

청년 AI 아카데미 19기 알고리즘 실습

DomJudge 튜토리얼 및 자료구조 실습

Orientation

TA

- A반: 강병욱 (kbu417@postech.ac.kr)
- B반: 최종민
- C반: 정묵권

평가

- 과제 90% (3문제 예정, 프로그램 실행 속도 순으로 점수차등제 적용)
- 필기 시험 10% (총점 10점, 3문제 출제 예정, 9/1 17:30~18:00)

알고리즘 실습의 목표

알고리즘: Input → HOW?? → Output

- 알고리즘은 왜 중요한가요? (feat. 수학은 왜 중요한가요?)
- 컴퓨터 학문의 기초이자, 컴퓨터처럼 접근하는 사고능력과 논리를 기를 수 있습니다!

알고리즘 실습의 중점적인 목표

- 계산 문제를 해결할 때 주로 사용되는 기법(알고리즘)의 **개념**을 배웁니다.
Ex> 분할 정복, 욕심쟁이 기법, 동적 계획법 등...
- 이론을 바탕으로 **대표적인 알고리즘 및 문제** 등을 실습합니다.
Ex> Knapsack, BFS, DFS 등...
- 강의 내용을 **응용**하여 새로운 문제를 스스로 해결하는 **연습**을 합니다.
- 설계한 알고리즘의 분석 및 **코드로 구현**하는 연습을 합니다.
→ 바로 코딩보다는...

효율적인 알고리즘 설계의 중요성

	$O(n^3)$	$O(n^2)$	$O(n \lg n)$ (approx.)
100	1,000,000x	10,000x	1,650x
1000	100,000,000x	1,000,000x	16,500x
10000	1e+12x	100,000,000x	165,000x
100000	1e+15x	10,000,000,000x	1,650,000x

In 1 second?

Loose

Very tight

Impossible

알고리즘 실습의 목표

영우 나무위키

1. 개요

[편집]

코딩 테스트는 소프트웨어 관련 기업에서 지원자의 기술적인 역량을 시험하기 위해 실시하는 [알고리즘](#)이나 직무 연관성이 있는 기술에 대한 구현 테스트를 의미한다.

2. 상세

[편집]

'코테'라고 줄여서 부르기도 한다.

시험 응시자들은 해당 문제를 해결하는 컴퓨터 프로그램을 작성하고, 소스 코드를 온라인 판정(Online Judge) 프로그램에게 제출해야 한다. 온라인 저지 프로그램은 미리 준비되어 있는 테스트 케이스들을 자동화된 방식으로 채점한다. 카카오, NHN, 삼성전자 등의 주요 IT 관련 기업들이 [프로그래머](#)를 채용하기 위한 시험 방식의 하나로 활용하고 있다.

[적성검사](#)가 여러 기업들 사이에서 축소되거나 폐지되는 가운데, 코딩과 관련된 IT계열에서는 굳이 대기업이 아니더라도 입사 전 인적성검사 대신 코딩 테스트를 치르는 기업이 늘고 있는 추세이다.



상시 SW 역량테스트 구성

평가기준 : TestCase 전체 Pass, 실행속도, 코드리뷰 등

구분	검정시간	지원언어	사용가능한 라이브러리	샘플문제	추천 연습문제
A형	3시간	C/C++/Java/Python	제한 없음	풀어보기	D2~4
B형	4시간	C/C++/Java	라이브러리 사용 불가 (단, C언어의 경우 동적할당을 위한 <malloc> 가능)	풀어보기	D4~6
C형	4시간	C/C++	라이브러리 사용 불가 (단, C언어의 경우 동적할당을 위한 <malloc> 가능)	풀어보기	D5~7

Today

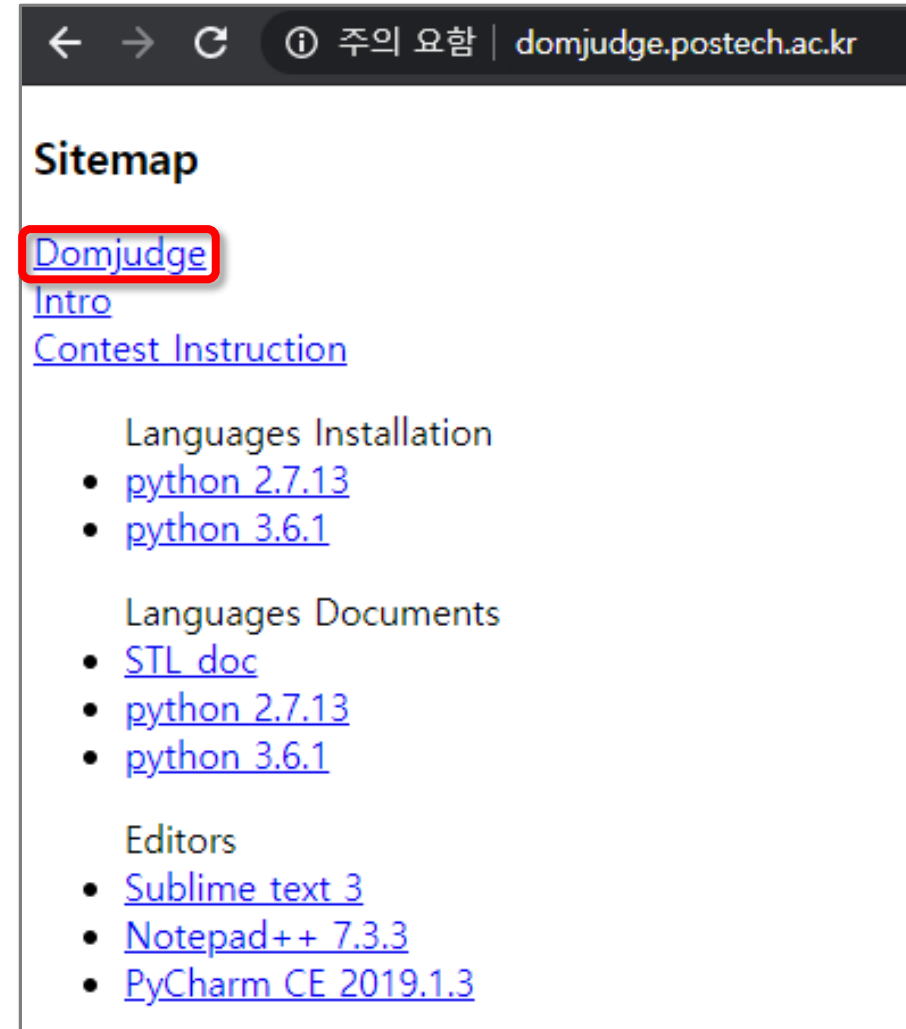
- 수업 진행 방식 소개
 - Domjudge 사용법
- 자료구조
 - 스택
 - 큐
 - 우선순위 큐

Introduction to Domjudge

Domjudge

- <https://domjudge.postech.ac.kr>
- 코딩 경시대회 플랫폼
- 코드 제출 및 채점 기능

“Domjudge” 버튼을 클릭하세요.



Introduction to Domjudge

Domjudge의 "Register now"를 클릭하여 아이디를 등록하세요.

DOMjudge | Scoreboard | Problemset | [Login](#) | 698d 8:15:58

Demo contest | starts: 20:00 - ends: 01:00

Filter

RANK	TEAM	SCORE	BOOLFIND	FLTCMP	HELLO
1	♥ Example teamname	0 0			
SUMMARY		0	0 0 0 0 0 0 0 n/a	0 0 0 0 0 0 0 n/a	0 0 0 0 0 0 0 n/a

Cell colours

- Solved first
- Solved
- Tried, incorrect
- Tried, pending
- Untried

DOMjudge

Please sign in

[Sign in](#)

[Don't have an account? Register now.](#)

Introduction to Domjudge

가입 시 다음과 같이 기입합니다.

- Username: 로그인할 아이디
- Full name: 생략
- Email: 생략
- Team name: 조+본인 이름 (영어로 작성)
 - Ex) A1Jongmin
- Password: 패스워드
- Repeat Password: 패스워드 확인

Register Account

Enter the following information to register your account with DOMjudge.

Domjudge Home

① DOMjudge Home Problemset Scoreboard Submit Logout demo 698d 8:09:46

Change Contest
test

RANK	TEAM	SCORE	BOOLFIND	FLTCMP	HELLO
1	rucatia	0	0		

Submissions

No submissions

Clarifications

No clarifications.

Clarification Requests

No clarification request.

request clarification

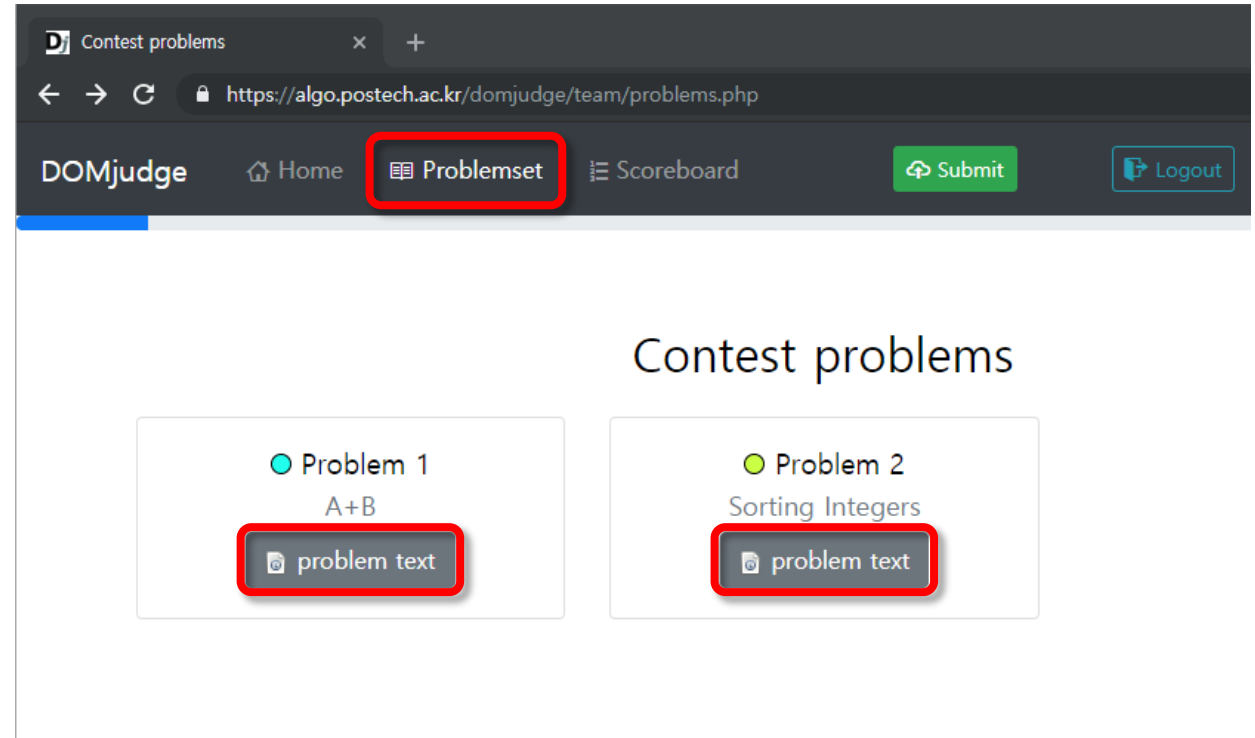
로그인 후의 홈 화면입니다.

① 이 버튼을 누르면 홈 화면으로 돌아오게 됩니다.

② 이 버튼을 통해 contest를 선택하시면 해당하는 날짜에 진행되는 실습에 참가하실 수 있습니다.

Problem Text

- 상단의 **Problemset** 버튼을 누르면 다음과 같이 문제 설명을 볼 수 있는 페이지로 넘어옵니다.
- “**problem text**” 버튼을 눌러 각 문제에 대한 설명을 확인하실 수 있습니다.



How to Submit

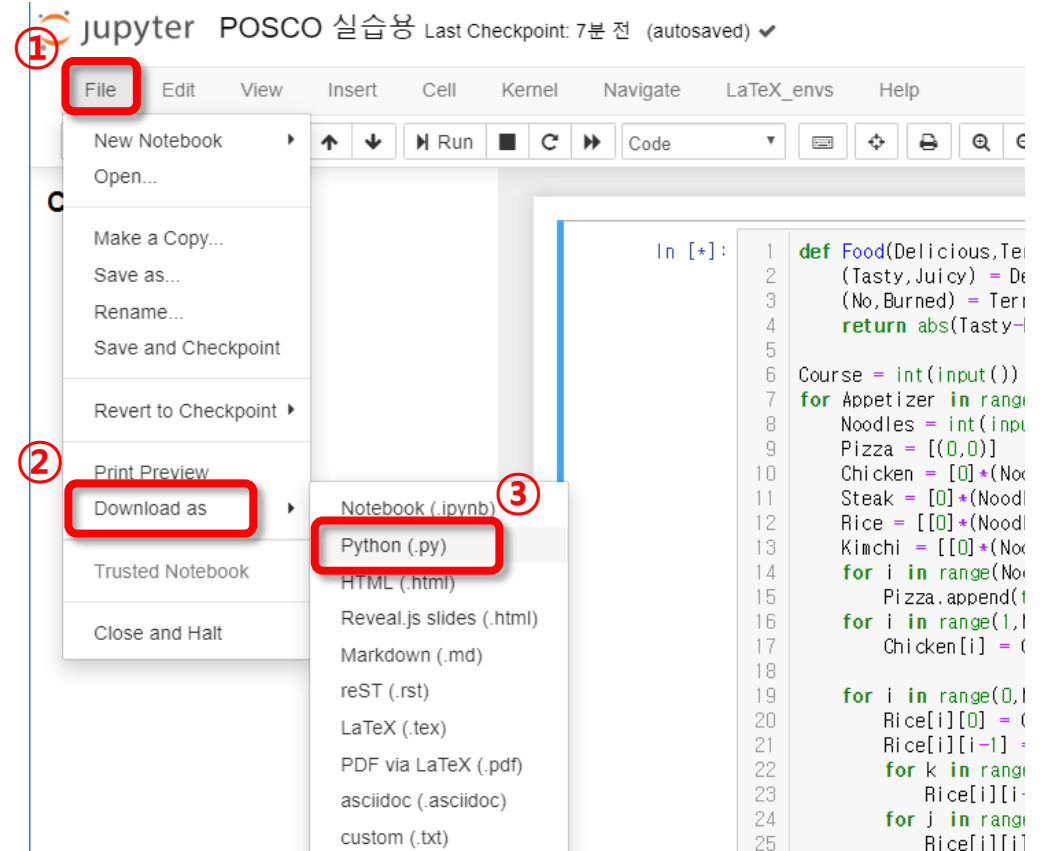
Jupyter Notebook을 쓰신다면, 오른쪽과 같은 방법으로 개인 컴퓨터에 py 파일을 저장 가능합니다.

- ① 열려있는 ipynb 파일에서 File 선택
- ② Download as 선택
- ③ Python (.py) 선택

자동으로 다운로드가 되며, 컴퓨터의 다운로드 폴더에 있습니다. (이 부분은 개인이 인터넷 옵션을 어떻게 설정하느냐에 따라 다릅니다.)

※주의! Cell이 여러 개 있다면 모두 묶어서 하나의 파일로 다운로드가 됩니다. 제출 시 다른 Cell의 주석처리 혹은 파일을 분리해서 하는 것을 추천합니다.

+ 제출 전 코드를 확인해보세요!



How to Submit

상단의 **Submit** 버튼을 통해서 코드를 제출하실 수 있습니다.

- ① 소스 코드를 선택하여 붙여 넣고(혹은 Browse)
- ② 제출할 문제를 선택한 후
- ③ 제출 언어를 Python3로 설정합니다.
(제출한 소스코드의 확장자가 .py인 경우 자동으로 선택됩니다.)

DOMjudge Home Problemset Scoreboard **Submit** Logout

Submit

① Source files
No file selected Browse

② Problem
Select a problem

③ Language
Select a language

Cancel Submit

※ 제출하는 소스 코드 파일 이름에 공백이 있으면 안 됩니다!

※ 문제가 많으니 제출은 한꺼번에 하는 것보다 그때 그때 하는 것을 추천합니다!

Submission Results

코드를 제출하게 되면 다음과 같이 제출 결과들이 표시 됩니다.

RANK	TEAM	SCORE	1	2
1	sloth	1 1467	1467 12 tries	1 try

Submissions			
time	problem	lang	result
15:06	1	py3	TIMELIMIT
15:04	1	py3	CORRECT
15:04	1	py3	RUN-ERROR
15:03	1	py3	RUN-ERROR
15:02	2	py3	RUN-ERROR
15:00	1	py3	RUN-ERROR
14:59	1	CPP	WRONG-ANSWER
14:58	1	CPP	COMPILER-ERROR

PENDING: 서버에서 코드를 채점 중입니다. 기다린 후에 새로고침(F5)을 누르시기 바랍니다.

COMPILER-ERROR: 파일을 컴파일하는 과정에서 오류가 발생했습니다. 코드에 문법적 오류가 있는지 확인해보세요.

RUN-ERROR: 프로그램이 작동하는 중에 오류가 발생했습니다. 입력 형식에 맞게 프로그램이 작성되었는지, list index와 메모리 관리가 적절한지 확인하세요. 이유를 모르겠으면 조교에게 문의하시길 바랍니다.

NO-OUTPUT: 프로그램이 아무것도 출력하지 않았습니다.

WRONG-ANSWER: 프로그램의 출력이 정답과 다릅니다. 알고리즘을 다시 생각해 보세요.

TIMELIMIT: 프로그램이 시간 제한으로 인해 종료되었습니다. 더 효율적인 알고리즘을 생각해 보시기 바랍니다.

CORRECT: 정답입니다!

Clarification

- **Clarification**은 Domjudge 서버를 통해 조교에게 질문을 할 수 있는 기능입니다.
- 실습시간 외에 조교에게 질문하고 싶은 것이 있다면 이 기능을 활용해주세요.
- “**request clarification**” 버튼을 통해 clarification을 보낼 수 있습니다.

The screenshot shows the DOMjudge interface. At the top, there's a navigation bar with links: Home, Problemset, Scoreboard, Jury, Submit, and Logout. The current contest is 'judgetest' and the time remaining is 330d 5:36:11.

The main content area displays a scoreboard for the 'admin' team. The scoreboard has columns for RANK, TEAM, SCORE, and 11 problem numbers (01-11). The 'admin' team is ranked 2nd with a score of 7 (181434). The scores for each problem are: 01: 44 (1 try), 02: 9 tries, 03: 50 (1 try), 04: 2 tries, 05: 27716 (1 try), 06: 27754 (1 try), 07: 29950 (11 tries), 08: 47933 (2 tries), 09: 47987 (9 tries), 10: 47987 (9 tries), 11: 47987 (9 tries).

Below the scoreboard, there are two tabs: 'Submissions' and 'Clarifications'. The 'Submissions' tab is active, showing a list of submissions with columns: time, problem, lang, and result. The submissions are as follows:

time	problem	lang	result
Mar 31 23:38	10	py3	CORRECT
Mar 31 23:22	10	py3	CORRECT
Mar 31 23:18	10	py3	CORRECT
Mar 31 23:12	10	py3	RUN-ERROR
Mar 31 23:04	10	py3	RUN-ERROR
Mar 31 23:03	10	py3	RUN-ERROR
Mar 31 22:46	10	py3	WRONG-ANSWER
Mar 31 22:45	10	py3	WRONG-ANSWER

The 'Clarifications' tab is also visible, showing 'No clarifications.' and 'Clarification Requests' with 'No clarification requests.' Below the 'Clarification Requests' section, there is a button labeled 'request clarification' which is highlighted with a red box.

Clarification

① 질문할 문제를 선택합니다. 문제에 관한 질문이 아닌 경우, "**General issue**"를 선택해주세요.

② 질문 내용을 입력 후

③ "**Send**" 버튼을 누르면 조교에게 clarification이 전달됩니다.

Send clarification request

×

Recipient

Jury

①

Subject

General issue

②

Message

Cancel

Send

③

Clarification

답변이 오면 다음과 같이 홈 화면
오른쪽에서 확인하실 수 있습니다.

The screenshot shows the DOMjudge interface. At the top, there's a navigation bar with links for Home, Problemset, Scoreboard, and Jury. A 'Submit' button and a 'Logout' button are also present. The current contest is 'HynixDay01' and the time remaining is 28 days. Below the navigation bar is a scoreboard table with columns for Rank, Team, Score, and 12 problems. The 'admin' team is highlighted in pink. To the right of the scoreboard, there are two panels: 'Submissions' and 'Clarifications'. The 'Clarifications' panel is highlighted with a red border and contains a table with columns for time, from, to, subject, and text. A 'request clarification' button is located below the 'Clarifications' panel.

RANK	TEAM	SCORE	01	02	03	04	05	06	07	08	09	10	11	12
1	admin	15 225	8 1 try	8 1 try	9 1 try	9 1 try	9 1 try	35 3 tries	10 1 try	11 1 try	11 1 try	35 2 tries	14 2 tries	14 2 tries

Submissions				
time	problem	lang	result	
Apr 01 21:36	13	py3	CORRECT	
Apr 01 21:32	15	py3	CORRECT	
Apr 01 21:17	10	py3	CORRECT	
Apr 01 21:17	06	py3	CORRECT	
Apr 01 21:10	13	py3	CORRECT	
Apr 01 21:09	06	py3	RUN-ERROR	
Apr 01 20:56	12	py3	CORRECT	
Apr 01 20:56	11	py3	CORRECT	

Clarifications				
time	from	to	subject	text
Apr 02 10:06	Jury	You	General issue	돈 없어요.....

Clarification Requests				
time	from	to	subject	text
Apr 02 10:06	You	Jury	General issue	조교님 밥 사주실래요?

request clarification

Problem Text

문제 제목

문제에 대한 설명

프로그램의 입력 형식

프로그램의 출력 형식

입출력 예시

리스트의 합

문제 정의

정수로만 이루어진 리스트에서 각 원소들을 모두 더한 값을 출력하는 프로그램을 작성하세요.

입력 형식

- 입력의 첫 줄에 테스트 케이스의 숫자 t 가 주어진다.
- 그 후, t 줄 동안 리스트가 입력된다. 리스트의 원소들은 정수이며 공백으로 구분되어 있다. 원소가 존재하지 않는 경우는 없다.
- 각 리스트의 원소 갯수는 100,000개 이하이다.

출력 형식

- 각 테스트 케이스에서 입력 받은 리스트의 원소들의 합을 출력한다.

입력 예시

```
3
1 5 9 11 12
0 1 0 2 0 3 0 4 0 5
100 120 -20 -30
```

출력 예시

```
38
15
170
```

Time Complexity

프로그램 문제의 입력 형식에서는 보통 입력되는 데이터의 크기가 주어집니다.

- 형식에서 언급되는 크기 및 기타 조건에 대한 예외 처리는 생략하셔도 됩니다.
- N은 10만 이하의 자연수이다. → if $N \leq 100,000$: (필요 없음)

이스의 숫자 t 가 주어진다.

이스마다 정수 수열이 리스트로 주어진다. 원소들은 출발점에 순서대로 들어온 차량의 번호들을 의미하며, 서로 공백을 사이에 하나의 차량 번호는 리스트 내에서 반드시 두 번만 나타난다. 리스트의 크기는 200,000을 넘지 않는다

주어진 입력 데이터의 크기를 통해서 실습 문제에서 요구하는 시간복잡도가 추측 가능합니다.

n(데이터의 크기)	시간복잡도
10,000,000	$O(n)$
100,000	$O(n * \log n)$
1,000~5,000	$O(n^2)$

기본 라이브러리

알고리즘 실습에서는 Python 3가 제공하는 기본 라이브러리(math, heapq 등)만을 이용해서 실습을 하게 됩니다. 따라서 numpy 등을 사용할 수 없음을 주의해주세요.

※ 정적 배열(Array) 만드는 방법: List를 이용합니다. (Initial_Data는 직접 값을 넣는 곳)

- 크기 N의 1차원 배열: $Arr = [Initial_Data]*N$
- N*N 2차원 배열: $Matrix = [[Initial_Data]*N \text{ for } _ \text{ in range}(N)]$
- N*M 2차원 배열: $Matrix = [[Initial_Data]*M \text{ for } _ \text{ in range}(N)]$

※ 정해진 크기를 가지게 만들었으나, 언제든지 Append나 Pop 등을 이용하여 크기를 바꿀 수 있습니다.

01. A+B (1)

두 수를 입력 받아서 그 합을 출력하는 프로그램을 제출해봅시다.

* Hint: input() 함수를 사용할 때에 괄호 안에 문구가 있으면 해당 문구가 출력으로 인식되어 wrong answer가 나오게 됩니다.

- input('두 수를 입력 받습니다.') → X
- input() → O

01. A+B (1)

```
1 a, b = input().split()  
2 print(int(a) + int(b))
```

01. A+B (1)

```
a, b = map(int, input().split())  
print(a+b)
```

02. A+B (2)

테스트 케이스만큼 입력을 받아서, 각각의 A+B를 전부 구하는 프로그램을 구현합니다.

<입력 예시>

3 #테스트케이스 수
2 8 #1번 테스트
-7 2 #2번 테스트
23 -10 #3번 테스트

3
2 8
10
-7 2
-5
23 -10
13

<출력 예시>

10 #1번 테스트 답
-5 #2번 테스트 답
13 #3번 테스트 답

02. A+B (2)

```
t = int(input())  
for _ in range(t):  
    a,b = map(int,input().split())  
    print(a+b)
```

03. 리스트 합

테스트 케이스만큼 리스트를 입력 받고, 그 리스트의 합을 출력하는 프로그램을 작성합니다.

03. 리스트 합

```
def Sum(l):  
    s = 0  
    for i in l:  
        s = s + i  
    return s
```

```
t = int(input())  
for _ in range(t):  
    l = list(map(int, input().split()))  
    print(Sum(l))
```

03. 리스트 합

```
t = int(input())  
for _ in range(t):  
    l = list(map(int, input().split()))  
    print(sum(l))
```

04. 최댓값과 최솟값의 차이 구하기

테스트 케이스만큼 리스트를 입력 받습니다.

리스트에서 최댓값과 최솟값을 구하고, 그 차를 출력하는 프로그램을 작성합니다.

※ max와 min 함수를 사용한다면 쉽게 해결할 수 있습니다.

※ max와 min 함수를 직접 구현을 해보는 것도 많은 도움이 될 것입니다.

04. 최댓값과 최솟값의 차이 구하기

```
def Max(l):  
    m = l[0]  
    for i in l:  
        if m < i:  
            m = i  
    return m
```

```
def Min(l):  
    m = l[0]  
    for i in l:  
        if m > i:  
            m = i  
    return m
```

```
t = int(input())  
for _ in range(t):  
    l = list(map(int, input().split()))  
    print(Max(l) - Min(l))
```

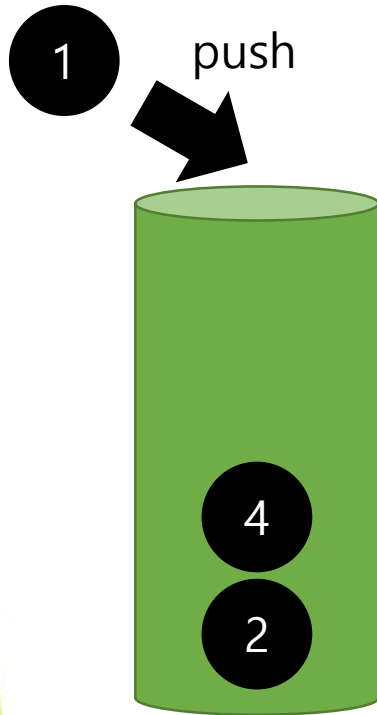
04. 최댓값과 최솟값의 차이 구하기

```
t = int(input())  
for _ in range(t):  
    l = list(map(int, input().split()))  
    print(max(l) - min(l))
```

05. 스택 구현하기

가장 기본적이고 중요한 자료 구조인 **스택**을 구현합니다.

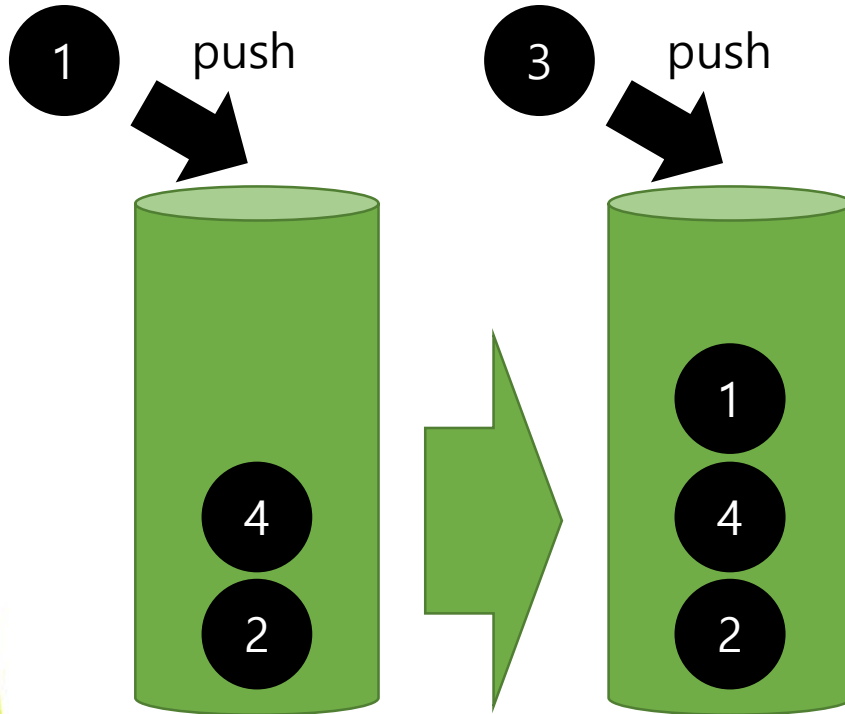
* Hint: list의 pop() 함수를 사용합시다!



05. 스택 구현하기

가장 기본적이고 중요한 자료 구조인 **스택**을 구현합니다.

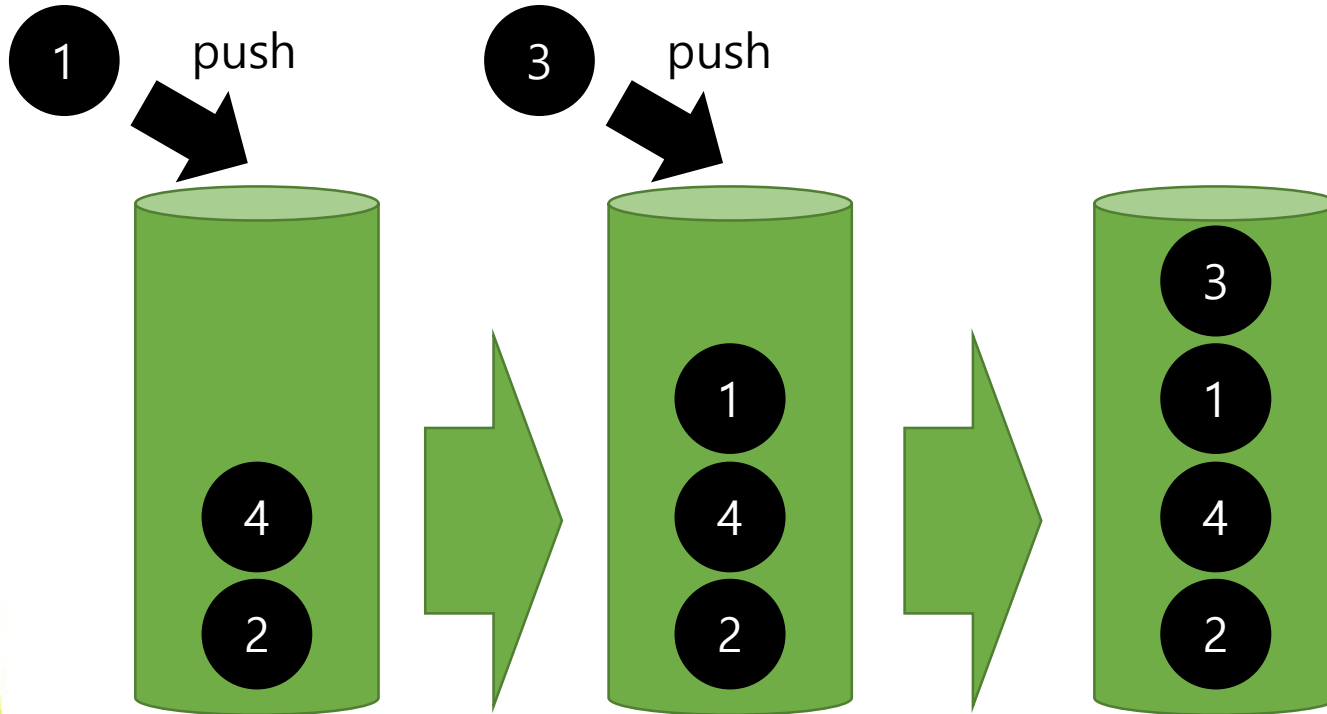
* Hint: list의 pop() 함수를 사용합시다!



05. 스택 구현하기

가장 기본적이고 중요한 자료 구조인 **스택**을 구현합니다.

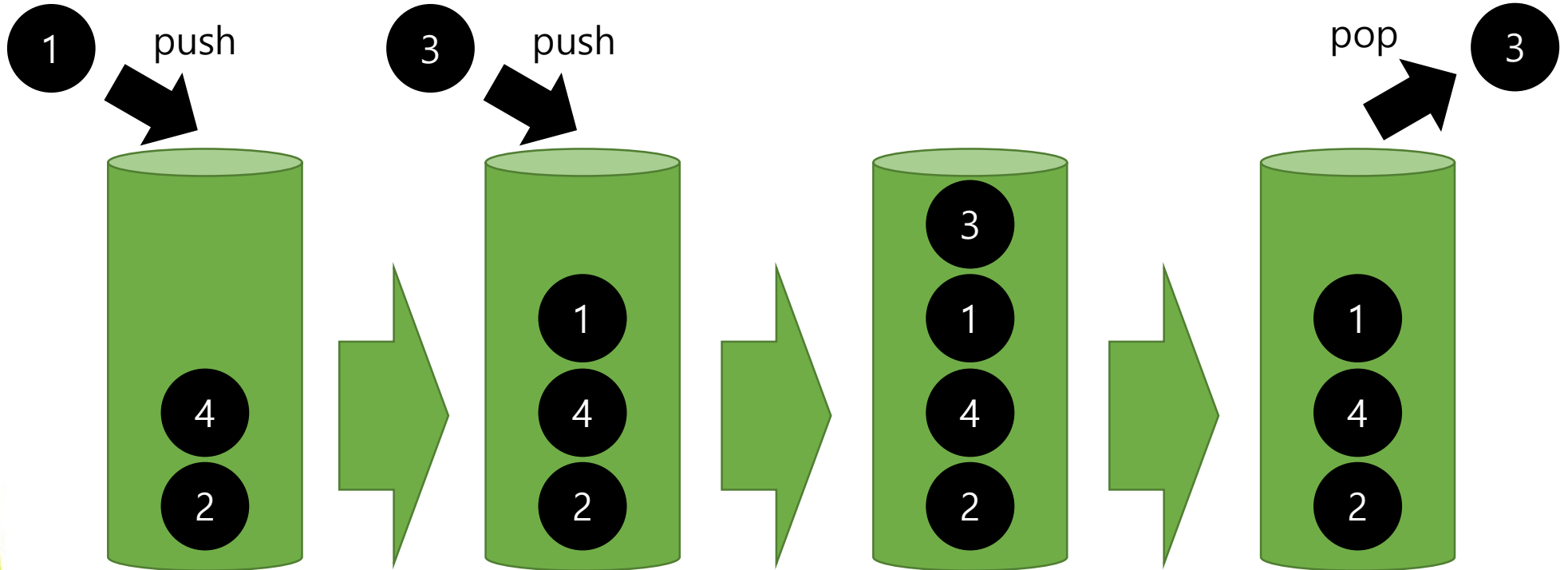
* Hint: list의 pop() 함수를 사용합시다!



05. 스택 구현하기

가장 기본적이고 중요한 자료 구조인 **스택**을 구현합니다.

* Hint: list의 pop() 함수를 사용합시다!



05. 스택 구현하기

```
for _ in range(int(input())):
    stack = []
    qry = list(map(int, input().split()))
    for q in qry:
        if q > 0 :
            stack.append(q)
        else :
            print(stack.pop(), end = ' ')
    print() # 다음 테스트 케이스 출력을 위해 줄 띄우기
```

05. 스택 구현하기

```
for _ in range(int(input())):
    stack = []
    answer = []
    qry = list(map(int, input().split()))
    for q in qry:
        if q > 0 :
            stack.append(q)
        else :
            answer.append(stack.pop())
    print(*answer)
```

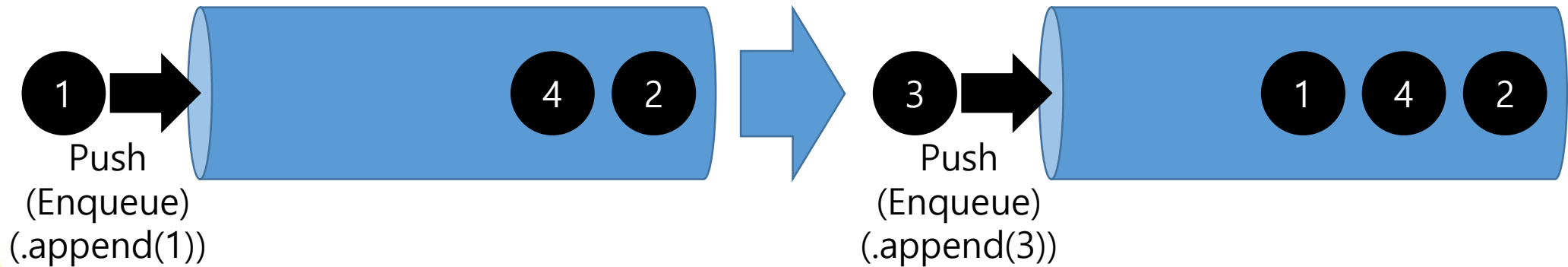
06. 큐 구현하기

가장 기본적이고 중요한 자료 구조인 **큐**를 구현합니다.



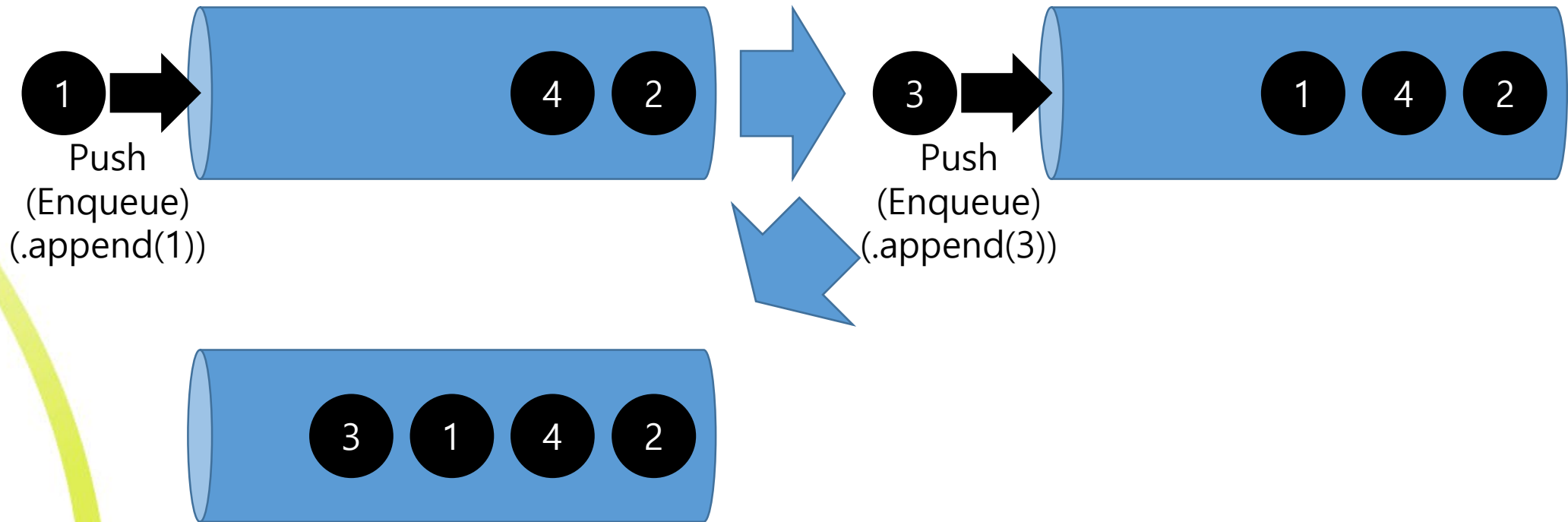
06. 큐 구현하기

가장 기본적이고 중요한 자료 구조인 **큐**를 구현합니다.



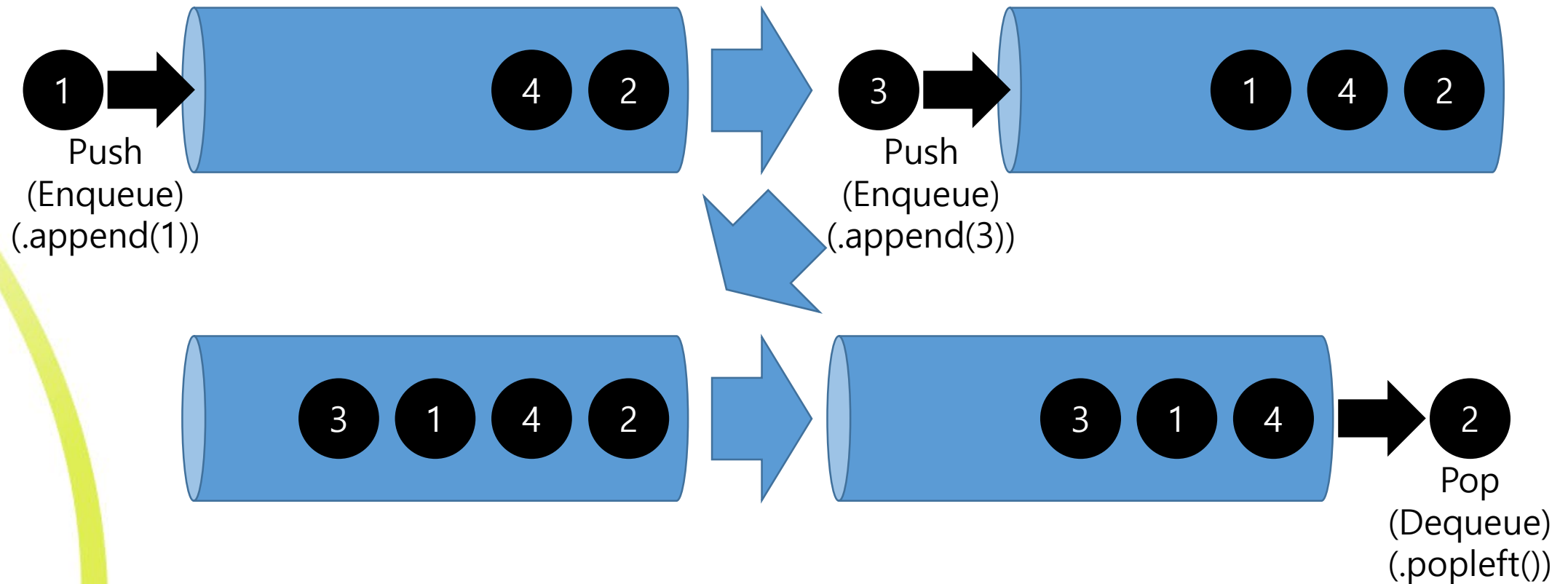
06. 큐 구현하기

가장 기본적이고 중요한 자료 구조인 **큐**를 구현합니다.



06. 큐 구현하기

가장 기본적이고 중요한 자료 구조인 **큐**를 구현합니다.



06. 큐 구현하기

```
import collections

for _ in range(int(input())):
    queue=collections.deque([])
    qry = list(map(int, input().split()))
    for q in qry:
        if q > 0 :
            queue.append(q)
        else :
            print(queue.popleft())
```

06. 큐 구현하기

List를 이용하여 구현하는 방법

```
import collections

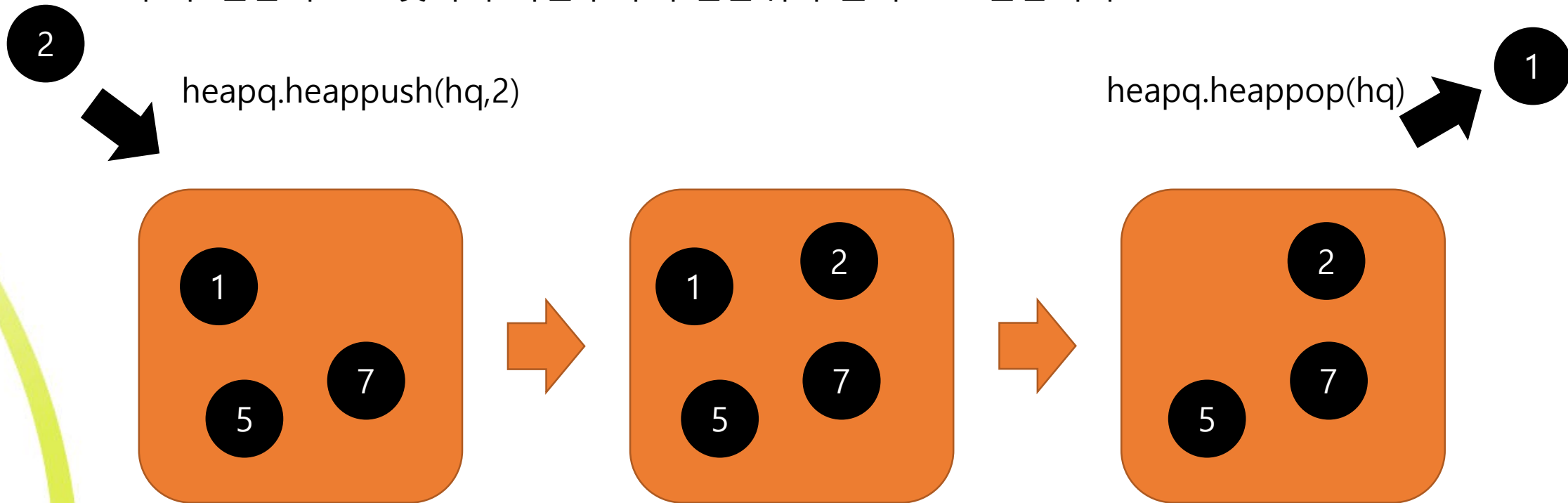
for _ in range(int(input())):
    queue = [] # 테스트 케이스마다 빈 큐로 초기화
    head = 0 # 큐의 가장 왼쪽 인덱스
    tail = 0 # 큐의 가장 오른쪽 인덱스 + 1
    qry = list(map(int, input().split()))
    for q in qry:
        if q > 0 :
            queue.append(q)
            tail += 1 # 오른쪽 인덱스 업데이트
        else :
            print(queue[head])
            head += 1 # 왼쪽 인덱스 업데이트
```

07. 우선순위 큐 사용하기

가장 기본적이고 중요한 자료 구조 중 하나인 **우선순위 큐**를 사용합니다.

* heapq를 이용해 보도록 합니다.

* 주의: 일반적으로 숫자가 작을수록 우선순위가 높다고 표현합니다.



07. 우선순위 큐 사용하기

```
import heapq

for _ in range(int(input())):
    hq=[]
    qry = list(map(int, input().split()))
    for q in qry:
        if q > 0 :
            heapq.heappush(hq,q)
        else :
            print(heapq.heappop(hq))
```

ADD01. 괄호 검사

괄호가 올바른 괄호열인지 체크하는 프로그램을 작성합니다.
괄호의 개수는 **20만개**를 넘지 않습니다.

올바른 괄호열

((()))

((){}[])

올바르지 못한 괄호열

{}

[()]

((((([]))

))))(((

[[[[]]

Hint : 스택을 이용합니다!

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })

스택: []



ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: ['(']

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: ['(' '[']

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: ['(']
'('와 ')'의 괄호열이 맞음

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: ['(' '{']

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: ['(' '{' '(']

ADD01. 괄호 검사

Idea: 왼쪽 괄호가 나오면 stack에 **push**,
오른쪽 괄호가 나오면 stack에서 **pop** 한 뒤 비교하자!

([] { () })



스택: []

순서대로 ')', '}', ')'이 스택에서 pop한 왼쪽 괄호들과 맞음

ADD01. 괄호 검사

구현할 때 더 생각해야할 점:

1. 만약 괄호 검사가 끝난 시점에서 스택이 비어있지 않다면?
Ex: ((())
2. 오른쪽 괄호를 처리할 때 스택이 비어 있어서 pop을 할 수 없다면?
Ex:)

ADD01. 괄호 검사

```
t = int(input())

for _ in range(t):
    bracket = input()
    stk = []      #스택 초기화
    correct = True #올바른 괄호열인지 저장하는 변수
    for b in bracket:
        if                #여는 괄호의 경우, 스택에 넣음

        elif              #스택이 비어있거나, pop된 괄호와 매칭 x인 경우

        elif              :

        elif              :

    if      #괄호열이 끝났는데, 닫혀지지 않은 괄호가 남은 경우
        correct = False
    if      :
        print('YES')
    else:
        print('NO')
```

ADD02. 두 바퀴 레이스

각 차량의 첫 번째 바퀴를 순위와 두 번째 바퀴 순위가 같은지 체크해봅니다.
숫자의 개수는 **20만개**를 넘지 않습니다.

- 입력 데이터 내에서 **하나의 숫자는 반드시 두 번만** 나타납니다. (다른 예외 고려 X)

같으면

1	5	2	1	5	3	2	4	3	4
---	---	---	---	---	---	---	---	---	---

 ➡ NO

다르면

6	3	6	3	7	5	7	9	9	5
---	---	---	---	---	---	---	---	---	---

 ➡ YES

힌트 : 큐를 이용합니다!

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: []

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [1 5 2]

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [5 2]

큐의 맨 첫번째 원소가 1이었으니 pop

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [2]

큐의 맨 첫번째 원소가 5였으니 pop

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [2 3]

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [3]

큐의 맨 첫번째 원소가 2였으니 pop

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: [3 4]

ADD02. 두 바퀴 레이스

Idea: 첫 번째 바퀴를 마친 차는 queue에 **push**하고,
두 번째 바퀴를 마친 차는 queue에서 **pop**하자



큐: []
순서대로 3, 4 pop

ADD02. 두 바퀴 레이스

구현할 때 더 생각해야할 점:

1. 만약 queue가 비어 있다면?
2. 레이스에서 순서가 바뀌었다면 검사 종료 후 queue는 어떤 상태일까?

ADD02. 두 바퀴 레이스

```
from collections import deque

t = int(input())

for _ in range(t):
    car = list(map(int, input().split()))
    queue = deque()  # 큐 초기화
    for c in car:
        if queue and queue[0] < c:  # 큐가 비어있거나, 첫 원소가 c가 아닌 경우
            queue.popleft()
        else:
            queue.append(c)

    if queue[0] < queue[-1]:  # 순서가 바뀐 경우, 큐가 비어있지 않음
        print('YES')
    else:
        print('NO')
```