알고리즘 이해 및 구현

류영표 강사
ryp1662@gmail.com

강사 소개

류영표

Youngpyo Ryu

동국대학교 수학과/응용수학 석사수료

前 Upstage AI X 네이버 부스트 캠프 AI tech 1~3기 멘토

前 Innovation on Quantum & CT(IQCT) 이사

前 한국파스퇴르연구소 Image Mining 인턴(Deep learning)

前 ㈜셈웨어(수학컨텐츠, 데이터 분석 개발 및 연구인턴)

강의 경력

- 현대자동차 연구원 강의 (인공지능/머신러닝/딥러닝/강화학습)
- ㈜ 모두의연구소 Aiffel 1기 퍼실리테이터(인공지능 교육)
- 인공지능 자연어처리(NLP) 기업데이터 분석 전문가 양성과정 멘토
- 공공데이터 청년 인턴 / SW공개개발자대회 멘토
- 고려대학교 선도대학 소속 30명 딥러닝 집중 강의
- 이젠 종로 아카데미(파이썬, ADSP 강사)
- 최적화된 도구(R/파이썬)을 활용한 애널리스트 양성과정(국비과정) 강사
- 한화, 하나금융사 교육
- 인공지능 신뢰성 확보를 위한 실무 전문가 자문위원
- 보건 · 바이오 AI활용 S/W개발 및 응용전문가 양성과정 강사
- Upstage AI X KT 융합기술원 기업교육 모델최적화 담당 조교

주요 프로젝트 및 기타사항

- 개인 맞춤형 당뇨병 예방·관리 인공지능 시스템 개발 및 고도화(안정화)
- 폐플라스틱 이미지 객체 검출 경진대회 3위
- 인공지능(AI)기반 데이터 사이언티스트 전문가 양성과정 1기 수료
- 제 1회 산업 수학 스터디 그룹 (질병에 영향을 미치는 유전자 정보 분석)
- 제 4,5회 산업 수학 스터디 그룹 (피부암, 유방암 분류)
- 빅데이터 여름학교 참석 (혼잡도를 최소화하는 새로운 노선 건설 위치의 최적화 문제)

알고리즘, Algorithm

- 수학과 컴퓨터과학, 언어학 또는 엮인 분야에서 어떠한 문제를 해결하기 위해 정해진 일련의 절차
- 9세기 페르시아의 수학자인 무함마드 알콰리즈미의 이름을 라틴어화한 알고리스무스에서 유래한 표현
- 문제 해결을 위해 여러 개의 후보 알고리즘 중, 정확성과 효율성 등을 평가한 후에 최적화 알고리즘을 선택

좋은 알고리즘의 분석 기준

- 정확성 : 적당한 입력에 대해서 유한 시간 내에 올바른 답을 산출하는 가를 판단
- 작업량 : 전체 알고리즘에서 수행되는 가장 중요한 연산들만으로 작업량 측정. 해결하고자 하는 문제의 중요 연산이 여러 개인 경우에는 각각의 중요 연산들의 합으로 간주하거나 중요 연산들에 가중치를 두어 계산
- 최적성 : 그 알고리즘보다 더 적은 연산을 수행하는 알고리즘은 없는가? 최적이란 가장 “잘 알려진” 이 아니라 가장 좋은 의미.
- 복잡도

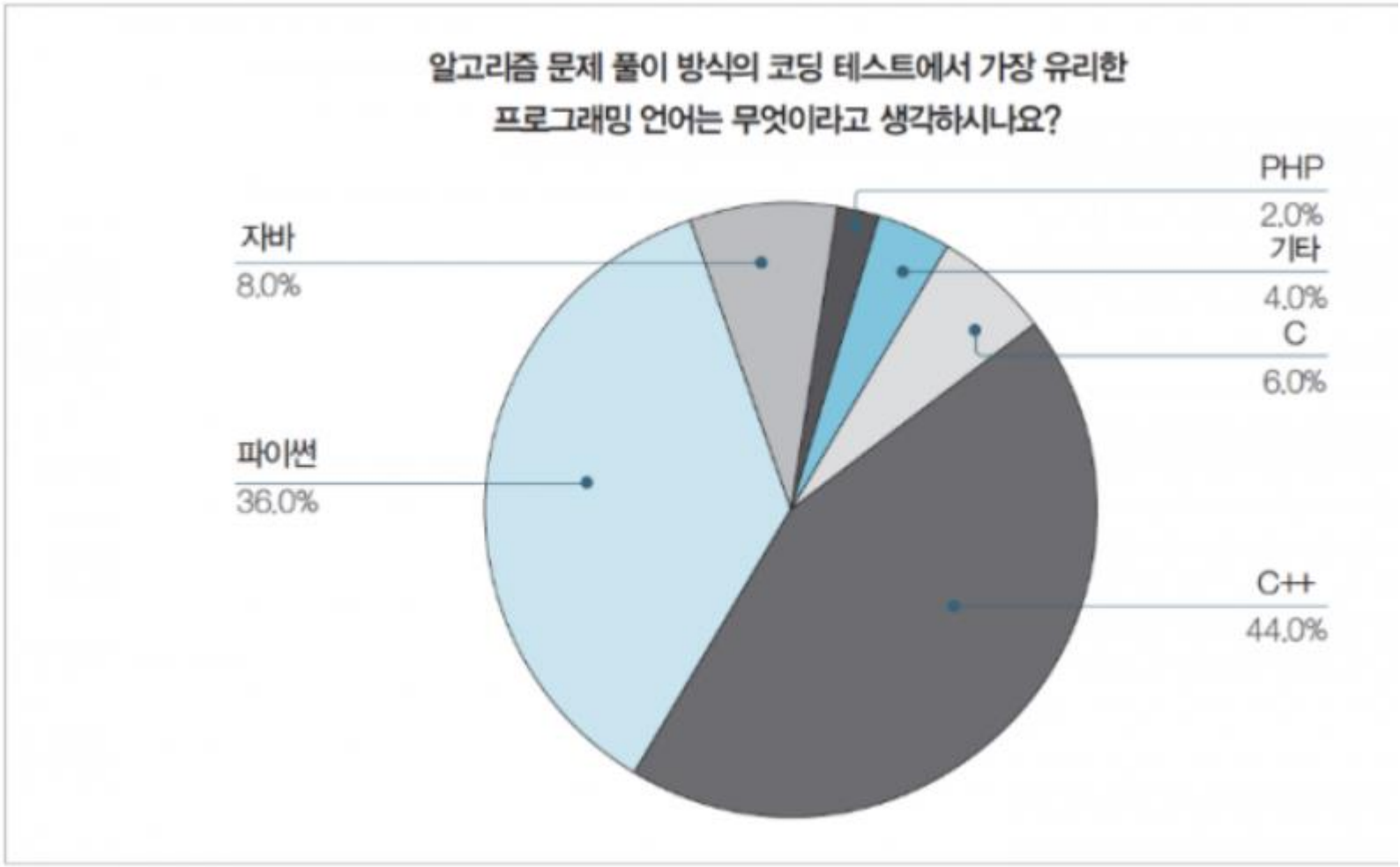
출처 : https://6u2ni.tistory.com/41

코딩 테스트

- 주어진 시간동안 주어진 문제를 요구사항에 맞게, 프로그래밍하여 ACCEPT나 점수를 받는 시험.
- 개발자로서의 기초 역량과 소양을 보기 위한 시험.
- 개발자 채용을 위한 과정중 하나로 '현업에서 개발할 수 있는 사람인가'를 확인하고자 하는 것.

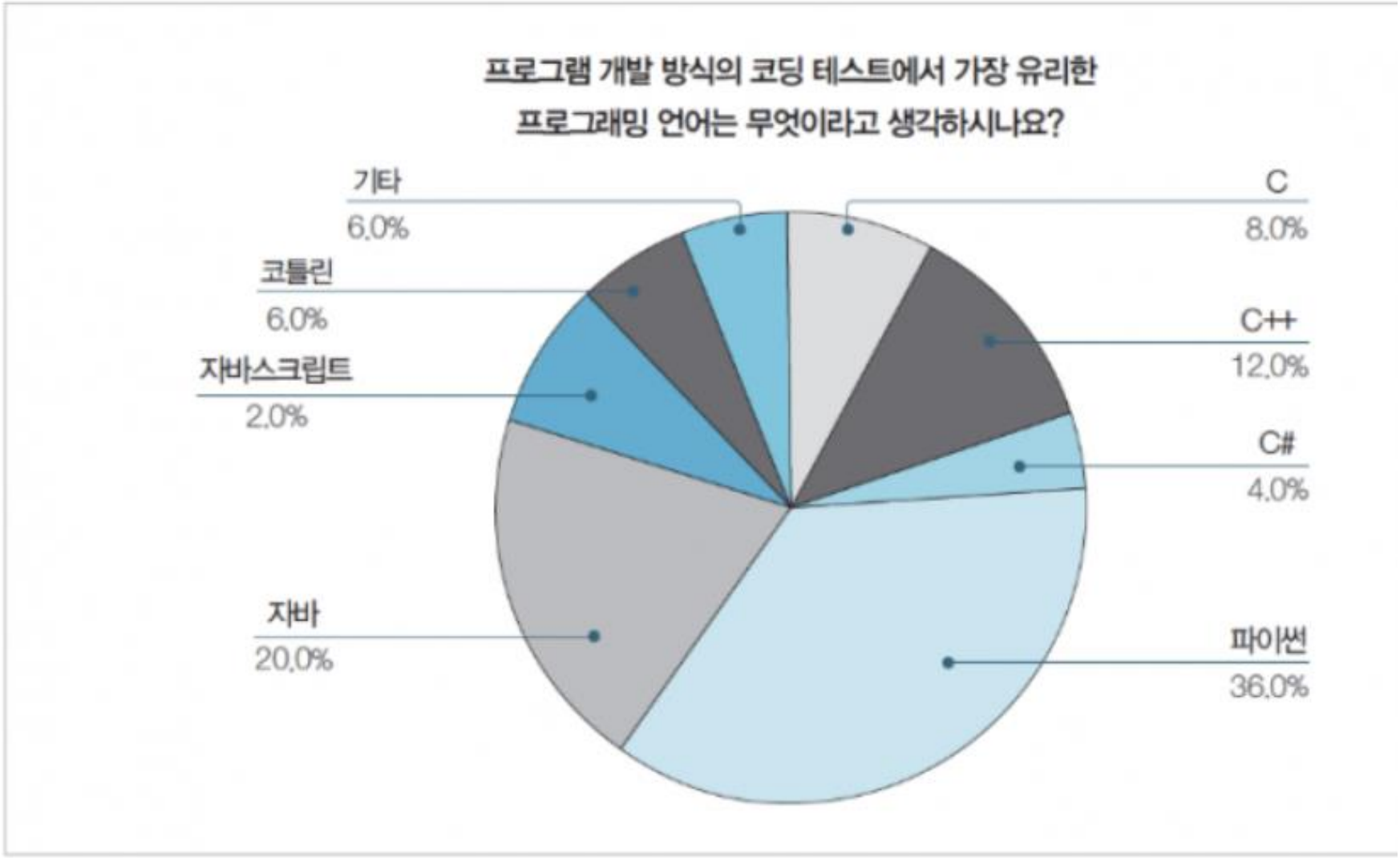
- 요구사항과 맞게 작성을 해야 함.

코딩 테스트



[출처: 이것이 취업을 위한 코딩 테스트다 with 파이썬(책)]

프로그램 개발 방식의 코딩 테스트



[출처: 이것이 취업을 위한 코딩 테스트다 with 파이썬(책)]

Why Python

TIOBE Index

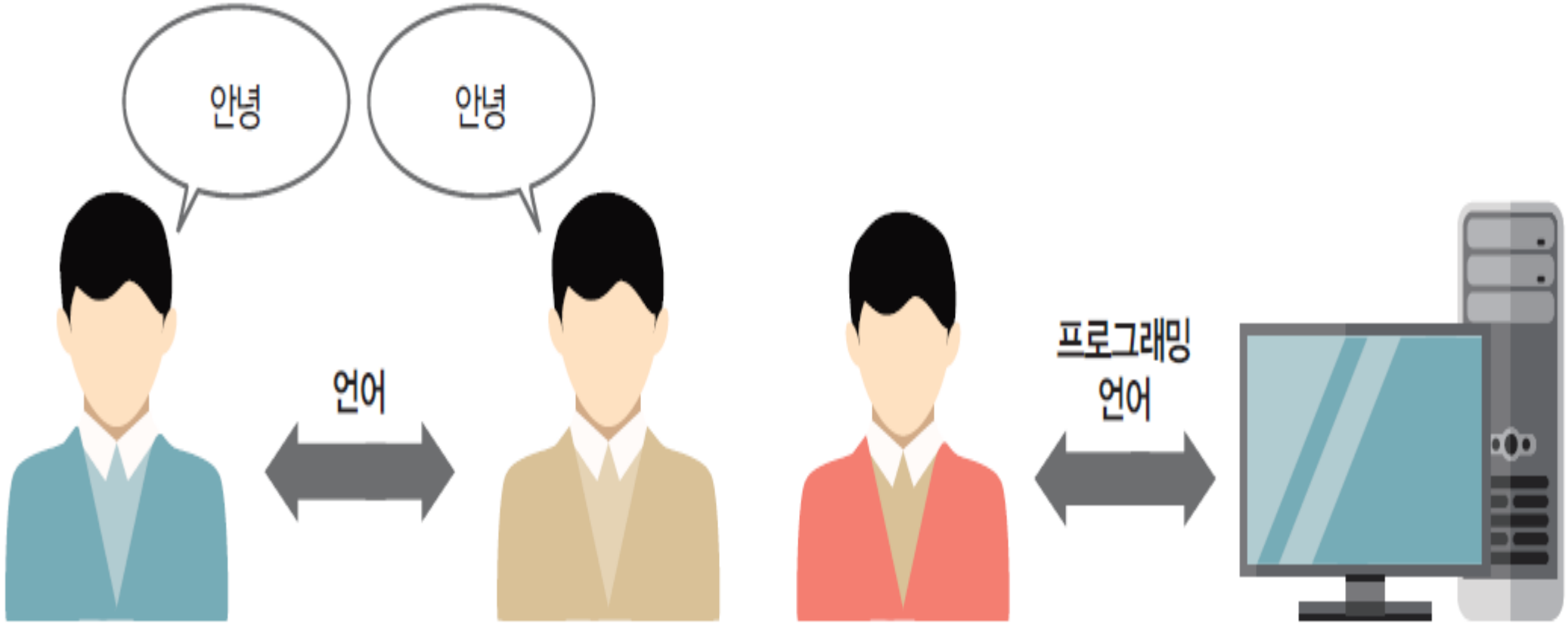
May 2021	May 2020	Change	Programming language	Ratings	Change
1	1		C	13.38%	-3.68%
2	3	↑	Python	11.87%	+2.75%
3	2	↓	Java	11.74%	-4.54%
4	4		C++	7.81%	+1.69%
5	5		C#	4.41%	+0.12%
6	6		Visual Basic	4.02%	-0.16%
7	7		JavaScript	2.45%	-0.23%
8	14	↑ ↑	Assembly language	2.43%	+1.31%
9	8	↓	PHP	1.86%	-0.63%
10	9	↓	SQL	1.71%	-0.38%
11	15	↑ ↑	Ruby	1.50%	+0.48%
12	17	↑ ↑	Classic Visual Basic	1.41%	+0.53%
13	10	↓	R	1.38%	-0.46%
14	38	↑ ↑	Groovy	1.25%	+0.96%
15	13	↓	MATLAB	1.23%	+0.06%
16	12	↓ ↓	Go	1.22%	-0.05%
17	23	↑ ↑	Delphi/Object Pascal	1.21%	+0.60%
18	11	↓ ↓	Swift	1.14%	-0.65%
19	18	↓	Perl	1.04%	+0.16%
20	34	↑ ↑	Fortran	0.83%	+0.51%

PYPL Index (Worldwide)

May 2021	Change	Programming language	Share	Trends
1		Python	29.9 %	-1.2 %
2		Java	17.72 %	-0.0 %
3		JavaScript	8.31 %	+0.4 %
4		C#	6.9 %	-0.1 %
5	↑	C/C++	6.62 %	+0.9 %
6	↓	PHP	6.15 %	+0.1 %
7		R	3.93 %	+0.0 %
8		Objective-C	2.52 %	+0.1 %
9		Swift	1.96 %	-0.2 %
10	↑	TypeScript	1.89 %	+0.0 %
11	↓	Matlab	1.71 %	-0.2 %
12		Kotlin	1.62 %	+0.1 %
13	↑	Go	1.42 %	+0.1 %
14	↓	VBA	1.33 %	-0.0 %
15	↑ ↑ ↑	Rust	1.13 %	+0.4 %
16	↓	Ruby	1.12 %	-0.1 %
17	↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑	Ada	0.72 %	+0.3 %
18	↓	Visual Basic	0.7 %	-0.2 %
19	↓ ↓ ↓	Scala	0.67 %	-0.4 %
20	↓	Abap	0.61 %	+0.1 %
21	↓	Dart	0.55 %	+0.0 %
22	↑ ↑	Lua	0.49 %	+0.1 %
23	↑ ↑ ↑	Julia	0.42 %	+0.1 %
24	↓ ↓ ↓	Groovy	0.41 %	-0.0 %
25	↓ ↓ ↓	Perl	0.4 %	-0.0 %
26	↓ ↓ ↓	Cobol	0.36 %	-0.1 %
27	↑	Delphi/Pascal	0.24 %	-0.0 %
28	↓	Haskell	0.21 %	-0.1 %

출처 : https://statisticstimes.com/tech/top-computer-languages.php

What is programming?



출처 : https://ubi-sonagi.tistory.com/45

코딩 교육

Fig. 1: 코딩교육은 왜 필요할까요?(코딩으로 준비하는 미래)

"세상의 모든 사람은 컴퓨터 프로그래밍을 배워야 합니다.
프로그래밍은 생각하는 법을 가르쳐 주기 때문입니다."

"Everybody in this country should learn to program a computer,
because it teaches you how to think"

- 스티브 잡스 -

"프로그램은 사고력과 문제해결력을 향상시킵니다."

"Learning to write programs stretches your mind,
and helps you think better."

- 빌 게이츠 -

"코딩 교육은 당신의 미래일 뿐만 아니라 조국의 미래다."

"Learning these skills isn't just important for your future,
it's important for our country's future."

- 버락 오바마 -

출처: 로보로보, google.com, BNK투자증권

파이썬 이란?

- 1991년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발
- C, C++, 자바 등 어떤 컴퓨터 프로그래밍 언어보다 배우기 쉬움
- 직관적이고 이해하기 쉬운 문법
- 객체 지향의 고수준 언어
- 앱(App)과 웹(WEB) 프로그램 개발 목적
- 웹 서버, 과학 연산, 사물 인터넷(IOT), 인공지능, 게임 등의 프로 그램 개발하는 강력한 도구

Python의 특징

C++

```
#include <iostream>
using namespace std;
int main() {
    cout<<"Hello, gabia!";
    return 0;
}
```

Java

```
public class HelloGabia {
    public static void main(String args[]) {
      System.out.println("Hello, gabia!");
    }
}
```

Python

```
print("Hello, gabia!")
```

C++

```
#include <iostream>
using namespace std;
int main() {
    int arr[5] = {1, 2, 3, 4, 5};

    for(int i=0; i<5; i++) {
        cout << arr[i] << endl;
    }
    return 0;
}
```

Java

```
public class HelloGabia {
    public static void main(String args[]) {
      int arr[] = {1, 2, 3, 4, 5};

      for(int i=0; i<5; i++) {
          System.out.println(arr[i]);
      }
    }
}
```

Python

```
arr = [1, 2, 3, 4, 5]
for el in arr:
    print(el)
```

출처 : https://library.gabia.com/contents/9256/

Python

출처 : https://www.youtube.com/watch?v=3DuyJf_XPtM

Google colab 사용법

- 풀 네임은 Google Colaboratory
- Google Drive + Jupyter Notebook
    - Google Drive처럼 협업 가능(동시에 수정 가능)
- https://colab.research.google.com/로 접속시 사용 가능
- 컴퓨터 사양(21년 01월 기준)
    - Ubuntu 18.04.5 LTS
    - CPU 제논 2.3GHz
    - 메모리 13G
    - **GPU : K80 또는 T4** :
    - TPU도 사용 가능
- GPU 사용시 최대 12시간 (Colab은 GPU를 무료로 사용가능)
- Github의 소스 코드를 Colab에서 사용 가능

Google colab

OS 확인

- !cat /etc/issue.net

하드웨어 사양

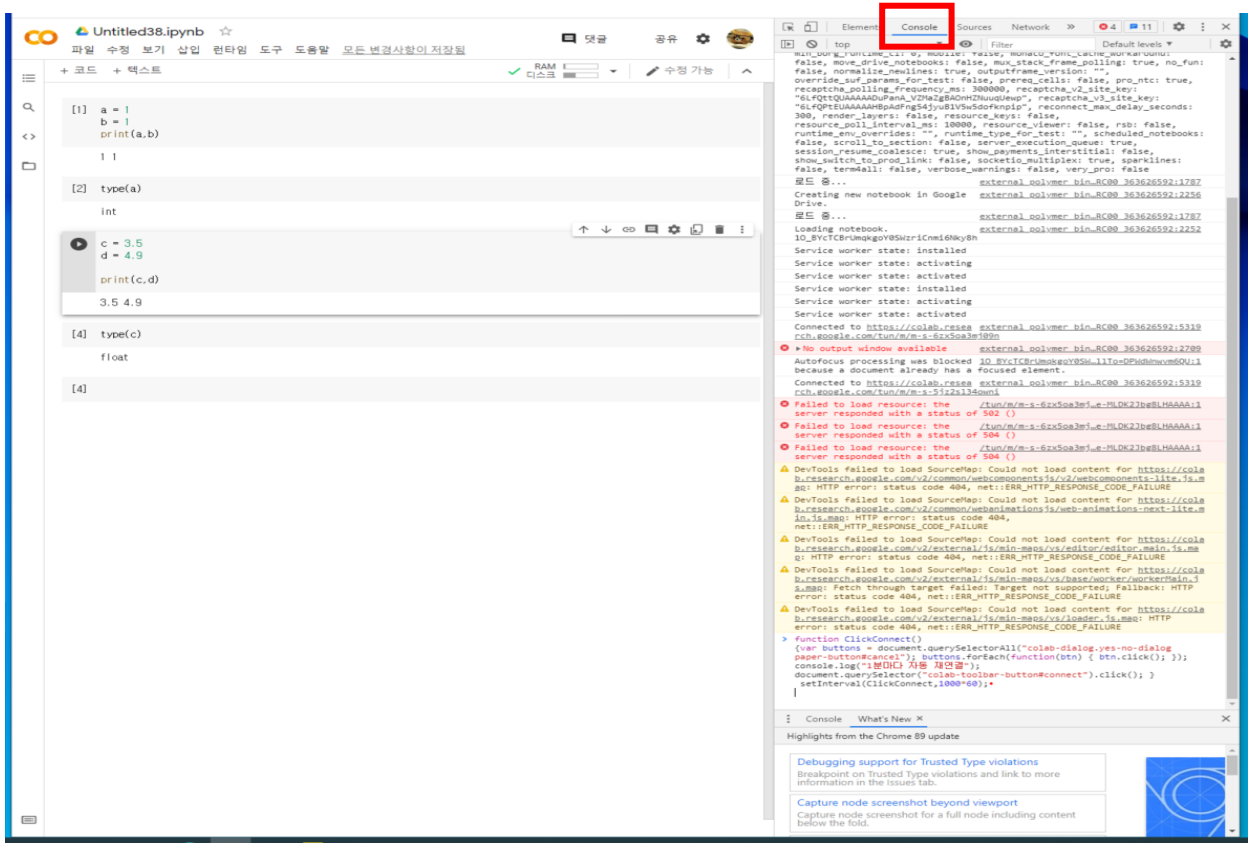
1. CPU
   - !cat /proc/cpuinfo
2. Memory
   - !cat /proc/meminfo
3. Disk
   - !df -h
4. GPU
   - !nvidia-smi

Google colab 런타임 방지

Google colab의 전체 세션 유지 시간은 12시간이고, 90분 이상 비 활성화 되어 있으면 자동으로 끊기고, 몇 분 동안 사용을 안하면 자동으로 세션이 끊긴다.

->방법 : 크롬 브라우저에서 F12(혹은 Ctrl+shift+i)를 눌러서 개발자 도구 창을 열고 console 창에서 입력해 주면 된다.

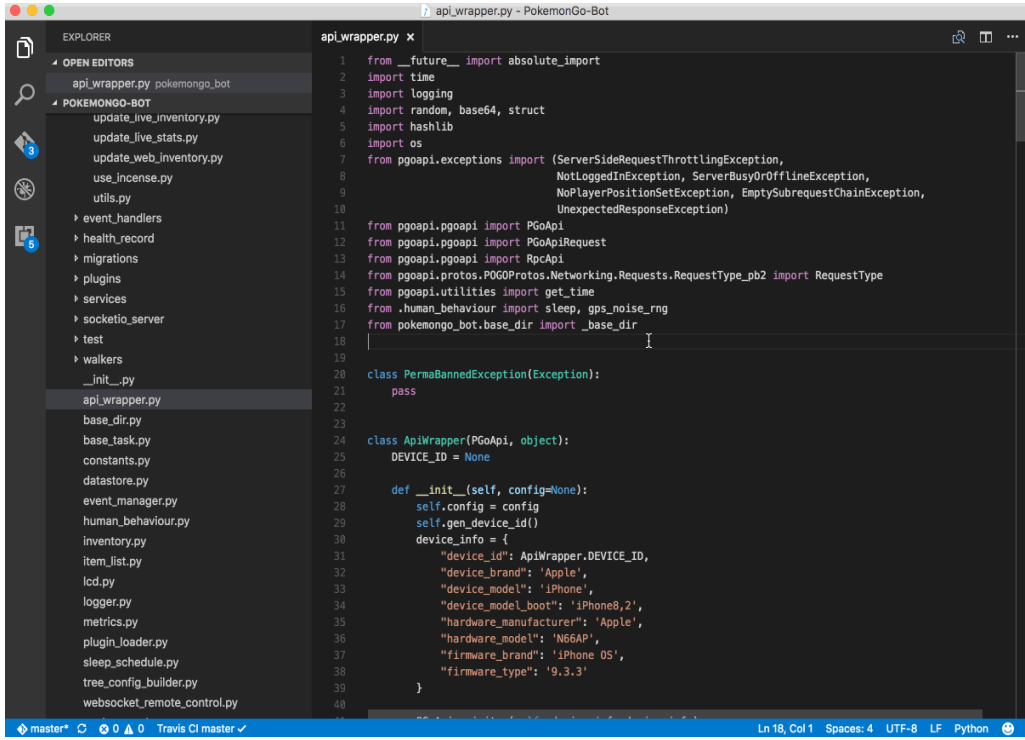
```
function ClickConnect()
{var buttons = document.querySelectorAll("colab-dialog.yes-no-dialog paper-button#cancel");
buttons.forEach(function(btn) { btn.click(); });
console.log("1분마다 자동 재연결");
document.querySelector("colab-toolbar-button#connect").click(); }
 setInterval(ClickConnect,1000*60);
```



Python IDE VS code 이란?

- VS Code의 장점

1. MS의 친절한 관리
   - MS의 프로그램은 비싸지만 그만큼 친절함.
2. 무료 정책
3. 점유율 1위
4. 마이크로 소프트가 직접 배포하는 확장성
5. 언어 확장성이 좋음(C,C++,JAVA 등 지원함)



Python IDE Pycharm 이란?

- Pycharm은 Python 언어에 사용되는 JetBrains사에 의해 개발된 IDE입니다.
- IDE는 컴파일, 코드 편집기,디버그, 배포 등의 모든 작업을 하나의 프로그램에서 할 수 있도록 해주는 통합 개발 환경

Pycharm 의 특징

1. 코드 분석 및 코딩 지원
2. 프로젝트 및 코드 탐색
3. 파이썬
4. 웹 프레임 워크 지원(Professional 전용, 유료버전)
5. 통합 파이썬 디버그
6. 라인 단위 테스트
7. Google App Engine python(Professional 전용, 유료버전)
8. 과학적 도구 지원(Professional 전용, 유료버전)

Python 속도

“2~5배 더 빠르게”… 귀도 반 로섬, 파이썬 속도 개선한다

Paul Krill | InfoWorld

파이썬(Python) 창시자 귀도 반 로섬이 파이콘 2021(Pycon 2021)에서 진행된 파이썬 랭귀지 서밋(Python Language Summit)에서 파이썬 속도를 2배에서 5배까지 더 빠르게 만들기 위한 단기 및 장기 계획을 발표했다.

ⓒGetty Images

파이썬에는 이미 파이파이(PyPy)와 같은 대체 런타임부터 C/C++로 작성된 랩핑 모듈까지 더 빠르게 실행할 수 있는 여러 방법이 있다. 하지만 C로 작성된 파이썬 참조 구현체이자 가장 널리 사용되는 언어 버전인 C파이썬(CPython) 자체의 속도를 높이는 방법은 거의 없다는 게 반 로섬의 설명이다.

출처 : https://www.ciokorea.com/news/193909

코딩 테스트 준비 사이트

- 해커링크(https://www.hackerrank.com)
- 코딜리티(https://www.codility.com)
- 리모트인터뷰 (https://www.remoteinterview.io)
- 리트코드 (https://www.letecode.com)
- 프로그래머스(국내) (https://www.programmers.co.kr)
- 백준(국내)(https://www.acmicpc.net)
- 코딩도장(https://dojang.io/)
- SW Expert Academy(https://swexpertacademy.com/main/main.do)

프론트 엔드 VS 백엔드

	프론트엔드	백엔드
정의	사용자들에게 보여지는 부분 Ex) 레이아웃, 텍스트, 그림 등	사용자들에게 보이지 않는 시스템 안쪽 부분 Ex) 내부적 DB
기술	**HTML&CSS** *HTML : 콘텐츠에 구조와 의미를 주기 위한 것 *CSS : 콘텐츠의 모양과 스타일을 주기 위한 것	**스프링프레임워크** *동적인 웹사이트 개발을 위한 여러가지 서비스 제공
	JAVA script *웹페이지에 기능을 더해 HTML 웹페이지를 동적이고 살아있게 만드는 것	**JAVA** *컴퓨터가 이해할 수 있는 0과 1로 이루어진 기계어
직무	**UI/UX 디자인** *UX디자인 : 사용자가 경험하게 될 전반적 디자인 *UI디자인 : 겉으로 시각화 되는 디자인	**서버코딩** *시스템이 운용되기 위해 필요한 시스템의 관련 기능 구현 등을 담당
	Front-end 개발 *웹상의 표현을 도와주는 여러 기술을 활용하여 UX와 Web 디자인 요소를 구현	**DB 활용** *데이터베이스를 구축하고 운영

출처 : https://www.jobindexworld.com/contents/view/4270 / https://www.facebook.com/coding.comu/posts/372409044728656

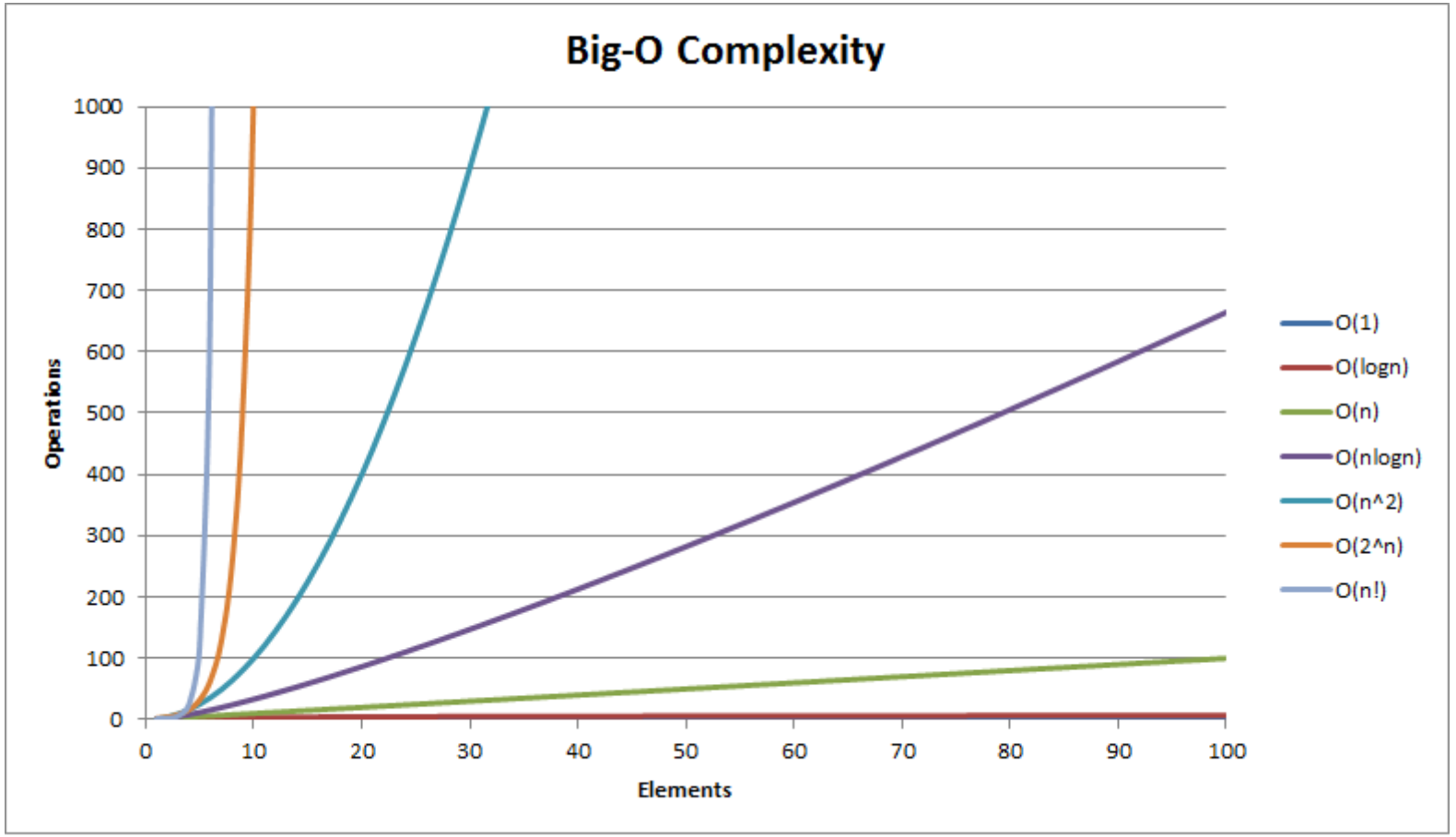
Time Complexity(시간 복잡도)

- 알고리즘의 효율성을 판단하기 위한 지표로서, 프로그램 수행에 걸리는 절대적 시간이 아닌, 알고리즘을 수행하는데 사용되는 연산들이 몇 번 이루어지는가에 대한 것을 상대적 지표로 나타낸 것.
- 연산에는 산술, 대입, 비교, 이동이 있다.
- 시간복잡도, 즉 성능 측정에 사용되는 표기법은 Big-O, Big-Omega(Ω), Theta(Θ) 크게 세가지로 나눔.

Big-O

- 빅오는 점근적 실행 시간(Asymptotic Running Time)를 표기할 때, 가장 널리 쓰이는 수학적 표기법 중 하나.
- 점근적 실행 시간이란, 입력값 n이 커질 때, 즉 입력값이 무한대를 향할 때 $\lim_{n \to \infty}$ 함수의 실행 시간의 추이를 의미.
- 알고리즘은 궁극적으로 컴퓨터로 구현되므로, 컴퓨터의 빠른 처리 능력을 감안하면, 아무리 복잡한 알고리즘도 입력의 크기가 작으면 금방 끝나게 됨. 충분히 큰 입력에서는 알고리즘의 효율성에 따라 수행 시간이 크게 차이가 날 수 있음.

Big-O


Big-O Complexity
Operations
Elements
0
100
200
300
400
500
600
700
800
900
1000
0
10
20
30
40
50
60
70
80
90
100
O(1)
O(logn)
O(n)
O(nlogn)
O(n^2)
O(2^n)
O(n!)

Big-O

Big-O 표기법별 대표 알고리즘

- O(1) : Operation push and pop on stack
- O(logn) : Binary Tree
- O(n) : for loop
- O(n × log n) : Quick sort, Merge Sort, Heap Sort
- O(n2) : Double for loop, Insert Sort, Bubble Sort, Selection Sort
- O(n^2) : Fibonacci Sequence
- O(n!) : 외판원 문제(Travelling Saleman problem)

Big-O

Sorting Algorithms

Sorting Algorithms	Space complexity	Time complexity		
	Worst case	Best case	Average case	Worst case
Insertion Sort	O(1)	O(n)	$O(n^2)$	$O(n^2)$
Selection Sort	O(1)	$O(n^2)$	$O(n^2)$	$O(n^2)$
Smooth Sort	O(1)	O(n)	O(n log n)	O(n log n)
Bubble Sort	O(1)	O(n)	$O(n^2)$	$O(n^2)$
Shell Sort	O(1)	O(n)	$O(n \log n^2)$	$O(n \log n^2)$
Mergesort	O(n)	O(n log n)	O(n log n)	O(n log n)
Quicksort	O(log n)	O(n log n)	O(n log n)	O(n log n)
Heapsort	O(1)	O(n log n)	O(n log n)	O(n log n)

Data Structures Comparison

Data Structures	Average Case			Worst Case		
	Search	Insert	Delete	Search	Insert	Delete
Array	O(n)	N/A	N/A	O(n)	N/A	N/A
Sorted Array	O(log n)	O(n)	O(n)	O(log n)	O(n)	O(n)
Linked List	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
Doubly Linked List	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
Stack	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
Hash table	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)
Binary Search Tree	O(log n)	O(log n)	O(log n)	O(n)	O(n)	O(n)
B-Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
Red-Black tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
AVL Tree	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)

출처 : https://cooervo.github.io/Algorithms-DataStructures-BigONotation/index.html


Level 1

문제 1, 약수의 개수와 덧셈

약수의 개수와 덧셈

문제 설명

두 정수 `left` 와 `right` 가 매개변수로 주어집니다. `left` 부터 `right` 까지의 모든 수들 중에서, 약수의 개수가 짝수인 수는 더하고, 약수의 개수가 홀수인 수는 뺀 수를 return 하도록 solution 함수를 완성해주세요.

제한사항

- 1 ≤ `left` ≤ `right` ≤ 1,000

입출력 예

left	right	result
13	17	43
24	27	52

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/77884?language=python3

문제 2, 직사각형 별 찍기

직사각형 별찍기

문제 설명

이 문제에는 표준 입력으로 두 개의 정수 n과 m이 주어집니다.
별(*) 문자를 이용해 가로의 길이가 n, 세로의 길이가 m인 직사각형 형태를 출력해보세요.

제한 조건

- n과 m은 각각 1000 이하인 자연수입니다.

예시

입력

```
5 3
```

출력

```
*****
*****
*****
```

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12969

문제 3, 복리

복리 이자율 7%로 1000만원 저금시 2000만원이 되기까지 몇 년이 걸리는가?

문제 4, 로또 번호 자동생성기

로또 번호 자동생성기를 작성하시오.

```
***** 로또 번호 자동 생성기*****
------------------------------
게임 수 : 5
[4, 20, 22, 26, 27, 30]
[1, 10, 18, 27, 36, 42]
[19, 23, 24, 32, 37, 40]
[10, 16, 23, 24, 40, 43]
[2, 18, 24, 25, 26, 32]
***** 로또 번호 자동 완료 *****
```

문제 5, 핸드폰 번호 가리기

핸드폰 번호 가리기

문제 설명

프로그래머스 모바일은 개인정보 보호를 위해 고지서를 보낼 때 고객들의 전화번호의 일부를 가립니다.
전화번호가 문자열 phone_number로 주어졌을 때, 전화번호의 뒷 4자리를 제외한 나머지 숫자를 전부 `*` 으로 가린 문자열을 리턴하는 함수, solution을 완성해주세요.

제한 조건

- phone_number는 길이 4 이상, 20이하인 문자열입니다.

입출력 예

phone_number	return
"01033334444"	"*******4444"
"027778888"	"*****8888"

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12948

문제 6, 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 사항

- 두 수는 1이상 1000000이하의 자연수입니다.

입출력 예

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

입출력 예 설명

입출력 예 #1
위의 설명과 같습니다.

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

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12940

문제 7, 수박수박수박수박수?

길이가 n이고, "수박수박수박수...."와 같은 패턴을 유지하는 문자열을 리턴하는 함수, solution을 완성하세요. 예를들어 n이 4이면 "수박수박"을 리턴하고 3이라면 "수박수"를 리턴하면 됩니다.

제한 조건

- n은 길이 10,000이하인 자연수입니다.

입출력 예

n	return
3	"수박수"
4	"수박수박"

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12922

문제 8, 소수 찾기

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
5	3

입출력 예 설명

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

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12921

문제 9, 소수 찾기 / 에라토스테네스의 체

- 대표적인 소수(Prime Number) 판별 알고리즘 / 소수들을 대량으로 빠르고 정확하게 구하는 방법
- 먼저 소수를 판별할 범위만큼 배열을 할당하여, 해당하는 값을 넣어주고, 이후에 하나씩 지워나가는 방법을 이용

1. 배열을 생성하여 초기화한다.
2. 2부터 시작해서 특정 수의 배수에 해당하는 수를 모두 지운다.

   (지울 때 자기자신은 지우지 않고 이미 지워진 수는 건너뛴다.)
3. 2부터 시작하여 남아있는 수를 모두 출력한다.

문제 9, 소수 찾기 / 에라토스테네스의 체

- 에라토스네세스의 체로 1부터 25까지를 판별해보자.
- 이차원 배열을 생성하여 값을 초기화

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

문제 9, 소수 찾기 / 에라토스테네스의 체

- 에라토스네세스의 체로 1부터 25까지를 판별해보자.
- 2부터 시작해서 특정 숫자의 배수에 해당되는 숫자들을 모두 지웁니다.(자기 자신은 지우지 않음)

1	2	3	~~4~~	5
~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15
~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25

문제 9, 소수 찾기 / 에라토스테네스의 체

- 에라토스네세스의 체로 1부터 25까지를 판별해보자.
- 3의 배수를 지움(자기 자신을 지우지 않음)

1	2	3	~~4~~	5
~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25

문제 9, 소수 찾기 / 에라토스테네스의 체

- 에라토스네세스의 체로 1부터 25까지를 판별해보자.
- 4의 배수를 지움(이미 지워져 있으므로, 지우지 않고 건너 뜀.이러한 과정을 반복함)

1	2	3	~~4~~	5
~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25

문제 9, 소수 찾기 / 에라토스테네스의 체

- 에라토스네세스의 체로 1부터 25까지를 판별해보자.
- 2부터 시작하여 남아 있는 숫자들을 출력함.

1	2	3	~~4~~	5
~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~
~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~

- 출력 : 2,3,7,11,13,17,19,23

문제 9, 소수 찾기 / 에라토스테네스의 체

```python
n = int(input())
a = [True] * (n + 1)
m = int(n**0.5)

for i in range(2, m + 1):
    if a[i] == True:
        for j in range(i + i, n + 1, i):
            a[j] = False

print([i for i in range(2, n + 1) if a[i] == True])
```

```c
#include <stdio.h>

int number = 100000;
int a[100001];

void primeNumberSieve() {
    for(int i = 2; i <= number; i++) {
        a[i] = i;
    }
    for(int i = 2; i <= number; i++) {
        if(a[i] == 0) continue;
        for(int j = i + i; j <= number; j += i) {
            a[j] = 0;
        }
    }
    for(int i = 2; i <= number; i++) {
        if(a[i] != 0) printf("%d ", a[i]);
    }
}

int main(void) {
    primeNumberSieve();
}
```

```java
public class Solution {

        // 예제와 같이 120까지의 소수를 구하기 위해 120 선언.
        static boolean prime[] = new boolean[121];

    public static void main(String[] args) throws Exception{

        // 구하고자 하는 숫자 범위
        int N = 120;

        // 소수는 false
        // 1은 소수가 아니므로 제외
        prime[0] = prime[1] = true;

        for(int i=2; i*i<=N; i++){
                // prime[i]가 소수라면
            if(!prime[i]){
                // prime[j] 소수가 아님 표시
                for(int j=i*i; j<=N; j+=i) prime[j] = true;
            }
        }

        // 소수 출력
        for(int i=1; i<=N;i++){
                if(!prime[i]) System.out.print(i+" ");
        }

    }
}
```

문제 10, 모의고사

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 설명

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.

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/42840

문제 11, 베스킨 라빈스 31 게임

게임규칙

```
게임의 참여자들은 차례를 정해 1부터 31까지의 수를 순차적으로 부른다. 한번에 1~3개까지 수를 연달아 부를 수 있으며, 마지막 31을 부른 사람이 진다.
```

- 컴퓨터가 무조건 먼저 시작하고,1P는 무조건 2번째로 말한다.컴퓨터가 무조건 이기게 만들어라.
- LEVEL1 예상
- 힌트1:4n-2라는 공식을 사용하면 됩니다.
- 힌트2:이 게임은 31을 부르면 지는 게임이니 30을 부르면 이깁니다.

출처 : https://codingdojang.com/scode/700

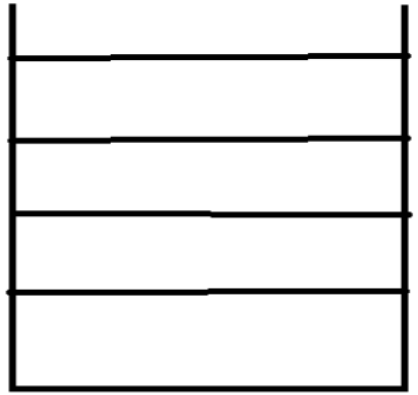
스택(Stack)

- 메모리의 스택 영역은 함수의 호출과 관계되는 지역변수, 매개변수, 리턴 값 등의 임시 데이터를 저장.
- “차곡 차곡 쌓여진 더미”를 의미.
- LIFO(Last in First Out, 후입선출)구조라고 함.

- 가장 먼저 저장되는 데이터는 스택의 아래 쪽(높은 주소)부터 쌓이고, 다음 저장되는 데이터가 바로 그 위(낮은 주소)에 쌓임

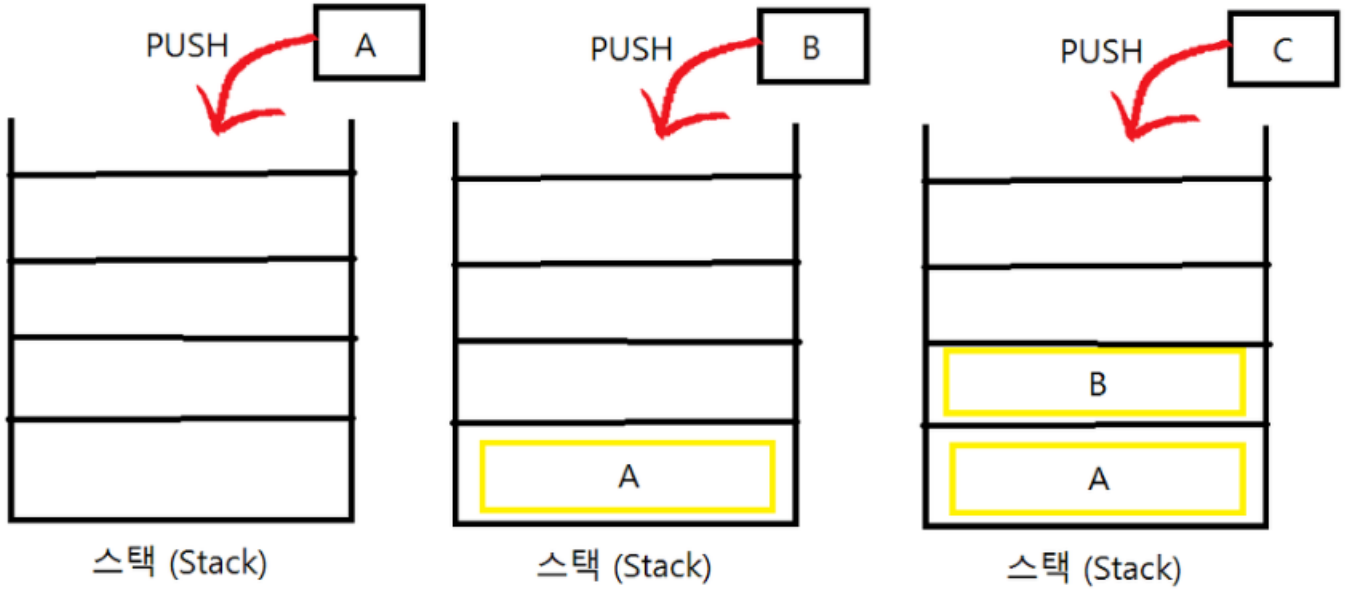
출처 : https://cwjuns.tistory.com/18

스택(Stack)



스택 (Stack)

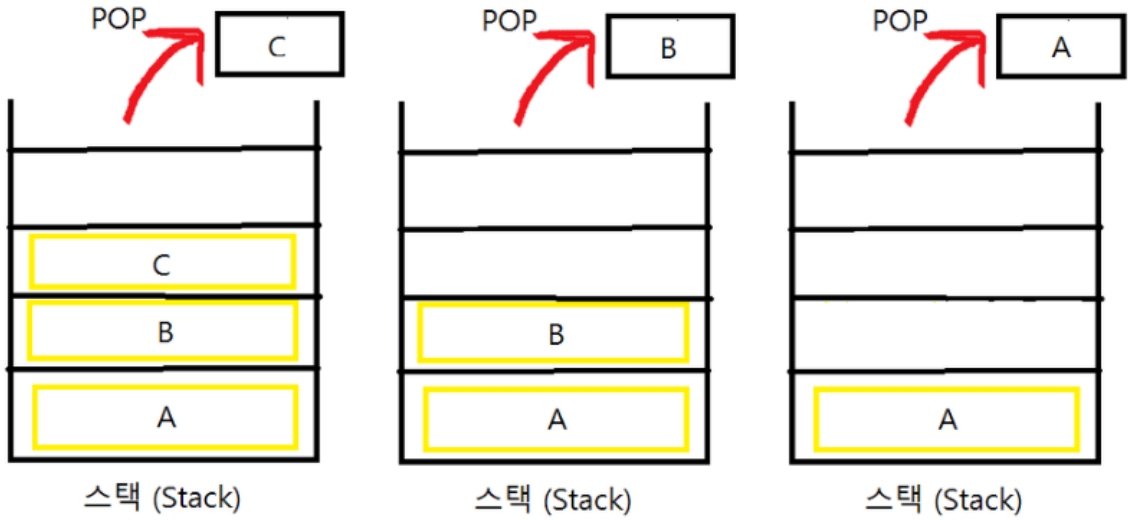
- 이 빈 통을 스택 영역이라고 생각



- 스택에 새로운 데이터를 추가하는 것을 PUSH라고 함. PUSH를 하면 기존 데이터 위에 새 데이터가 순서대로 쌓임.

출처 : https://cwjuns.tistory.com/18

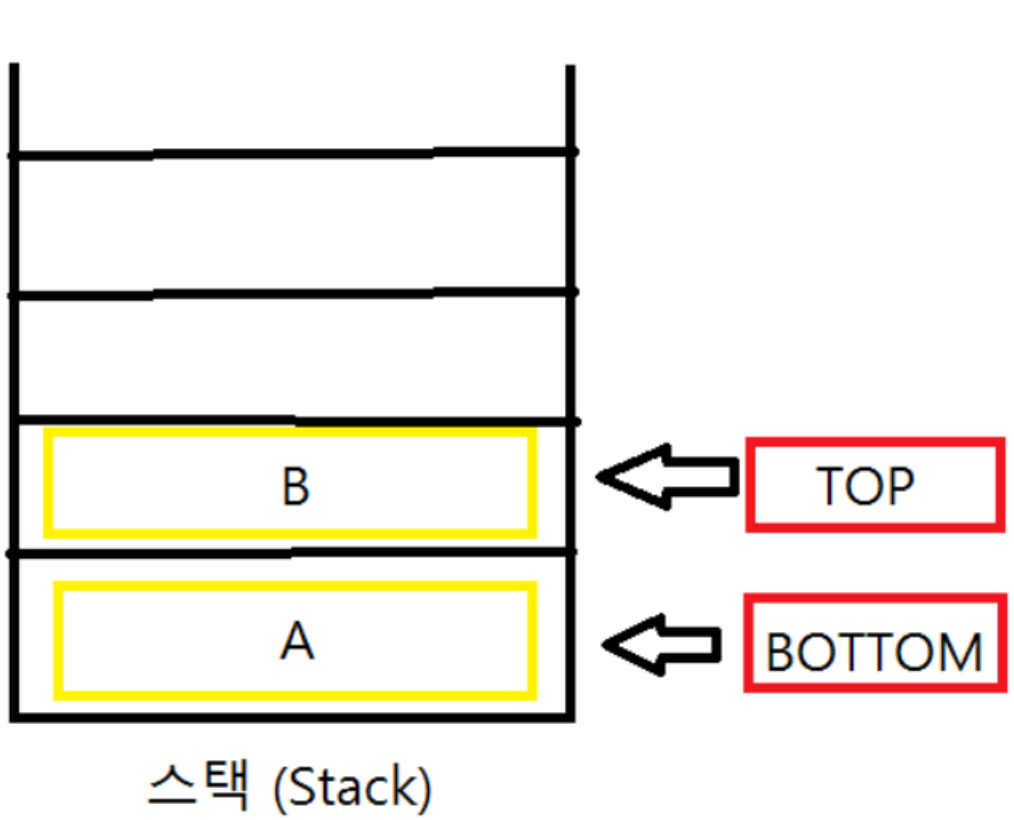
스택(Stack)



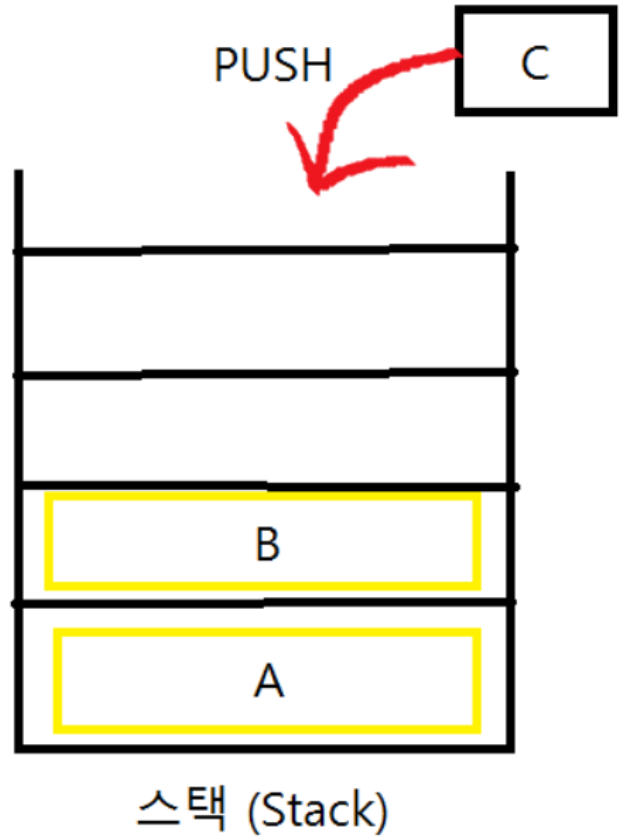
- 스택에 있던 데이터를 다시 빼내는 것을 POP
- PUSH/POP을 통해 데이터의 추가/제거가 가능하며 PUSH/POP 되는 데이터의 크기는 프로그래머가 정할 수 있음.
- 스택이 밑에서부터 데이터를 추가하는 이유는 커널영역을 침범하지 않기 위해서 밑에서부터 데이터를 추가함.

출처 : https://cwjuns.tistory.com/18

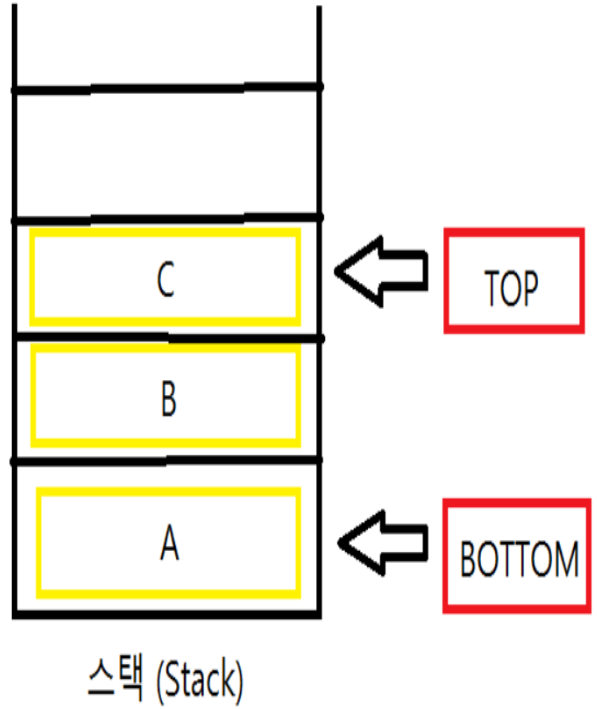
스택(Stack)



스택 (Stack)



스택 (Stack)

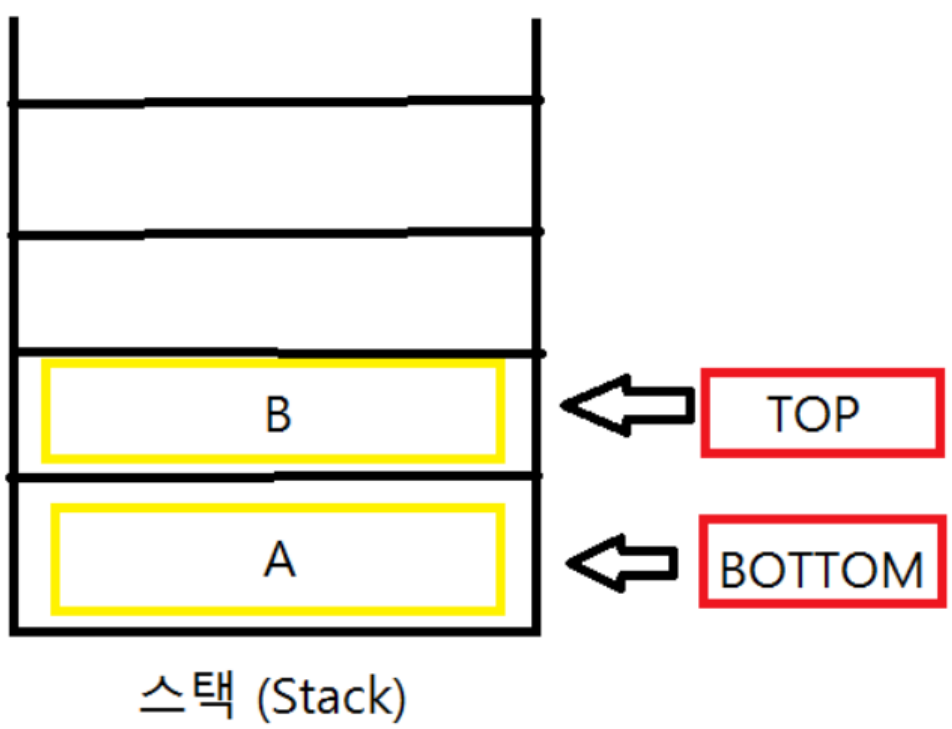


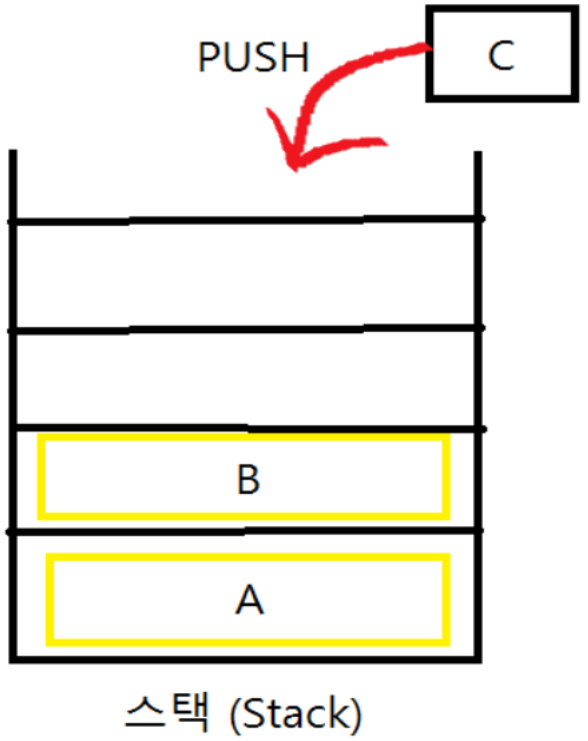
스택 (Stack)

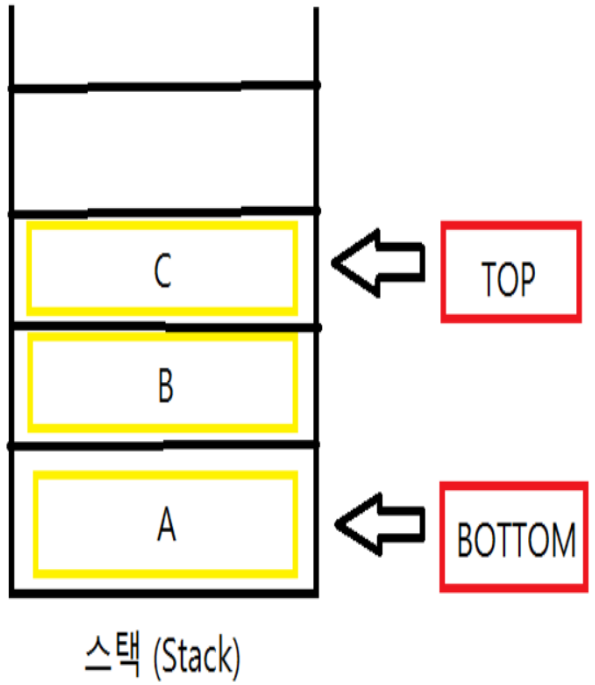
- TOP/BOTTOM은 스택의 특정위치를 가리킴.
- TOP는 가장 최근에 스택에 저장된 값, BOTTOM은 가장 처음 스택에 저장된 값을 가리킴.

출처 : https://cwjuns.tistory.com/18

스택(Stack)





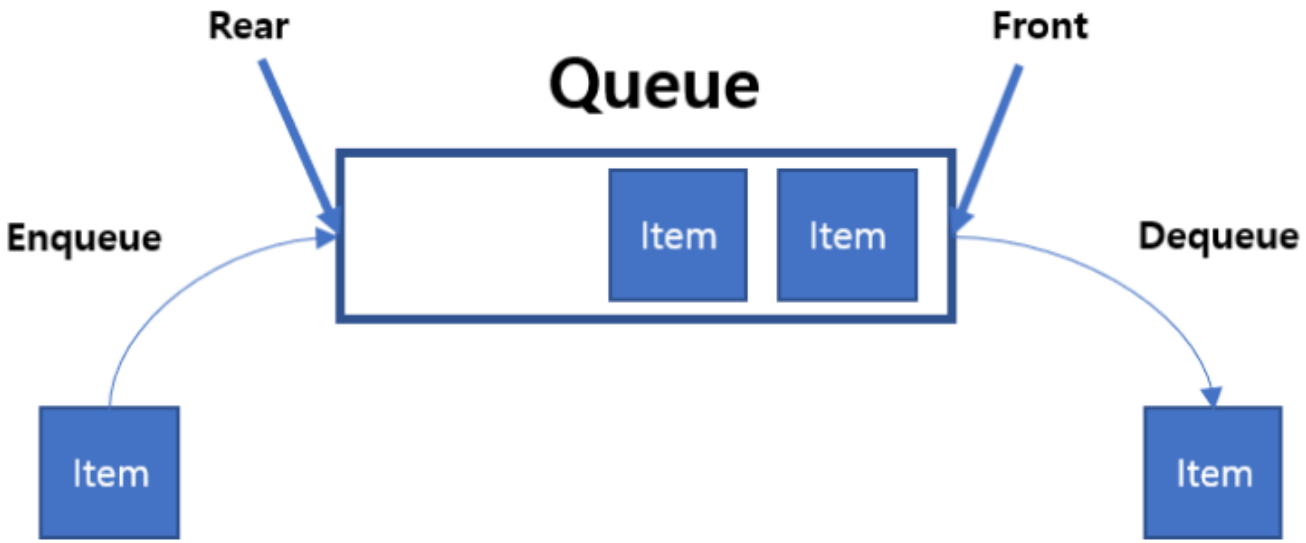


- TOP/BOTTOM은 스택의 특정위치를 가리킴.
- TOP는 가장 최근에 스택에 저장된 값, BOTTOM은 가장 처음 스택에 저장된 값을 가리킴.
- 활용 예시
  - 웹 브라우저 방문기록(뒤로 가기) : 가장 나중에 열린 페이지부터 다시 보여줌.
  - 역순 문자열 만들기 : 가장 나중에 입력된 문자를 출력
  - 수식의 괄호 검사(연산자 우선순위 표현을 위한 괄호 검사)

출처 : https://cwjuns.tistory.com/18

큐(QUEUE)

- 큐는 한쪽 끝(rear)에서는 삽입 연산만 이루어지며 다른 한쪽 끝(Front)에서는 삭제 연산만 이루어지는 유한 순서 리스트이다.



- 큐는 선입 선출(FIFO : First-In First-Out)

출처 : https://galid1.tistory.com/483

문제 12, 같은 숫자는 싫어

배열 arr가 주어집니다. 배열 arr의 각 원소는 숫자 0부터 9까지로 이루어져 있습니다. 이때, 배열 arr에서 연속적으로 나타나는 숫자는 하나만 남기고 전부 제거하려고 합니다. 단, 제거된 후 남은 수들을 반환할 때는 배열 arr의 원소들의 순서를 유지해야 합니다. 예를 들면,

- arr = [1, 1, 3, 3, 0, 1, 1] 이면 [1, 3, 0, 1] 을 return 합니다.
- arr = [4, 4, 4, 3, 3] 이면 [4, 3] 을 return 합니다.

배열 arr에서 연속적으로 나타나는 숫자는 제거하고 남은 수들을 return 하는 solution 함수를 완성해 주세요.

제한사항

- 배열 arr의 크기 : 1,000,000 이하의 자연수
- 배열 arr의 원소의 크기 : 0보다 크거나 같고 9보다 작거나 같은 정수

입출력 예

arr	answer
[1,1,3,3,0,1,1]	[1,3,0,1]
[4,4,4,3,3]	[4,3]

입출력 예 설명

입출력 예 #1,2

문제의 예시와 같습니다.

출처 : https://school.programmers.co.kr/learn/courses/30/lessons/12906

완전 탐색(Exhaustive Search)

- 모든 경우의 수를 시도 하는 방법
- “무식하게 푼다”(brute-force)는 컴퓨터의 빠른 계산 능력을 이용해 가능한 경우의 수를 일일이 나열하면서 답을 찾는 의미.
- 상대적으로 구현이 간단하고, 해가 존재하면 항상 찾게 됨.
- 경우의 수에 따라 실행 시간이 비례하기 때문에 입력 값의 범위가 작은 경우 유용.

단순 Brute-Force

- 단순히 반복문과 조건문으로 모든 경우를 만들어 답을 구하는 방법
- 이 방법만을 사용하는 문제는 거의 나오지 않음.

비트 마스크(BitMask)

- 나올 수 있는 모든 경우의 수가 각각의 원소가 포함되거나, 포함되지 않는지를 0,1로 구분하여 배열에 저장해둘 수 있음.
- Ex) ‘원소가 n개인 집합의 모든 부분 집합 ’ 을 구한다면, 각 원소가 포함되는지 포함되지 않는지를 0,1로 구분하여 배열에 저장해둘 수 있음.

비트 연산자

- 비트(bit) 단위로 논리 연산을 할 때 사용하는 연산자.
- AND 연산(&) : 대응하는 두 비트가 모두 1일 때, 1반환 ex) 1010 & 1111 = 1010
- OR 연산(|) : 대응하는 두 비트가 하나라도 1일 때, 1반환 ex) 1010 | 1111 = 1111
- XOR 연산(^) : 대응하는 두 비트가 서로 다르면, 1반환 ex) 1010^ 1111 = 0101

완전 탐색(Exhaustive Search)

재귀 함수(Recursion function)

- 비트마스크와 마찬가지로 각 원소가 두 가지 선택지를 가질 때 유용하게 사용.
- 포함이 되면 해당 원소를 넣어 함수를 호출하고, 포함되지 않으면 그 상태에서 함수를 호출하는 등의 식.
- 시간 복잡도 O(N)

```python
def function(n):
    if n == 0:
        return
    else:
        function(n-1)
        print(n)

function(3)

# 결과
1
2
3
```

```cpp
int Factorial(int num) {

    if (num == 1) {
        return 1;
    }

    return num * Factorial(num - 1);
}

int main() {
    int num = 5;

    cout << num << "! 값: " << Factorial(num) << endl;

    return 0;
}
```

```java
public class Factorial {

    public static void main(String[] args) {
        int input = 4; // 4!

        System.out.println(fact(input));
    }

    public static int fact(int n) {
        if (n <= 1)
            return n;
        else
            return fact(n-1) * n;
    }
}
```

완전 탐색(Exhaustive Search)

순열(Permutation)

- 서로 다른 N개를 일렬로 나열하는 방법(경우의 수)를 말함
- 순열의 경우의 수는 N!으로 완전 탐색을 이용하기 위해서는 N이 한자리 수는 되어야 함.
- 순열에 원소를 하나씩 채워가는 방식
- 재귀함수 이용 or C++의 next_permutation() 함수 이용.
- 시간복잡도 O(N!)

너비 우선 탐색(Breadth-First Search,BFS)는 하나의 요소를 방문하고 그 요소에 인접한 모든 요소를 우선 방문하는 방식

깊이 우선 탐색(Depth-First Search,DFS)는 트리의 한 요소(노드)와 다음 수준(level)의 지식 노드를 따라가는 탐색하는 방식

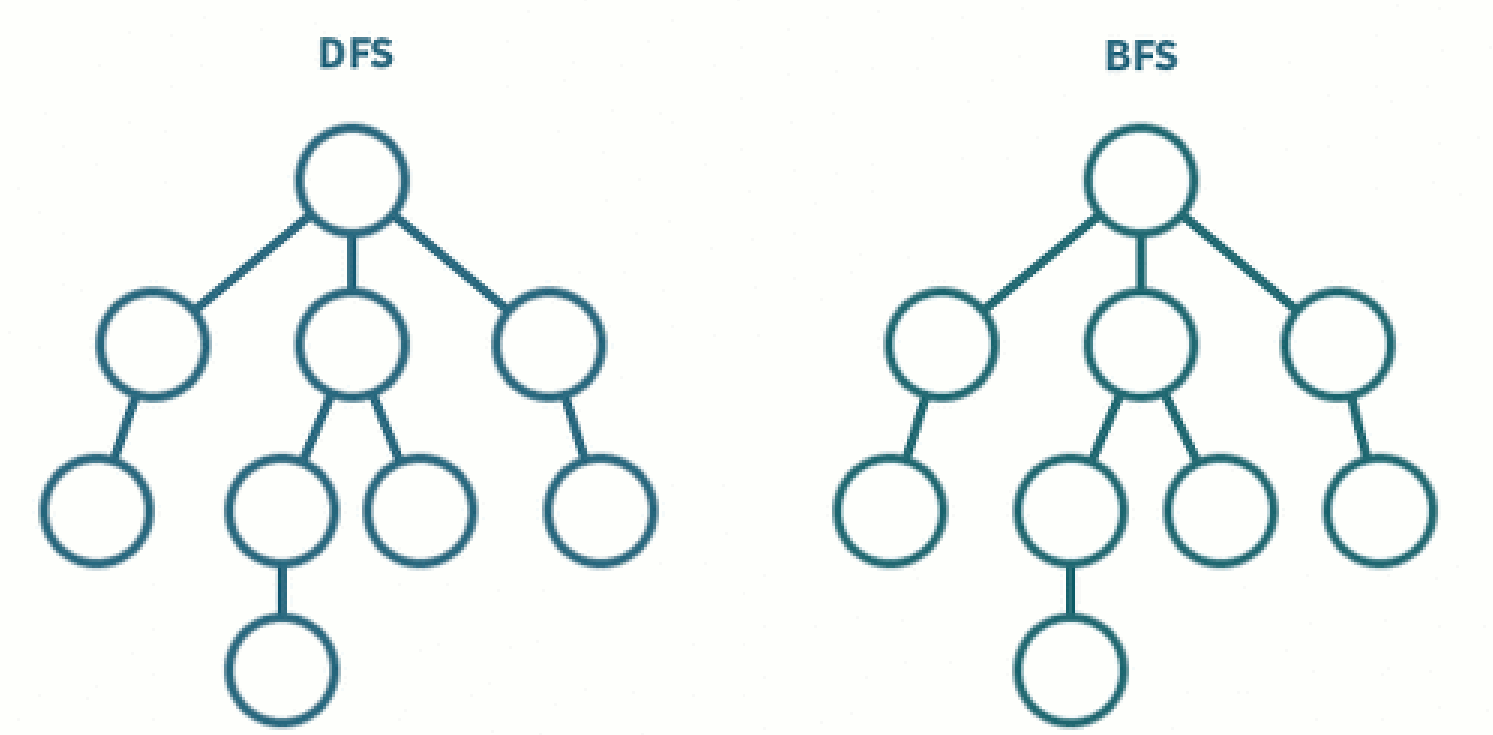
길 찾기 등에 주로 쓰이는 알고리즘

: 단순 길찾기에는 BFS/DFS만 써도 무방하지만, 장애물이 존재하는 등 추가적인 연산이 필요할 때 완전탐색 병용하기도 함.

Ex) 지구 상에 존재하는 모든 친구 관계를 그래프로 표현하고 A와 B 사이에 존재하는 경로를 찾을 때,

- DFS : 모든 친구 관계 다 살펴야 한다.
- BFS : A와 가까운 관계부터 탐색

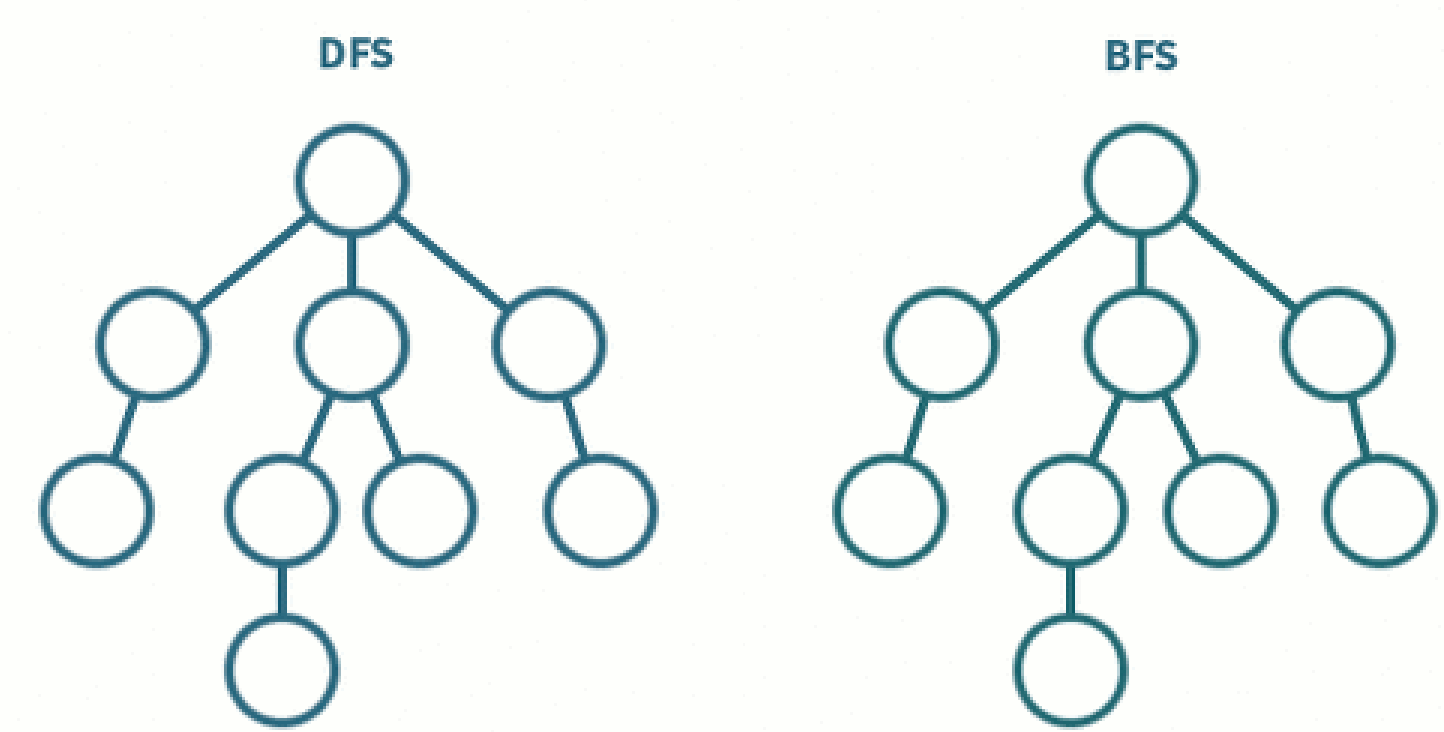
완전 탐색(Exhaustive Search)



깊이 우선 탐색(Depth-First Search,DFS)

- 재귀적으로 동작(재귀, 스택)
- 그래프 탐색의 경우, 어떤 노드를 방문했었는지 여부를 반드시 검사(검사하지 않으면 무한루프)
- 모든 노드 방문하고자 할 때 사용
- BFS보다 간단, BFS 비해서 검색 속도 느림.
- 모든 노드 방문하고자 할 때 사용.

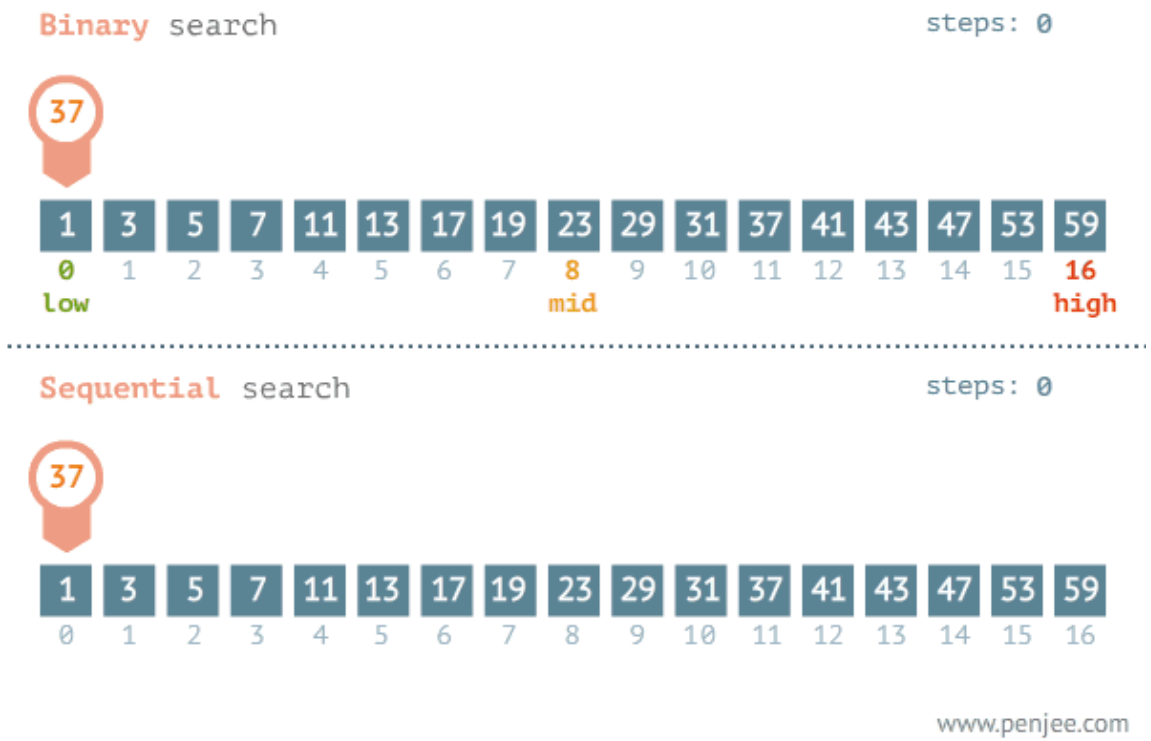
완전 탐색(Exhaustive Search)



너비 우선 탐색(Bread-First Search, BFS)

- 재귀적으로 동작하지 않음.
- 그래프 탐색의 경우, 어떤 노드를 방문 했었는지 여부를 반드시 검사(검사하지 않으면 무한루프)
- BFS는 방문한 노드들을 차례로 저장하고 꺼낼 수 있는 큐 사용(FIFO)
- 넓게 탐색
- 두 노드 사이의 최단 경로 혹은 임의의 경로를 찾고 싶을 때 이 방법을 사용.

이진 탐색 / 이진 탐색(Binary Search)



- 이진 탐색(이분 탐색) 알고리즘은 정렬되어 있는 리스트에서 탐색 범위를 절반씩 좁혀가며 데이터를 탐색하는 방법.
- 배열 내부의 데이터가 정렬되어 있어야만 사용할 수 있는 알고리즘.
- 변수 3개(start, end, mid)를 사용하여 탐색한다. 찾으려는 데이터와 중간점 위치에 있는 데이터를 반복적으로 비교해서 원하는 데이터를 찾는 것이다.

출처 : www.penjee.com


Thank you.
알고리즘 이해 및 구현/ 류영표 강사
ryp1662@gmail.com