C++ 코딩테스트

C++을 사용하여 코딩테스트를 준비할 때
필수적으로 알고있어야 하는 것들

01

하나,

제약사항 분석

자료형의 범위
시간복잡도와 공간복잡도

02

두울,

데이터

입력
문자열 함수

03

세엣,

자료구조

자료구조의 종류
자료구조의 차이

제약사항

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
10 초	512 MB	399	107	48	27.746%

시간 제한 = 시간 복잡도
메모리 제한 = 공간 복잡도
제약 조건 = 자료형과 자료형의 범위

제약조건

- 모든 입력은 정수로 주어진다.
- $1 \le \mathbf{T} \le 100$.
- $1 \le \mathbf{N} \le 1000$.
- $0 \le \mathbf{P_i} \le 100000$.
- $1 \le \mathbf{S_i} \le 1000$.

C/C++ 자료형의 범위

자료형	키워드	값의 범위	
정수형	short(2byte)	-32,768~32,767	32만
	int(4byte)	-2,147,483,648~2,147,438,647	21억
	long long(8byte)	-9,223,372,036,854,775,808~ 9,223,372,036,854,775,808	922경

제약조건에서 값의 범위가 21억을 넘어갈 경우 long long을 사용하여야 함

03

알고리즘의 속도(시간 복잡도)

빅 오	N	1억 + a 정도의 시간이면 1초에 가능
O(N)	약 1억	1억
O(N*logN)	약 1000만	log(1000만) = 7.1 x 1000만 = 7100만
O(N^2)	약 10000	10000 x 10000 = 100,000,000
O(N^3)	약 500	500 x 500 x 500 = 125,000,000

알고리즘을 Big-O 계산법으로 나타내었을 때의 시간

리스트의 크기(공간 복잡도)

* int형의 크기는 4byte

int a[1000]: 4KB
int a[1000000]: 4MB
int a[2000][2000]:16MB

코딩테스트에서는 보통 메모리 용량을 128~512MB 정도로 제한하기 때문에, 일반적으로 데이터의 개수가 1000만 단위를 넘지않도록 알고리즘을 설계해야함

에러의 종류(BOJ)

틀렸습니다: 출력 결과가 정답과 다른 경우입니다.

시간 초과: 프로그램이 제한된 시간 이내에 끝나지 않은 경우입니다. 시간 제한을 초과하면 실행을 즉시 중단합니다. 따라서, 정답이 맞았는지 틀렸는지는 알 수 없습니다.

메모리 초과: 프로그램이 허용된 메모리보다 많은 메모리를 사용한 경우입니다. 시간 초과와 마찬가지로, 메모리 제한을 초과하면 실행을 즉시 중단합니다.

출력 초과: 프로그램이 너무 많은 출력을 하는 경우에 발생합니다. 이 결과는 틀렸습니다와 같은 의미를 갖습니다. 정확하게는 미리 구해놓은 정답 파일 크기의 2배를 넘어가면 이 결과를 받게 됩니다. 만약, 출력하는 부분이 무한 루프에 빠진 경우, 시간을 초과하기 전에 출력 초과를 받을 수 있습니다.

런타임 에러: 프로그램이 비정상적으로 종료한 경우입니다. Exit code가 0이 아닌 경우, `segmentation fault`를 받은 경우가 대표적입니다.

컴파일 에러: 컴파일에 실패한 경우입니다. 경고(Warning)가 있어도 컴파일에 성공하면 이 결과를 받지 않습니다. 컴파일 에러를 클릭하면 컴파일 에러 메시지를 볼 수 있습니다.

01

하나,

제약사항 분석

자료형의 범위
시간복잡도와 공간복잡도

02

두울,

데이터

입력
문자열 함수

03

세엣,

자료구조

자료구조의 종류
자료구조의 차이

입력

```cpp
#include <bits/stdc++.h>
using namespace std;
int a;
int main(){
    cin >> a;
    scanf("%d", &a);
    return 0;
}
```

```cpp
scanf("%d.%d", &m1, &m2); //3.22 을 받을때
```

형식을 기반으로 입력이 주어진 경우: scanf
나머지 경우: cin

한 줄씩 입력받는 경우

```
#include <bits/stdc++.h>
using namespace std;
int a;
int main(){
    cin >> a;
    scanf("%d", &a);
    return 0;
}
```

getline

입력이 계속해서 이어질 때

```
while (scanf("%d", &n) != EOF)
while (cin >> n) // cin으로는
```

문제에 입력을 주다가 안줄 때 종료된다고 명시되어 있는 경우

문자열 함수

함수명	종류	
reverse	원래의 문자열을 바꿔버림.	reverse(s.begin(), s.end()) or reverse(s.begin(), s.begin()+3)
substr	시작지점으로부터 몇 개의 문자열을 뽑아냄	s.substr(0,3)
find	어떠한 문자열이 들어있는지를 찾음. 원하는 문자열을 찾지 못한다면 문자열의 끝 위치인 string::npos를 반환	s.find("abc")

split

```cpp
#include <bits/stdc++.h>
using namespace std;

vector<string> split(string input, string delimiter) {
    vector<string> ret;
    long long pos = 0;
    string token = "";
    while ((pos = input.find(delimiter)) != string::npos) {
        token = input.substr(0, pos);
        ret.push_back(token);
        input.erase(0, pos + delimiter.length());
    }
    ret.push_back(input);
    return ret;
}


int main(){
    string s = "안녕하세요 큰돌이는 바보에요 정말이에요!";
    string d = " ";
    vector<string> a = split(s, d);
    for(string b : a) cout << b << "\n";\
```

분해할 기준이 되는 delimiter를 기본으로 input에서 시작 위치를 찾고, 찾은 위치를 기반으로 input을 앞에서부터 지워가며 다시 찾는 로직

최종적으로 ret값을 반환함

C/C++은 Python과 다르게 split을 지원하지 않기 때문에 직접 함수를 구현해야함

01

하나,

목차를 입력해주세요

세부내용을 입력해주세요
세부내용을 입력해주세요

02

두울,

데이터

입력
문자열 함수

03

세엣,

자료구조

자료구조의 종류
자료구조 차이

pair와 tuple

```cpp
#include<bits/stdc++.h>
using namespace std;
pair<int, int> pi;
tuple<int, int, int> tl;
int a, b, c;
int main(){
    pi = {1, 2};
    tl = make_tuple(1, 2, 3);
    tie(a, b) = pi;
    cout << a << " : " << b << "\n";
    tie(a, b, c) = tl;
    cout << a << " : " << b << " : "<< c << "\n";
   return 0;
}
```

두 가지의 값을 담을 때 : pair
세 가지 이상의 값을 담을 때: tuple

```cpp
#include<bits/stdc++.h>
using namespace std;
pair<int, int> pi;
tuple<int, int, int> ti;
int a, b, c;
int main(){
    pi = {1, 2};
    a = pi.first;
    b = pi.second;
    cout << a << " : " << b << "\n";
    ti = make_tuple(1, 2, 3);
    a = get<0>(ti);
    b = get<1>(ti);
    c = get<2>(ti);
    cout << a << " : " << b << " : "<< c << "\n";
   return 0;
}
```

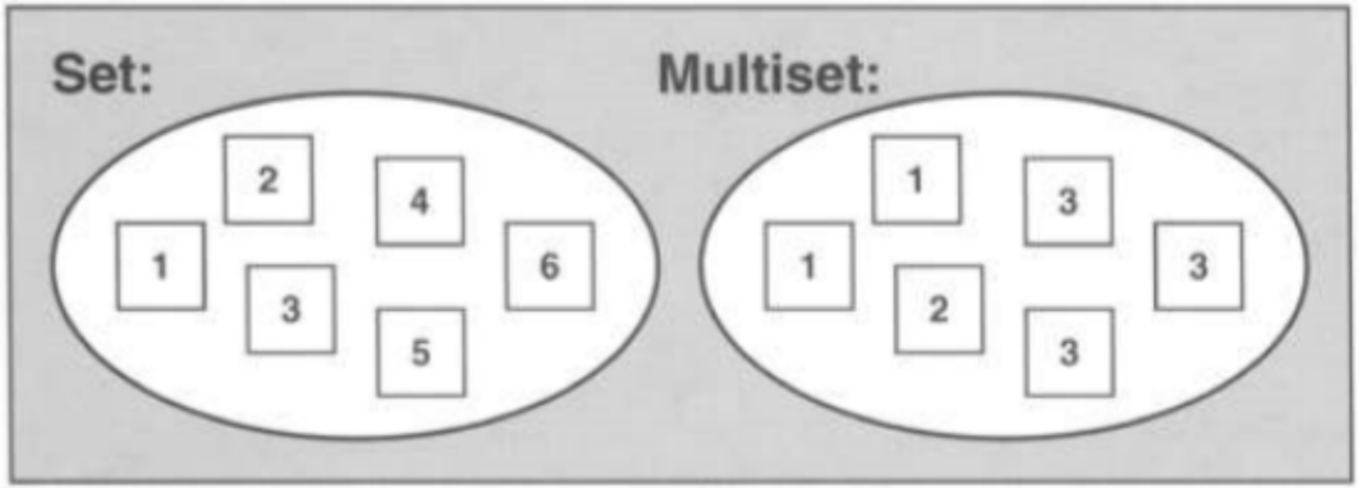
값을 가져올 때
pair도 make_pair사용가능

map과 unordered_map

```
unordered_map<string, int> umap;
map<string, int> _map;
```

key와 value형태로 이루어져 있는 자료구조
map: 정렬 / 레드블랙트리 / 탐색, 삽입, 삭제에 O(logN)
unordered_map: 정렬 x / 해시테이블 / 탐색, 삽입, 삭제에 평균적으로 O(1), 최악의 경우 O(N)

set과 multiset



map과 다르게 key만 저장하는 자료구조
set: 유니크한 값만 넣을 수 있음
multiset: 중복되는 원소도 집어넣을 수 있음

자료구조 사용

벡터(vector)

- 저장할 개수가 가변적일때 사용함
- 중간에 데이터 삽입이나 삭제가 일어나지 않을때
- 빈번한 검색이 별로 없을 때(순차저장이기 때문에 검색 속도가 느림)
  -> (검색에는 map, set, hash_map가 좋다.)
- 데이터 랜덤접근이 용이하다.
- 배열같은 특성이 있어 랜덤 접근이 가능함

Set

- 맵과는 다르게 '키(Key)'만 저장한다.
- 정렬해야 할 때 사용한다.
- key가 있는지 없는지 알아야 할 때 사용한다.
- 많은 자료를 저장하고, 검색 속도가 빨라야 할 때 사용한다.
- 중복키 사용 불가

자료구조 사용

hash_map

- 많은 자료(수천단위)를 저장하고, 검색 속도가 빨라야 할때 사용
- 너무 빈번하게 자료를 삽입, 삭제를 할경우에는 사용하지 않음
- 중복키 사용 불가
- unordered_map

map

- 많은 자료들을 정렬해야 할때 사용
- 많은 자료를 저장하고, 검색이 빨라야 할때 사용
- 번번하게 삽입, 삭제를 할경우에는 사용하지 않는다.
- 해쉬맵과 다른점은 맵은 자동으로 정렬을 하기 때문에 정렬이 필요하지 않는 곳에서 맵을 사용하는것은 낭비
- 중복키 사용 불가

C++을 활용하여 코딩테스트를 준비할 때
필수적으로 알고 있어야 하는 것들

감사합니다!