2022 3Q~4Q

KUGODS Algorithm Study

week 1. 자료구조와 STL (1)

Study Introduction

- 허준환 (junhwan26@korea.ac.kr)
- TA: 유지연 (ji04230423@gmail.com)
- 매주 목 19시~20시 (C++)
- 시험기간 휴강 (시험기간이 다가오면 협의)
- 과제 (백준)
  - 매주 과제 6문제, 미리 푸는 건 자유
  - 기초 문제 4~5문제, 기출 문제(초, 중등 올림피아드) 1~2 문제로 구성
  - (월, 화, 수, 금, 토, 일 검사) 벌금 2,000원

Study Introduction

- 전주 과제의 올림피아드 문제(6, 7번) 코드 리뷰 or 발표 (10분 ~ 20분)
  - 6번 발표 시 벌금 면제권 2장, 7번 발표시 다음 주 과제 면제 (선택 불가)
  - 6번 브실 수준의 기출
  - 7번 골플 수준의 기출 (배운 범위 내 or 구현)
- 알고리즘 설명 (30분 ~ 40분)
- 이번주 과제의 기출 문제 설명 (10분 내외)

Study Goal

- Problem Solve (PS) 주어진 조건에 적합한 알고리즘을 생각하고 구현하는 능력
- 취업 준비 코딩테스트?
- Medium : 한국정보올림피아드 **"초등부"** 본선 수준의 알고리즘 능력
- Maximum : 한국정보올림피아드 **"중등부"** 수준의 알고리즘 능력
- 자료구조 & 알고리즘 수업 이상의 구현/적용 능력
- 자구/알고에 좋은 학점을 받았다 -> PS를 잘한다? (Maybe No?)
- PS를 잘 한다? -> 자구/알고에 좋은 학점을 받는다? (YES!!)

ICE Breaking

https://www.menti.com/al2uhpmvz992

ICE Breaking

https://www.menti.com/alebgj28q4nm

C++ & STL

```c
#include<stdio.h>

int main() {
    int a;
    scanf("%d",&a);
    printf("%d",a);
}
```

⇨

```c
#include<iostream>

int main() {
    int a, b;
    std::cin >> a >> b;
    std::cout << a;
}
```

⇨

```c
#include<iostream>
using namespace std;
int main() {
    int a;
    cin >> a;
    cout << a;
}
```

STL : Standard Template Library

프로그래머가 자료구조나 알고리즘을 직접 구현하지 않고도 쓸 수 있게 해주는 라이브러리

- Container (컨테이너) : 미리 구현된 자료구조
- Iterator (반복자) : C에서의 Pointer 역할
- Algorithm (알고리즘) : 미리 구현된 알고리즘

자료구조 Vector with STL

```cpp
#include <iostream>
#include <vector>
using namespace std;
vector<int> v;

int main() {
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        int tmp;
        cin >> tmp;
        v.push_back(tmp);
    }
    for (int i = 0; i < N; i++)
        cout << v[i] << " ";
}
```

자주 쓰는 함수

v.assign(n, x) : v에 n개의 x 할당

v.push_back(x) : v의 마지막에 x 삽입

v.pop_back() : v의 마지막 원소 삭제

v.front() : v의 맨 처음 원소 참조

v.back() : v의 마지막 원소 참조

v.clear() : 모든 원소를 제거(메모리는 유지)

v.size() : 벡터의 원소 개수 반환

v.empty() : v가 비었는지 반환

v.begin() : 첫번째 원소를 가리키는 iterator 반환

v.end() : 마지막의 다음 iterator 반환

v.insert(it) : 중간에 원소 삽입

자료구조 Stack with STL

```cpp
#include <iostream>
#include <stack>
using namespace std;
stack<int> st;

int main() {
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        int tmp;
        cin >> tmp;
        st.push(tmp);
    }
    while(!st.empty()){
        cout << st.top() << " ";
        st.pop();
    }
}
```

자주 쓰는 함수

st.push(x) : st의 top에 x 삽입

st.pop() : st의 top 삭제

st.top() : st의 top 참조

st.empty() : st가 비었는지 반환

st.size() : st의 크기 반환

자료구조 Queue with STL

```cpp
#include <iostream>
#include <queue>
using namespace std;
queue<int> q;

int main() {
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        int tmp;
        cin >> tmp;
        q.push(tmp);
    }
    while(!q.empty()){
        cout << q.front() << " ";
        q.pop();
    }
}
```

자주 쓰는 함수

`q.push(x)` : q의 맨 뒤에 x 삽입

`q.pop()` : q의 front 삭제

`q.front()` : q의 front 참조

`q.empty()` : q가 비었는지 반환

`q.size()` : q의 크기 반환

자료구조 Deque with STL

```cpp
#include <iostream>
#include <deque>
using namespace std;
deque<int> pq;

int main() {
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        int tmp;
        cin >> tmp;
        dq.push_back(tmp);
    }
    while(!dq.empty()){
        cout << dq.front() << " ";
        dq.pop();
    }
}
```

자주 쓰는 함수

dq.push_front(x) : dq의 front에 x 삽입

dq.push_back(x) : dq의 back에 x 삽입

dq.pop_front() : dq의 front 삭제

dq.pop_back() : dq의 back 삭제

dq.empty() : dq가 비었는지 반환

자료구조 Priority Queue with STL

```cpp
#include <iostream>
#include <queue>
using namespace std;
priority_queue<int> pq; //maxheap (내림차순)
priority_queue<int,vector<int>,greater<int> > pq;
int main() {
    int N;
    cin >> N;
    for (int i = 0; i < N; i++){
        int tmp;
        cin >> tmp;
        pq.push_back(tmp);
    }
    while(!pq.empty()){
        cout << pq.front() << " ";
        pq.pop();
    }
}
```

자주 쓰는 함수

`pq.push(x)` : pq에 x 삽입

`pq.pop()` : pq의 front 삭제

`pq.top()` : pq의 front 참조

`pq.empty()` : pq가 비었는지 반환

`pq.size()` : pq의 크기 반환

비교함수를 직접 구현해서 사용하기도함

자료구조 String with STL

```cpp
#include<iostream>
#include<string>
using namespace std;

int main() {
    string s;
    cin >> s;
    s + "a";
    cout << s;
}
```

생성자	
`string s`	기본 생성자로 s 생성
`string s(str)`	str 문자열로 s 생성
`string s(str, n);`	str 문자열에서 n개의 문자로 s를 생성
`string s(n, c);`	n개의 c문자로 s를 생성
`string s(iter1, iter2);`	반복자(포인터) 구간 [iter1, iter2)의 문자로 s를 생성