

# AlCall - 신바람 알고리즘 스터디

## 1주차 - Part 1

자료구조 / 이분탐색 / 정렬

---

2019. 07. 08.

부산대학교 정보컴퓨터공학부

박상운

high1uck@pusan.ac.kr

Init. BOJ 그룹 가입하기

# BOJ

- <https://www.acmicpc.net/>
- 회원가입 해주세요

# 그룹 들어가기

- 아이디 알려주세요

# 그룹 들어가기

내가 속한 그룹

그룹 만들기

전체 보기

그룹 이름	그룹장	멤버
부산대학교 동아리 연합 알고리즘 세미나	Rche	2
리듬게임열심히하는사람들의모임	Rche	2
부산대학교 알골(ALCALL)	algoshipda	65

# 연습

## 부산대학교 동아리 연합 알고리즘 세미나

부산대학교 동아리 연합 알고리즘 세미나 그룹입니다.

[메인](#)[멤버](#)[문제집](#)[만들기](#)[채점 현황](#)[연습](#)[연습 만들기](#)[랭킹](#)[게시판](#)[글쓰기](#)[파일](#)[관리](#)

연습 이름	우승	준우승	시작	종료	상태	수정
1주차 - 자료구조 / 이분탐색 / 정렬			2019년 7월 8일 19시 00분	2019년 7월 21일 23시 59분	시작까지 3시간 2분 56초	<a href="#">수정</a>

## 연습

## 멤버

## 만들기

연습

## 랭킹

글쓰기

## 그룹 나가기

관리

A - 아시아 정보올림피아드
B - 올림픽
C - 중복 빼고 정렬하기
D - 듣보잡
E - 빈도 정렬
F - 암기왕
G - 베스트셀러
H - 유학 금지
I - 명령 프롬프트
J - 이상한 곱셈
K - 싱기한 네자리 숫자
L - 숫자의 개수
M - 돌전 게임

순위	아이디	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	
1	jcdgods	1 / 1674	2 / 1712	1 / 1695	1 / 1698	2 / 1738	6 / 1831	1 / 1798	1 / 1801	1 / 1805	1 / 1808	1 / 1814	1 / 1815	1 / 1822	4 / 1919	1 / 1864	1 / 1872	1 / 1875	5 / 2017	18 / 32558
2	robustflame	1 / 1459	1 / 1473	1 / 1477	1 / 1489	2 / 1649	1 / 1506	1 / 1518	1 / 1523	1 / 1528	1 / 1533	1 / 1551	1 / 1553	1 / 1564	2 / 1986	1 / 1575	1 / 1583	1 / 1590	0 / --	17 / 26557
3	jak5022	1 / 1528	3 / 1677	1 / 1585	1 / 1458	2 / 2895	2 / 1644	2 / 1495	1 / 1616	1 / 1479	2 / 1665	0 / --	1 / 2851	1 / 1508	0 / --	1 / 1511	1 / 2865	2 / 1515	4 / 2893	16 / 30185
4	Rche	1 / 1469	2 / 1504	1 / 1489	1 / 1515	5 / 1665	2 / 1620	1 / 1611	2 / 1642	1 / 1618	1 / 1628	2 / 1656	1 / 1640	1 / 1864	0 / --	0 / --	0 / --	0 / --	0 / --	13 / 20921
5	wjin0526	1 / 1467	3 / 1542	1 / 1510	2 / 1561	2 / 1597	2 / 1610	2 / 1870	1 / 1868	1 / 1874	3 / 1941	1 / 1908	1 / 1915	1 / 1936	1 / --	0 / --	0 / --	0 / --	0 / --	13 / 22599
6	kium100	0 / --	0 / --	1 / 1484	0 / --	3 / 1902	1 / 1493	2 / 1841	1 / 1516	0 / --	1 / 1888	1 / 1549	1 / 1557	1 / 1592	4 / 1967	1 / 1696	0 / --	1 / 1661	0 / --	12 / 20146
7	seong2035	2 / 1515	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	1 / 2630	0 / --	0 / --	0 / --	0 / --	2 / 4145
8	algoshipda	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	0 / --	1 / 4859	1 / 4859

# 문제

아시아 정보올림피아드

성공

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	1026	715	635	72.323%

## 문제

최근 아시아 지역의 학생들만 참여하는 정보 올림피아드 대회가 만들어졌다. 이 대회는 온라인으로 치러지기 때문에 각 나라에서 이 대회에 참여하는 학생 수의 제한은 없다.

참여한 학생들의 성적순서대로 세 명에게만 금, 은, 동메달을 수여한다. 단, 동점자는 없다고 가정한다. 그리고 나라별 메달 수는 최대 두 개다.

예를 들어, 대회 결과가 다음의 표와 같이 주어졌다고 하자.

참가국	학생번호	점수
1	1	230
1	2	210
1	3	205
2	1	100
2	2	150
3	1	175
3	2	190
3	3	180
3	4	195



# 제출

2535번

제출

맞은 사람

숏코딩

풀이

풀이 작성

풀이 요청

재채점/수정

채점 현황

내 소스

강의▼

질문 검색

## 아시아 정보올림피아드

언어

C++17

소스 코드 공개

☒ 공개

☐ 비공개

☐ 맞았을 때만 공개

소스 코드

1

제출

# 결과 확인

2535번

제출

맞은 사람

숫코딩

풀이

풀이 작성

풀이 요청

재채점/수정

채점 현황

내 소스

강의 ▾

질문 검색

2019년 7-8월 알고리즘 강의

×

2535

Rche

모든 언어 ▾

모든 결과 ▾

검색

채점 번호	아이디	문제 번호	결과	메모리	시간	언어	코드 길이	제출한 시간
13817286	Rche	2535	맞았습니다!!	1120 KB	0 ms	C++ / 수정	675 B	1분 전
4981838	Rche	2535	맞았습니다!!	1120 KB	0 ms	C++ / 수정	675 B	2년 전

대기 중: 5개, 채점 준비 중: 1개, 채점 중: 3개, 채점 서버: 6개, 평균 채점 시간: 7.18초

첫 페이지

# 1. 기본 자료구조

# 시간복잡도

- 알고리즘은 거의 무조건 빠른 게 답이다
- 프로그램 수행시 걸리는 시간의 단위를 표기한다
- $O(1)$  : 입력의 크기와 상관없이 실행시간이 일정 (ex: 구구단 출력하기)
- $O(N)$  : 입력의 크기에 정비례하여 실행시간이 결정 (ex: 배열 순회)
- $O(N^2)$  : 입력의 크기의 제곱에 비례 (ex: 선택 정렬, 이중 for문)
- $O(\lg N)$  : 입력의 로그값에 비례 (ex: 이진 탐색)
- $O(2^N)$  : 입력의 2의 거듭제곱에 비례 (ex: 부분집합 구하기)

# 부분합

- 배열 A에서 A[i]~A[j]의 Sum을 구하기
- 기본적인 생각: for문 순회

```
sum = 0;  
for(int k=i; k<=j; k++) {  
    sum += A[k];  
}  
return sum;
```

→O(N)의 시간이 걸린다.

→실험이나 과제 할 때 이렇게 짜도 여태까지 별 문제는 없었다.

# 부분합

- 질문이 여러 번 들어온다면?
  - index 1 ~ index 100 : 0.1ms
  - index 2000 ~ index 30000 : 1ms
  - index 1 ~ index 99999999 : 1s
  - ...
- 합쳐보면 시간이 많이 걸린다

# 부분합

- $S[i] : A[0] \sim A[i]$ 까지의 합을 저장해둔 배열

A	0	1	2	3	4	5	6	7	8
	1	2	3	4	5	6	7	8	9

S	0	1	2	3	4	5	6	7	8
	1	3	6	10	15	21	28	36	45

→ 특정 연속구간의 합을  $O(1)$ 에 구할 수 있다.

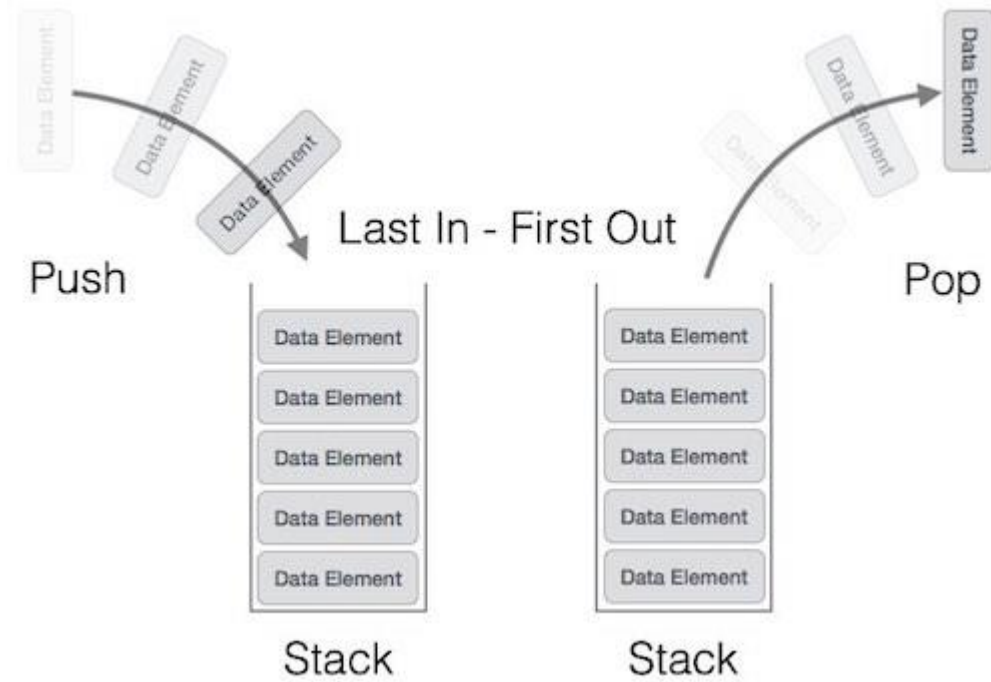
ex:  $\text{Sum}(A[2] \sim A[7]) = S[7] - S[1]$

# 부분합 구현

```
S[0] = A[0];  
for(int i=1; i<N; i++) {  
    S[i] = S[i-1] + A[i];  
}  
  
// Sum from A[i] to A[j]  
cout << S[j]-S[i-1];
```



# 스택



# 스택

- Push
  - 데이터를 위로 집어넣는 연산
- Pop
  - 맨 위에 위치한 데이터를 빼는 연산
- Top
  - 스택 맨 위에 위치한 데이터를 가리키는 연산
- isEmpty
  - 스택이 비었는지 차있는지를 반환하는 연산

# 스택

- Overflow
  - 스택이 꽉 차있는데 Push가 실행될 경우
- Underflow
  - 스택이 비어있는데 Pop이나 Top이 실행될 경우

# 스택 구현

```
int _size = 0;
const int LIMIT = 10000;
int my_stack[LIMIT];

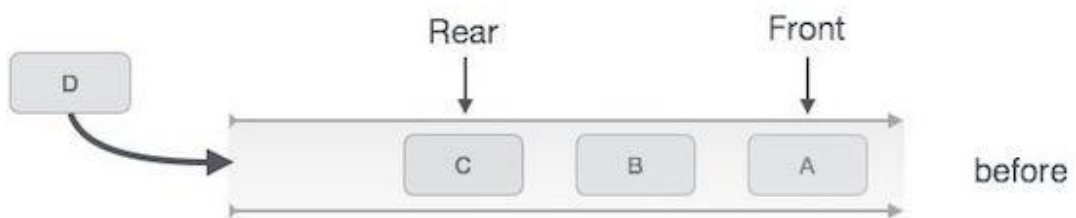
void push(int value) {
    if(_size == LIMIT) {
        cout << "Overflow"; return;
    }
    my_stack[_size] = value;
    _size += 1;
}

void pop() {
    if(_size == 0) {
        cout << "Underflow"; return;
    }
    my_stack[_size] = 0;
    _size -= 1;
}
```

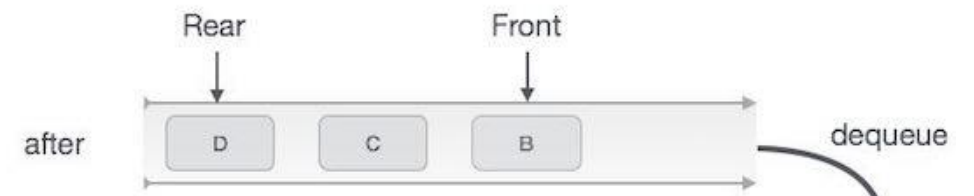
# 스택 구현

```
int top() {  
    if(_size == 0) {  
        cout << "Underflow";  
        return -1;  
    }  
    return my_stack[_size-1];  
}  
  
bool is_empty() {  
    return _size == 0;  
}
```

三



Queue Enqueue



Queue Dequeue

# 큐

- Push
  - 데이터를 맨 앞으로 집어넣는 연산
- Pop
  - 맨 뒤에 위치한 데이터를 빼는 연산
- Front
  - 큐 맨 앞에 위치한 데이터를 가리키는 연산
- End
  - 큐 맨 뒤에 위치한 데이터를 가리키는 연산
- isEmpty
  - 큐가 비었는지 차있는지를 반환하는 연산

# 큐 구현

```
int _end = 0;
int _front = 0;
const int LIMIT = 10000;
int my_queue[LIMIT];
void push(int value) {
    if(_end == LIMIT) {
        cout << "Overflow";
        return;
    }
    my_queue[_end] = value;
    _end += 1;
}
void pop() {
    if(_front == end) {
        cout << "Underflow"; return;
    }
    my_queue[_front] = 0;
    _front += 1;
}
```



# 큐 구현

```
int front() {
    if(_front == _end) {
        cout << "Underflow";
        return -1;
    }
    return my_queue[_front];
}

int end() {
    if(_front == _end) {
        cout << "Underflow"; return -1;
    }
    return my_queue[_end-1];
}

bool is_empty() {
    return _front == _end;
}
```

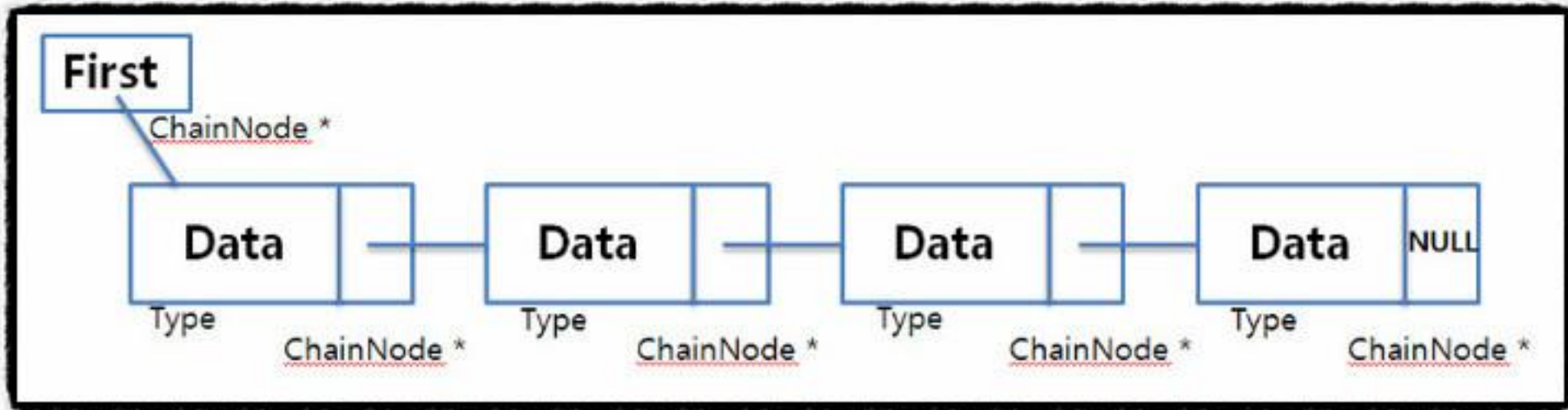
# 큐 구현

```
int front() {
    if(_front == _end) {
        cout << "Underflow";
        return -1;
    }
    return my_queue[_front];
}

int end() {
    if(_front == _end) {
        cout << "Underflow"; return -1;
    }
    return my_queue[_end-1];
}

bool is_empty() {
    return _front == _end;
}
```

# 연결 리스트



# 연결 리스트 구현

<https://thrillfighter.tistory.com/149>

Extra. Q&A