmicpc.net/problem/27966>  
<https://www.acmicpc.net/problem/9372>

“실패를 기반으로 쌓아간다. 그것을 디딤돌로 삼는다. 과거를 닫되, 실수를 잊으려 하지 말고, 거기에 머무르지도 마라.”

— Johnny Cash

1 영역 구하기	성공	다국어
2 촌수계산	성공	
4 이분 그래프	성공	
3 내리막 길	성공	
4 빙산	성공	
4 트리의 지름	성공	
1 그림	성공	
2 트리의 지름	성공	
5 트리	성공	
3 팀 프로젝트	성공	
5 ABCDE	성공	다국어
1 효율적인 해킹	성공	
3 욕심쟁이 판다	다국어	
2 알고리즘 수업 - 깊이 우선 탐색 1	성공	
1 음식물 피하기	성공	
3 게리맨더링	다국어	
2 현내기는 친구가 필요해	성공	
키스	성공	
숫자고르기	성공	
빵집	성공	
2 알고리즘 수업 - 깊이 우선 탐색 2	성공	다국어
물통	성공	
트리와 큐리	다국어	





# End.

Next Lecture. 25.05.14 / BFS & DFS

---