02. STL & Sort

Div. 3 알고리즘 스터디 / 임지환

Review for last week

- Bubble Sort & Insertion Sort
- Linear Search & Binary Search
- Analyzing Time Complexity

Review for last week

- Bubble Sort & Insertion Sort → Both $O(n^2)$
- Linear Search & Binary Search
- Analyzing Time Complexity

Review for last week

- Bubble Sort & Insertion Sort → Both $O(n^2)$
- Linear Search & Binary Search → $O(n)$ & $O(\log n)$
- Analyzing Time Complexity

Faster Sorting Method

- 주로 쓰는 알고리즘으로 Quick Sort, Merge Sort, Heap Sort,...등등
- 특수한 상황을 제외하고, 정렬을 하는데 드는 복잡도는 $O(n \log n)$이 최선

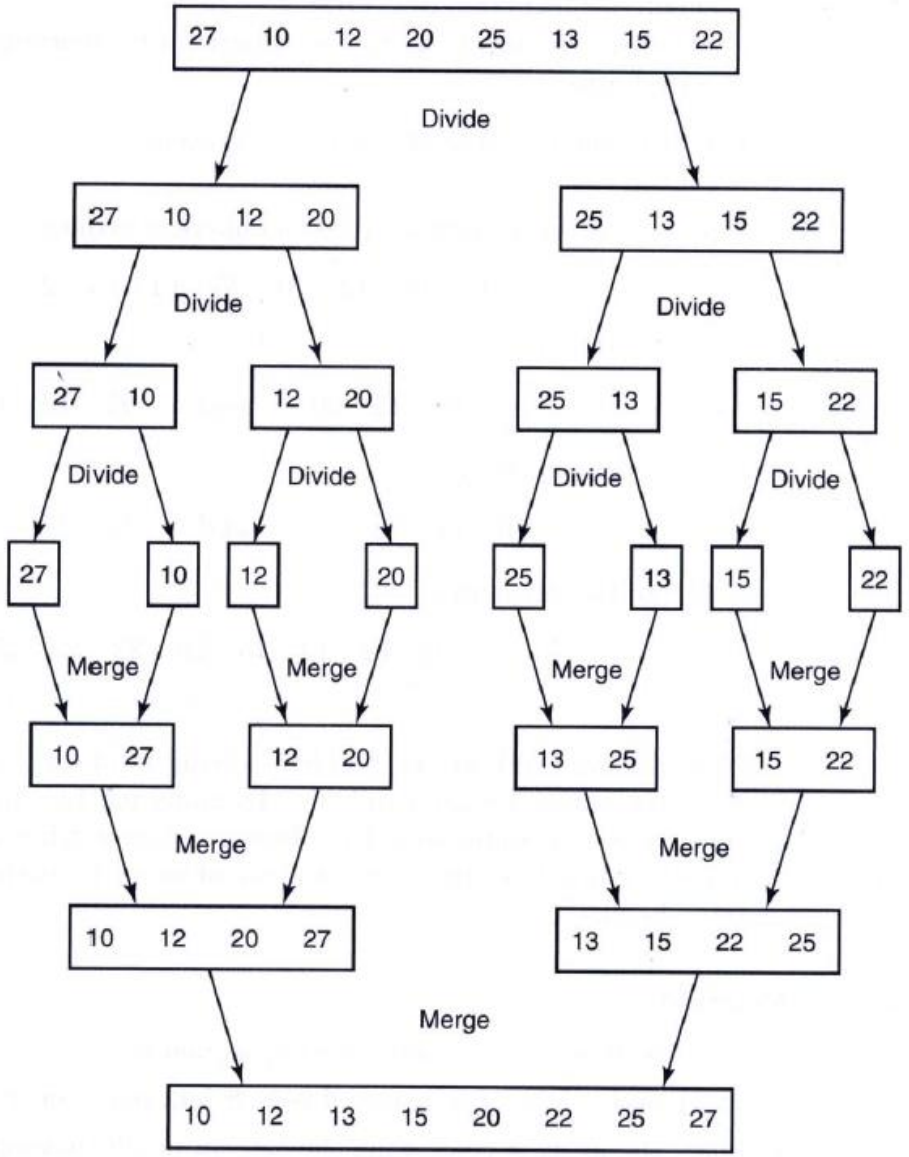
why $O(n \log n)$??

example) Merge Sort

$$T(n) = 2T(\tfrac{n}{2}) + cn$$

$$T(1) = 1$$

$$T(n) = O(n \log n)$$

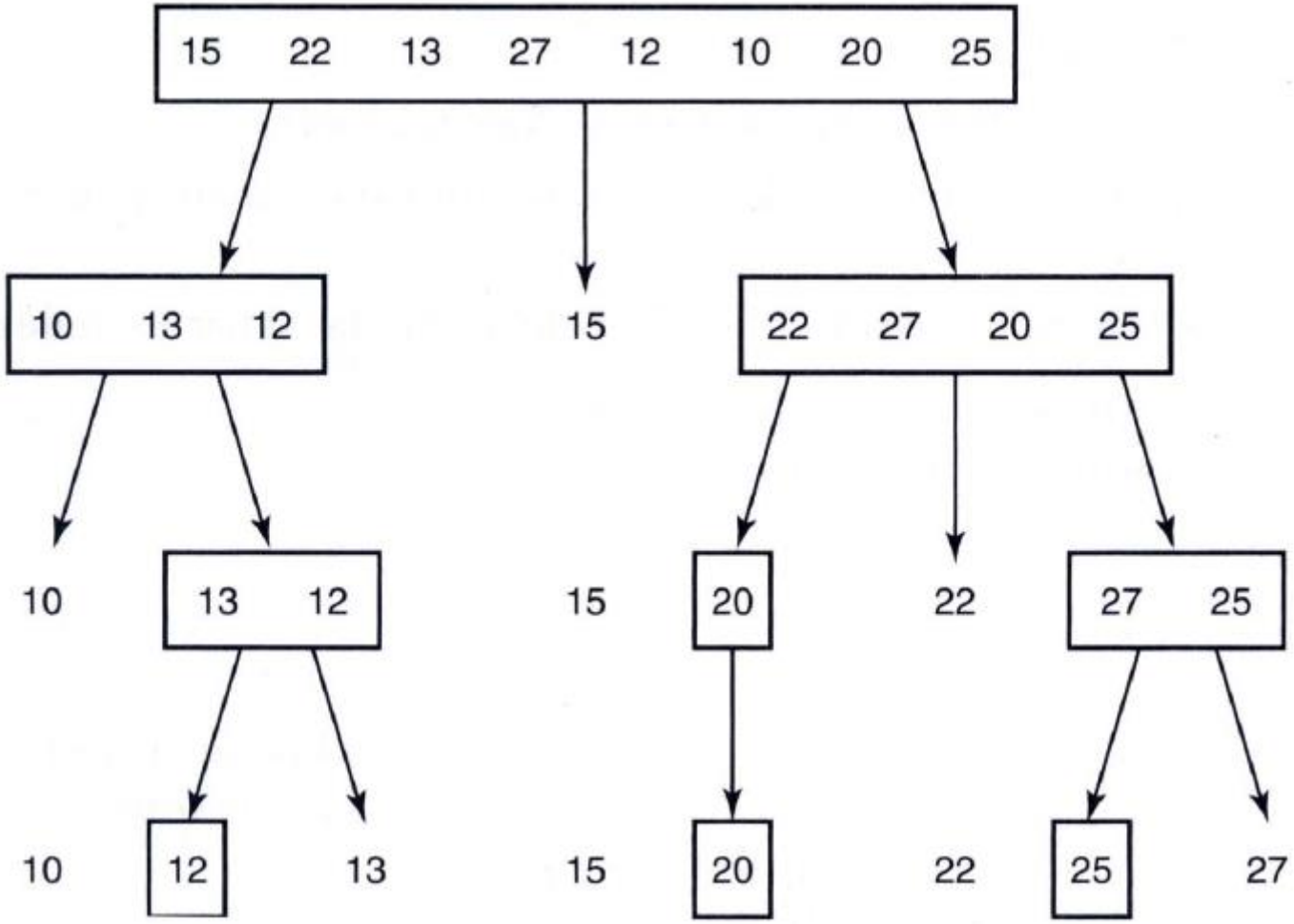


example 2) Quick Sort

$$T(n) = \Sigma_{p=1}^{n} \frac{1}{n}\{T(p-1) + T(n-p)\} + cn$$

$$T(0) = 1$$

$$T(n) = O(n \log n)$$



std :: sort

- 데이터의 크기가 n일 때 $O(n \log n)$으로 자료를 정렬해주는 함수
- Quick Sort 기반
- 단일 type 배열, 구조체, std::vector, std::list, std::set, std::map 등 다양한 형태를 모두 정렬할 수 있게 해준다!

```
#include <algorithm>
std::sort(정렬할 데이터의 시작 주소, 정렬할 자료의 마지막 주소, [비교함수]);
```


```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 5,2,3,4,1 };
    std::sort(arr, arr + 5);

    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);
}
```

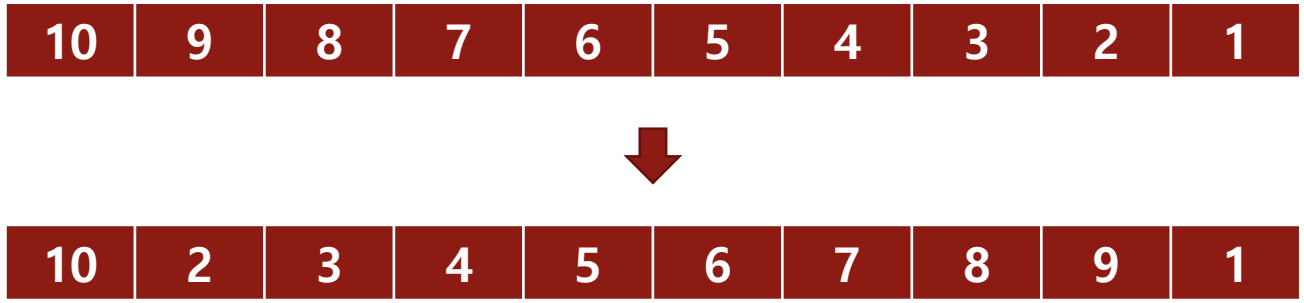
```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 5,2,3,4,1 };
    std::sort(arr, arr + 5);

    for (int i = 0; i < 5; i++)
        printf("%d ", arr[i]);
}
```


```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    std::sort(arr + 1, arr + 9);

    for (int i = 0; i < 10; i++)
        printf("%d ", arr[i]);
}
```

```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    std::sort(arr + 1, arr + 9);

    for (int i = 0; i < 10; i++)
        printf("%d ", arr[i]);
}
```

```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    std::sort(arr + 1, arr + 9);

    for (int i = 0; i < 10; i++)
        printf("%d ", arr[i]);
}
```

```cpp
#include <stdio.h>
#include <algorithm>

int main() {
    int arr[] = { 10,9,8,7,6,5,4,3,2,1 };
    std::sort(arr + 1, arr + 9);

    for (int i = 0; i < 10; i++)
        printf("%d ", arr[i]);
}
```



std:: 왜 자꾸 붙지

using namespace std;

를 실행한 다음이면 std::를 생략 가능.

실제 개발에서는 그러지 말라고 합니다.

#2751 수 정렬하기 2

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	39064	11922	7676	34.260%	0.51

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

#2751 수 정렬하기 2

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	39064	11922	7676	34.260%	0.51

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

$$n \log n = 106 \log 10^6 \approx 2 \times 10^7$$

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

#2751 수 정렬하기 2

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	39064	11922	7676	34.260%	0.51

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

$$n \log n = 106 \log 10^6 \approx 2 \times 10^7$$

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

```
#include <stdio.h>
#include <algorithm>
int arr[1000001];
int main() {
    int n; scanf("%d", &n);
    for(int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    std::sort(arr, arr + n);

    for (int i = 0; i < n; i++)
        printf("%d\n", arr[i]);
}
```

#11931 수 정렬하기 4

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	2482	1291	1023	57.537%	3.93

문제

N개의 수가 주어졌을 때, 이를 내림차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 내림차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

#11931 수 정렬하기 4

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	2482	1291	1023	57.537%	3.93

문제

N개의 수가 주어졌을 때, 이를 내림차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 내림차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

```
std::sort(정렬할 데이터의 시작 주소, 정렬할 자료의 마지막 주소, [비교함수]);
```

정렬 기준을 위한 비교 함수

- 비교함수가 없을 시 오름차순 정렬을 base
- 내림차순, 구조체 정렬 등 특수한 경우에 대해서는 정의가 필요

Examples

- 오름차순

```
bool compare(int a, int b) {
    return a < b;
}
```

왼쪽의 원소가 오른쪽의 원소보다
값이 작도록 정렬

- 내림차순

```
bool compare(int a, int b) {
    return a > b;
}
```

왼쪽의 원소가 오른쪽의 원소보다
값이 크도록 정렬

Examples

- 변수가 여러 개인 구조체에 대하여 정렬

```
struct P {
    int x, y;
}

bool compare(P a, P b) {
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}
```

왼쪽 원소의 x값이 작은 순으로 정렬하되,
같을 경우 y값이 작은 순으로 정렬

#11931 수 정렬하기 4

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	2482	1291	1023	57.537%	3.93

문제

N개의 수가 주어졌을 때, 이를 내림차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 1,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 절댓값이 1,000,000보다 작거나 같은 정수이다. 수는 중복되지 않는다.

출력

첫째 줄부터 N개의 줄에 내림차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

```cpp
#include <stdio.h>
#include <algorithm>
using namespace std;
int arr[1000001];
bool comp(int a, int b){ return a > b; }
int main() {
    int n; scanf("%d", &n);
    for(int i = 0; i < n; i++)
        scanf("%d", &arr[i]);
    sort(arr, arr + n, comp);
    for (int i = 0; i < n; i++)
        printf("%d\n", arr[i]);
}
```

#11650 좌표 정렬하기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
1 초	256 MB	9891	5031	3806	51.677%	1.07

문제

2차원 평면 위의 점 N개가 주어진다. 좌표를 x좌표가 증가하는 순으로, x좌표가 같으면 y좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 점의 개수 N ($1 \le N \le 100,000$)이 주어진다. 둘째 줄부터 N개의 줄에는 i번점의 위치 x_i와 y_i가 주어진다. ($-100,000 \le x_i, y_i \le 100,000$) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

출력

첫째 줄부터 N개의 줄에 점을 정렬한 결과를 출력한다.

#11650 좌표 정렬하기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
1 초	256 MB	9891	5031	3806	51.677%	1.07

문제

2차원 평면 위의 점 N개가 주어진다. 좌표를 x좌표가 증가하는 순으로, x좌표가 같으면 y좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 점의 개수 N ($1 \le N \le 100,000$)이 주어진다. 둘째 줄부터 N개의 줄에는 i번점의 위치 x_i와 y_i가 주어진다. ($-100,000 \le x_i, y_i \le 100,000$) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

출력

첫째 줄부터 N개의 줄에 점을 정렬한 결과를 출력한다.

1. 좌표 단위로 관리하기 위한 자료형 필요

2. 해당 자료형에 대해 정렬 기준 필요

```
struct P {
    int x, y;
}

bool compare(P a, P b) {
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}
```

#11650 좌표 정렬하기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
1 초	256 MB	9891	5031	3806	51.677%	1.07

문제

2차원 평면 위의 점 N개가 주어진다. 좌표를 x좌표가 증가하는 순으로, x좌표가 같으면 y좌표가 증가하는 순서로 정렬한 다음 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 점의 개수 N (1 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N개의 줄에는 i번점의 위치 x_i와 y_i가 주어진다. (-100,000 ≤ x_i, y_i ≤ 100,000) 좌표는 항상 정수이고, 위치가 같은 두 점은 없다.

출력

첫째 줄부터 N개의 줄에 점을 정렬한 결과를 출력한다.

```
#include <stdio.h>
#include <algorithm>
using namespace std;
struct P { int x, y; } arr[100001];
bool comp(P a, P b){
    if (a.x == b.x) return a.y < b.y;
    return a.x < b.x;
}
int main() {
    int n; scanf(“%d”, &n);
    for(int i = 0; i < n; i++)
        scanf(“%d%d”, &arr[i].x, &arr[i].y);
    sort(arr, arr + n, comp);
    for (int i = 0; i < n; i++)
        printf("%d %d\n", arr[i].x, arr[i].y);
}
```

정수만 되느냐?

- 다른 데이터 타입도 가능

ex) char의 경우, 각각의 아스키코드상의 번호로 치환 => 대소비교 가능

길이가 다른 문자열에 대해서도 가능

<string> Data type

- c에서 문자열을 다룰 경우

  char[ ]을 통해 표현.

- <string>헤더를 추가할 경우 string을 하나의 Data type으로서 취급 가능!
- 입출력 헤더로 <stdio.h>가 아닌 <iostream> 사용
- cin & cout을 통한 입출력

```c
#include <stdio.h>
#include <string.h>
int main() {
    int a, b;
    char buffer[10];
    scanf ("%d%d", &a, &b);
    scanf ("%s", buffer);
    printf ("%d+%d=%d\n", a, b, a + b);
    return 0;
}
```

=

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    int a, b;
    string buffer;
    cin >> a >> b;
    cin >> buffer;
    cout << a << "+" << b << "=" << a + b;
    return 0;
}
```

<string> 사용법

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

int main() {
    string str = "hello";


    printf("%s\n", str);    //wrong method
    cout << str << '\n';
    string str2 = "world!\n";
    str = str + ' ';    //same as str.push_back(' ');
    str += str2;
    cout << str;
    int Size = str.size();
    string str3;
    cin >> str3;
    str3.pop_back();
    return 0;
}
```

More info :

https://en.cppreference.com/w/cpp/string/basic_string

여러 문자열의 사전식 정렬

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

int main() {
    int n;
    string str[100];

    cin >> n;
    for (int i = 0; i < n; i++)
    cin >> str[i];
    sort(str, str + n);

    for (int i = 0; i < n; i++)
        cout << str << '\n';
}
```

#1181 단어정렬

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	18925	6835	4873	37.057%	0.79

문제

알파벳 소문자로 이루어진 N개의 단어가 들어오면 아래와 같은 조건에 따라 정렬하는 프로그램을 작성하시오.

1. 길이가 짧은 것부터
2. 길이가 같으면 사전 순으로

입력

첫째 줄에 단어의 개수 N이 주어진다. (1≤N≤20,000) 둘째 줄부터 N개의 줄에 걸쳐 알파벳 소문자로 이루어진 단어가 한 줄에 하나씩 주어진다. 주어지는 문자열의 길이는 50을 넘지 않는다.

출력

조건에 따라 정렬하여 단어들을 출력한다. 단, 같은 단어가 여러 번 입력된 경우에는 한 번씩만 출력한다.

#1181 단어정렬

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율	Tier
2 초	256 MB	18925	6835	4873	37.057%	0.79

문제

알파벳 소문자로 이루어진 N개의 단어가 들어오면 아래와 같은 조건에 따라 정렬하는 프로그램을 작성하시오.

1. 길이가 짧은 것부터
2. 길이가 같으면 사전 순으로

입력

첫째 줄에 단어의 개수 N이 주어진다. (1≤N≤20,000) 둘째 줄부터 N개의 줄에 걸쳐 알파벳 소문자로 이루어진 단어가 한 줄에 하나씩 주어진다. 주어지는 문자열의 길이는 50을 넘지 않는다.

출력

조건에 따라 정렬하여 단어들을 출력한다. 단, 같은 단어가 여러 번 입력된 경우에는 한 번씩만 출력한다.

```
bool cmp(string a, string b) {
    if (a.size() != b.size())
        return a.size() < b.size();
    return a < b;
}
```

vector

- 동적 배열
- malloc, calloc, free를 할 필요 없이 크기 변경, 삭제 및 제거가 용이
- 배열이기 때문에 각 원소를 index로 참조 가능


```cpp
#include <stdio.h>
#include <vector>
using namespace std;
int main() {
    vector<int> arr;
    arr.resize(100);
    for (int i = 0; i < 100; i++)
        arr[i] = i;

return 0;
}
```

=

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    int *arr;
    arr = (int *)malloc (sizeof(int) * 100);
    for (int i = 0; i < 100; i++)
        arr[i] = i;
    free(arr);
    return 0;
}
```

```cpp
#include <stdio.h>
#include <vector>
using namespace std;
int main() {
    vector<int> arr;
    arr.resize(100);
    for (int i = 0; i < 100; i++)
        arr[i] = i;

return 0;
}
```

=

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    int *arr;
    arr = (int *)malloc (sizeof(int) * 100);
    for (int i = 0; i < 100; i++)
        arr[i] = i;
    free(arr);
    return 0;
}
```

추가적인 vector 사용법

```cpp
int main() {
    vector<int> arr;

    arr.push_back(1);
    arr.push_back(2);
    for (int i = 0; i < arr.size(); i++)
        printf("%d\n", arr[i]);
    arr.pop_back();
    arr.push_back(2);
    printf("%d\n", arr.front());
    printf("%d\n", arr.back());
    arr.clear();

    arr.resize(10);
    vector<int> arr2(5);
    vector<int> arr3(3, -1);
    vector < vector<int>> two_dim_arr;
    return 0;
}
```

More info :

https://en.cppreference.com/w/cpp/container/vector

binary_search

- 범위 내에 찾고자 하는 값이 존재하는지 여부를 반환

```
#include <algorithm>

using namespace std;


//return boolean type result

bool binary_search(시작 주소, 끝 주소, key value);
```

#1920 수 찾기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	31965	8388	5545	27.553%

문제

N개의 정수 A[1], A[2], ..., A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 N(1≤N≤100,000)이 주어진다. 다음 줄에는 N개의 정수 A[1], A[2], ..., A[N]이 주어진다. 다음 줄에는 M(1≤M≤100,000)이 주어진다. 다음 줄에는 M개의 수들이 주어지는데, 이 수들이 A안에 존재하는지 알아내면 된다. 모든 정수들의 범위는 int 로 한다.

출력

M개의 줄에 답을 출력한다. 존재하면 1을, 존재하지 않으면 0을 출력한다.

#1920 수 찾기

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	31965	8388	5545	27.553%

문제

N개의 정수 A[1], A[2], ..., A[N]이 주어져 있을 때, 이 안에 X라는 정수가 존재하는지 알아내는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 N(1≤N≤100,000)이 주어진다. 다음 줄에는 N개의 정수 A[1], A[2], ..., A[N]이 주어진다. 다음 줄에는 M(1≤M≤100,000)이 주어진다. 다음 줄에는 M개의 수들이 주어지는데, 이 수들이 A안에 존재하는지 알아내면 된다. 모든 정수들의 범위는 int 로 한다.

출력

M개의 줄에 답을 출력한다. 존재하면 1을, 존재하지 않으면 0을 출력한다.

```
vector <int> arr;
int main() {
    int n, m; scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        int tmp; scanf("%d", &tmp);
        arr.push_back(tmp);
}
    sort(arr.begin(), arr.end());
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        int key; scanf("%d", &key);
        if (binary_search(arr.begin(), arr.end(), key))
            printf("1\n");
        else printf("0\n");
    }
}
```

Problem set

- 2751 수 정렬하기 2
- 11931 수 정렬하기 4
- 11650 좌표 정렬하기
- 1181 단어정렬
- 5052 전화번호 목록
- 1920 수 찾기
- 2805 나무 자르기

- 이외 문제집

여러 정렬 방법에 익숙해지기 위한 문제들(Silver)

이분 탐색(Silver)