시간복잡도
정렬
이분탐색

SAL 1주차

공부법

알고리즘 학습 → 관련된 문제 풀이
백준 단계별, solved.ac 클래스

올림피아드문제, 대학교대회 문제, 랜덤디펜스 (브랜디, 실랜디)
코딩테스트와 대회에서는 알고리즘 분류를 알려주지 않음
→ 알고리즘 분류를 가리고 도전 → 충분히 고민했으면 보기
→ 모르는 알고리즘이 있으면 공부하기
→ 아는 알고리즘인 경우에는 더 고민 후 풀이 보기

```
#include <bits/stdc++.h>
```

https://godog.tistory.com/entry/C-include-bitsstdch-%ED%97%A4%EB%8D%94-%EC%82%AC%EC%9A%A9%ED%95%98%EA%B8%B0

개발에서 사용하면 혼남!

using namespace std;

```cpp
#include <bits/stdc++.h>
int main() {
    int n;
    std::cin >> n;
    std::cout << n << std::endl;
    return 0;
}
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int n;
    cin >> n;
    cout << n << endl;
    return 0;
}
```

개발에서 사용하면 혼남!

```
ios::sync_with_stdio(0);
cin.tie(0);
```

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n;
    cin >> n;
    cout << n << endl;
    return 0;
}
```

std::endl은 속도가 느림

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n;
    cin >> n;
    cout << n << '\n';
    return 0;
}
```

https://www.acmicpc.net/problem/15552

자료형은 항상 주의

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    long long n;
    cin >> n;
    cout << n << '\n';
    return 0;
}
```

https://www.acmicpc.net/problem/11382

typedef long long ll;

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    ll n;
    cin >> n;
    cout << n << '\n';
    return 0;
}
```

시간복잡도

보통 1초에 10억번의 연산 가능
문제를 풀 때는 1초에 최대 1억번의 연산이라고 생각하면 편함

Python `arr.sort()`는 1개의 연산?
하나의 문장을 썼다고 1개의 연산이 아니다!

시간복잡도

보통 문제에는 입력의 크기 n이 주어짐
프로그램의 실행시간을 n을 이용해서 보통 나타냄
최악의 실행시간이 중요함

Big-O Notation으로 나타냄
수학적인 정의는 여기서 다루지 않겠음

가장 중요한 정보만 표시함

시간복잡도 예시

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n, ans = 0; cin >> n;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) ans++;
    }
    cout << ans << '\n';
    return 0;
}
```

시간복잡도 예시

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n, ans = 1; cin >> n;
    for (int i = 1; i * 2 <= n; i++) {
        if (n % i == 0) ans++;
    }
    cout << ans << '\n';
    return 0;
}
```

시간복잡도 예시

더 많은 예시는 정렬, 이분탐색을 통해 알아보자

정렬

주어진 데이터를 순서대로 나열하는 것

꼭 정렬하는 데이터가 수일 필요는 없다!

문자열을 사전순으로 정렬
ABC, ABB, ACB → ABB, ABC, ACB

비교기반정렬

버블정렬, 선택정렬, 삽입정렬: $O(N^2)$
합병정렬, 힙정렬: $O(N\log N)$
퀵정렬: 평균 $O(N\log N)$, 최악 $O(N^2)$

정렬

정렬은 이미 구현 되어있는 함수를 쓰는 게 좋음
C++에서는 `std::sort` → $O(N\log N)$ 보장
비교기반정렬에서는 비교함수가 중요

비교기반정렬은 $O(N\log N)$보다 빠른 것이 불가능함
증명은 생략

std::sort (1)

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n; cin >> n; vector<int> a(n);
    for (auto& x : a) cin >> x;
    sort(a.begin(), a.end());
    for (int i = 0; i < n; i++) {
        cout << a[i] << " \n"[i == n - 1];
    }
    return 0;
}
```

std::sort (2)

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n; cin >> n; vector<int> a(n);
    for (auto& x : a) cin >> x;
    sort(a.rbegin(), a.rend());
    for (int i = 0; i < n; i++) {
        cout << a[i] << " \n"[i == n - 1];
    }
    return 0;
}
```

std::sort (3)

```cpp
#include <bits/stdc++.h>
using namespace std;
bool comp(const int& a, const int& b) {
    return abs(a) < abs(b);
}
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n; cin >> n; vector<int> a(n);
    for (auto& x : a) cin >> x;
    sort(a.begin(), a.end(), comp);
    for (int i = 0; i < n; i++) {
        cout << a[i] << " \n"[i == n - 1];
    }
    return 0;
}
```

https://www.acmicpc.net/problem/1181

비교함수

비교함수 Comp(a, b)가 참이면 a가 b보다 왼쪽에 있다는 뜻
비교함수는 $a < b$이고 $b < c$이면 $a < c$를 만족해야 함

이게 뭔 당연한 소리냐? 그럼 쉬는 동안 재밌는 영상을 봅시다

https://www.youtube.com/watch?v=td7I40BT3p8

비교기반정렬이 아닌 경우는?

https://www.acmicpc.net/problem/10989

PS에서 실수 다루기

실수를 다루는 것은 매우 위험하다!

https://www.acmicpc.net/problem/2417

이분탐색

PS에서 가장 많이 사용되는 방법 중에 하나
업다운 게임에서 사용하는 전략이 대표적인 예시

정렬된 수열이 주어질 때, 특정 값을 찾기
모든 값을 다 확인하면 $O(n)$
이분탐색을 이용하면 $O(\log n)$

이분탐색

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    ios::sync_with_stdio(0); cin.tie(0);
    int n, k; cin >> n >> k; vector<int> a(n);
    for (auto& x : a) cin >> x;
    int lo = 0, hi = n - 1;
    while (lo < hi) {
        int mid = (lo + hi) / 2;
        if (a[mid] < k) lo = mid + 1;
        else hi = mid;
    }
    if (a[lo] == k) cout << "Found!\n";
    else cout << "Not found\n";
    return 0;
}
```

이분탐색

이분탐색에서 가장 중요한 것은
정렬된 데이터에서만 쓸 수 있다는 사실

→ 정렬이 되지 않은 데이터가 주어지면 사용할 수 없음

https://www.acmicpc.net/problem/1920

이분탐색

매우 많은 응용이 가능

https://www.acmicpc.net/problem/1654

맞왜틀?

맞는데 왜 틀려요?

문제를 풀다보면 분명히 맞게 짠 것 같은데 틀렸다고 나오는 경우가 많음 (오버플로우, 예외 처리, 실수 오차 등등)

많이 문제를 풀면서 디버깅 능력을 향상시키는 것이 중요

과제

https://www.acmicpc.net/group/workbook/17293

풀 수 있는 만큼만!
문제는 모두 다 읽어보는 것을 권장
영어로 된 문제도 있음