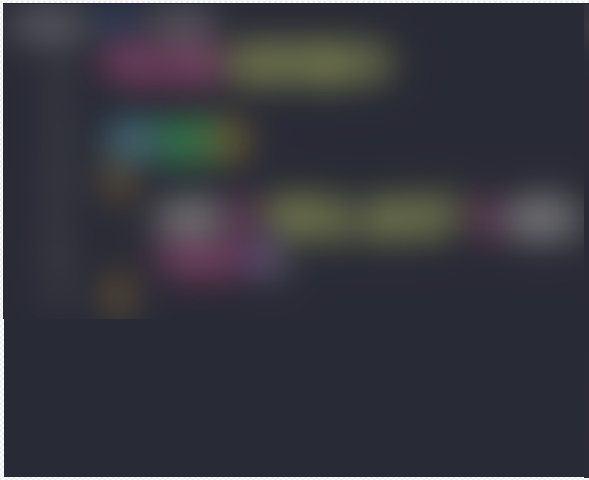
알고리즘 스터디

1주차: 시간/공간 복잡도, 기본 자료구조

시간복잡도: 그전에,, 아주 간단한 알고리즘 문제

" Hello, world!” 를 **한번** 출력하라



" Hello, world!” 를 **입력받은 횟수만큼** 출력하라

시간복잡도: 그전에,, 아주 간단한 알고리즘 문제

" Hello, world!" 를 **한번** 출력하라

study > C++ 1.cpp

```cpp
#include <iostream.h>

int main()
{
    cout << "Hello, world!" << endl;
    return 0;
}
```

" Hello, world!" 를 **입력받은 횟수만큼** 출력하라

study > C++ 2.cpp

```cpp
#include <iostream.h>

int main()
{
    int iteration_count{};
    cin >> iteration_count;

    for (int i = 0; i < iteration_count; ++i)
    {
        cout << "Hello, world!" << endl;
    }
}
```

시간복잡도: 그전에,, 아주 간단한 알고리즘 문제

" Hello, world!” 를 입력받은 **횟수의 제곱 만큼** 출력하라

" Hello, world!” 를 입력받은 **횟수만큼** 출력하라

시간복잡도: 그전에,, 아주 간단한 알고리즘 문제

" Hello, world!” 를 입력받은 **횟수의 제곱 만큼** 출력하라

study > C++ 3.cpp

```cpp
#include <iostream.h>

int main()
{
    int iteration_count{};
    cin >> iteration_count;

    for (int i = 0; i < iteration_count * iteration_count; ++i)
    {
        cout << "Hello, world!" << endl;
    }
}
```

" Hello, world!” 를 입력받은 **횟수만큼** 출력하라

study > C++ 2.cpp

```cpp
#include <iostream.h>

int main()
{
    int iteration_count{};
    cin >> iteration_count;

    for (int i = 0; i < iteration_count; ++i)
    {
        cout << "Hello, world!" << endl;
    }
}
```

시간 복잡도: 정의

- **"입력된 데이터의 크기에 따라 얼마나 더 느려지는가"**를 나타내는 척도

- O(polynomial of N)로 표현함
  - 입력데이터에 상관없이 일정한 시간 : $O(1)$
  - 입력데이터에 정비례 : $O(N)$
  - 입력데이터의 크기의 로그에 비례 : $O(\log N)$

- 같은 문제를 풀더라도 소요되는 시간은 달라질 수 있다
  - 문제의 시간제약조건 내에 풀릴 수 있는 시간복잡도를 갖는 알고리즘 선택
  - 그중 가장 빠르게 구현할 수 있는 것을 선택하자(웬만하면)

공간 복잡도: 정의

- **"~ 얼마나 더 많은 공간(메모리)을 차지하는가"**를 나타내는 척도

- O(polynomial of N)로 표현함
  - 입력데이터에 상관없이 일정한 공간 : $O(1)$
  - 입력데이터에 정비례 : $O(N)$
  - 입력데이터의 크기의 로그에 비례 : $O(\log N)$

- 같은 문제를 풀더라도 소요되는 시간은 달라질 수 있다
  - 문제의 시간제약조건 내에 풀릴 수 있는 시간복잡도를 갖는 알고리즘 선택
  - 그중 가장 빠르게 구현할 수 있는 것을 선택하자(웬만하면)

시간/공간 복잡도

- **“어떤 방법으로 문제를 풀어야 할까”**에 대한 답을 얻을 수 있겠다
- 일반적으로 컴퓨터는 1초에 100,000,000회 연산 가능하다고 가정

- 길이가 2^10 ≈ 1,000 인 오름차순 정렬된 배열에서 원하는 숫자가 있는지 찾는 문제
  - 선형탐색으로 풀 수 있겠다~
- 길이가 2^100 ≈ 10^30 인 오름차순 정렬된 배열에서 원하는 숫자가 있는지 찾는 문제
  - 선형탐색 알고리즘으로 10^22초 ≈ 3*10^14년 걸린다
  - TLE(Time Limit Exceeded)에러와 함께 풀이 실패
  - 다른 방법을... 찾아야 하겠지...?

배열, 링크드 리스트: Array, Linked-List

배열

- 연속적인 메모리 공간상에 저장되는 목록형태의 자료구조
- 인덱스 연산자를 통한 접근으로 Read에 O(1)시간

링크드 리스트(Linked-List)

- 요소가 인접 요소를 가리키는 방식으로 사슬(Chain)처럼 연결되는 자료구조

배열: 조금 더 자세히...

- 연속적인 메모리 공간상에 저장되는 목록형태의 자료구조
- 인덱스 연산자를 통한 접근으로 Read에 O(1)시간
  - `arr[0]` 는 곧 `*(arr + 0)` 로 치환된다
  - []는 연산자에요~
- 하지만 삽입, 삭제등의 작업은 O(n)이 소요된다
  - 중간에 끼워넣거나 삭제하려면 앞/뒤 요소를 복사해야 하니까...
- 언어별 가변배열 선언 방법은...
  - C: **`malloc`** / C++: **`new`** / Java: **`new`** / Python: 애초에 동적이다

링크드 리스트: 조금 더 자세히...

- 요소가 인접 요소를 가리키는 방식으로 사슬(Chain)처럼 연결되는 자료구조
- 하나의 요소는 아래 성질을 가짐
  - 배열처럼 자기 자신의 값을 가짐
  - 포인터를 통해 인접 요소를 가리킨다
- 그로 인해, 삽입과 삭제가 O(1)에 가능함
- 동시에, Random Access가 불가능해짐

연습문제: 4 요세푸스 문제[1]

- 1번부터 N번까지 **N명의 사람이 원을 이루면서 앉아있고**, 양의 정수 K(≤ N)가 주어진다. 이제 **순서대로 K번째 사람을 제거**한다. 한 사람이 제거되면 **남은 사람들로 이루어진 원을 따라 이 과정을 계속**해 나간다. 이 과정은 N명의 **사람이 모두 제거될 때까지 계속**된다. 원에서 사람들이 제거되는 순서를 (N, K)-요세푸스 순열이라고 한다. 예를 들어 (7, 3)-요세푸스 순열은 <3, 6, 2, 7, 5, 1, 4>이다.

- **(N, K)-요세푸스 순열을 구하는 프로그램을 작성하시오.**

[1] https://www.acmicpc.net/problem/1158