

# 8주차 : 문자열과 시간복잡도

---

강사: 표지원

# 문자열

---

이걸 몰라?

## 문자열이란?

“문자 들의 열” 말 그대로, 문자들의 배열이다!



char



char[]

```
1  #include <stdio.h>
2
3  √ int main()
4  {
5      char word = 'a';
6      char name[12] = "Hello world";
7  }
```

일반적인 값의 수정은 다른 배열들과 마찬가지로 방식이다.

## 문자열이란?

배열은 배열인데, 좀 특별한 특징이 있다면 **마지막 칸에 Null 값**이 들어간다!

(Null값은 문자열이 끝났음을 의미한다.)

name[12] =

0	1	2	3	4	5	6	7	8	9	10	11
'H'	'e'	'l'	'l'	'o'	' '	'w'	'o'	'r'	'l'	'd'	0

띄어쓰기  
기도문  
자야

널값은  
0이야

## 문자열이란?

근데 사실, 여러분은 이미 다 저 내용들을 알고있다.  
근데 사실, 여러분은 이미 다 저 내용들을 알고있다.  
근데 사실, 여러분은 이미 다 저 내용들을 알고있다.  
근데 사실, 여러분은 이미 다 저 내용들을 알고있다.

소입설 들었자나.....

따라서! 이번 강의에서는, **C\_style** 문법과, **C++ style**의 비교를 위주로 볼것이고  
**비교한 결과 까지 한번 볼것이다!**

# char [] or string

---

갠.취.존.중

C스타일? 혹은 C++ 스타일?

## C-style 문법

C스타일이라고 하면, 크게 어렵거나 복잡한 것은 아니다.

여러분들이 소입설에서 주로 다루었고, 또 1학기 ALOHA수업에서 다룬게 다이다.

그니까,

```
1  ✓ #include <stdio.h>  
2    #include <string.h>
```

이 두 헤더를 먼저 include 하고 쓰는 것들이 전부임!

# C-style 문법

아는 걸 정리해본다면,

```
int main()
{
    const int LENGTH = 100;
    char str[LENGTH];
}
```

입력

scanf("%s", str)



scanf는 VS2019에서는 안될 수도 있다!

출력

printf("%s", str)

수정

str[5] = 'A' 혹은, string.h의 함수 사용.



0-based이고, 마지막 값은 건드리면 안된다!

삭제

덮어쓰우기 외엔, C에서는 배열 값의 삭제는 불가능하다.



# C-style 문법

좀 더 확장해보는다면?

△ 입력

출력

△ 수정

삭제

`scanf("%s", str)`

`printf("%s", str)`

`str[5] = 'A'` 혹은, `string.h`의 함수 사용.

덮어씌우기외엔, C에서는 배열 값의 삭제는 불가능하다.

```
int main()
{
    const int LENGTH = 100;
    char str[LENGTH];
}
```

한 줄 전체 입력

한 글자 입력

다른 방식의 출력

`scanf("%[^\\n]", str)`

`ch = getchar()`

`puts(str)`

`fgets(str, sizeof(str), stdin)`



개행문자 (\\n)가 문자열 마지막에 담긴다.



버퍼에서 한 글자를 가져오는 방식이다.  
While문과 사용하거나, `scanf`가 남기는 '\\n'을 제거

# C-style 문법

좀 더 확장해보는다면?

문자열 **비교**

strcmp(str1, str2)



str1 - str2 라고 생각하자  
문자열이 일치하면 0을 반환

문자열 **이어붙이기**

strcat(str1, str2)



str1뒤에 str2를 붙여줌

문자열 **길이 조사**

strlen(str)

△ 입력

scanf("%s", str)

출력

printf("%s", str)

△ 수정

str[5] = 'A' 혹은, string.h의 함수 사용.

삭제

뺏어썩우기외엔, C에서는 배열 값의 삭제는 불가능하다.

```
int main()
{
    const int LENGTH = 100;
    char str[LENGTH];
}
```

# C-style 문법

⚠ 입력

`scanf("%s", str)`

출력

`printf("%s", str)`

⚠ 수정

`str[5] = 'A'` 혹은, `string.h`의 함수 사용.

삭제

뺏어썩우기외엔, C에서는 배열 값의 삭제는 불가능하다.

```
int main()
{
    const int LENGTH = 100;
    char str[LENGTH];
}
```

scanf는 정수만 입력받으면,

개행문자를 남긴다!

그래서 fgets랑 활용이 불가능할 때도 있다.

# C-style 문법

```
1  #include <stdio.h>
2
3  ✓ int main()
4  {
5      int n;
6      char s[25];
7      scanf("%d", &n);
8
9      fgets(s, sizeof(s), stdin);
10
11     printf("%s", s);
12 }
```

△ 입력	scanf("%s", str)
출력	printf("%s", str)
△ 수정	str[5] = 'A' 혹은, string.h의 함수 사용.
삭제	뺄어썬우기외엔, C에서는 배열 값의 삭제는 불가능하다.

```
int main()
{
    const int LENGTH = 100;
    char str[LENGTH];
}
```

Microsoft Visual Studio 디버그 콘솔

```
123
C:\Users\USER\source\repos\PS_prac\Debug\PS
이 창을 닫으려면 아무 키나 누르세요.
```

이 문제는 **입력버퍼**랑 관련이 있다!!

(123하고 다음 입력을 받아야 하는데, 개행문자가 입력에 들어가버렸다 ㅠㅠ)

잠깐! 버퍼요?

“C언어의 입출력 함수들은 내부적으로 입출력 **버퍼**를 사용합니다.”

**버퍼**란?

프로그램 및 운영체제가 데이터를 효율적으로 처리하기 위해 사용하는 **임시 저장소**와 같은 것!

우리가 어떠한 값들을 **키보드로 입력하면**,  
그 값들은 사실 버퍼에 순차적으로 담긴다.

## 잠깐! 버퍼요?

그렇게 순차적으로 담긴 값에 대해서 입출력 함수가 데이터를 입맞대로 가져간다.

예를들어, `scanf("%d", &n)`는 White Space(띄어쓰기, Tab, 개행 등) **직전까지** 값을 받고, White space 값을 무시한다.

그말은 즉슨, 12345(엔터) 를 입력했을 때!

**버퍼**에는 **12345W**`n` 이 순서대로 담기고,

`scanf`는 **12345까지만 버퍼에서 뽑아간다**는 얘기이다!

## 잠깐! 버퍼요?

그럼 버퍼에는 **wn만이 남아있을 것**인데, 이 상태에서 `scanf("%s", str)`을 쓰면?

`scanf("%s", str)`는 **White Space까지 값을 다 뽑아**와서,  
**White Space 대신 NULL값을 문자열 마지막 칸에 넣어준다.** (문자열 마지막 값!)

그래서! 숫자를 입력하고 엔터를 눌렀을 때 문자열에는 wn만이 들어가기 때문에,  
str은 빈 문자열이 되는 것이다!!!

## C-style 문법

그럴 땐 이렇게 해결할 수 있다.

```
scanf("%d", &n)
```

```
getchar();
```

애를 추가!

```
fgets(s, sizeof(s), stdin);
```



getchar()는 입력버퍼에서 한글자를 가져오는 기능이 있다.



## C-style 문법

scanf끼리 입력받는 경우엔?  
이렇게 해결할 수 있다.

scanf("%s", str)



scanf(" %s", str)



scanf에서, 앞의 공백은 이전 wn값등을 무시하는 기능이 있다.

C-style 문법  
활용해보자



**1380 귀걸이**

# 한 줄 입력 받기  
# 배열관리

C-style 문법  
활용해보자



**1380 귀걸이**

# 한 줄 입력 받기  
# 배열관리

학생 이름을 다 **입력받고**

2N - 1번 **카운트** 해주고

1번만 불린 놈만 체크해주면 되겠다!

## C-style 문법 활용해보자



**1380 귀걸이**

# 한 줄 입력 받기  
# 배열관리

주의할 점은,

이름은 엄청 길거나, 띄어쓰기가 많을 수 있다.

따라서 입력 받을 때 조심!

A,B 체크할 때도 getchar() 신경써줘야한다!

# C-style 문법 활용해보자



## 1380 귀걸이

# 한 줄 입력 받기  
# 배열관리

```
1  #include <stdio.h>
2  int main()
3  {
4      int n, a, cnt = 1;
5      char b;
6      while (1)
7      {
8          scanf("%d", &n);
9          if (n == 0) break;      scanf도 getchar() 잘써줘야한다.
10         getchar();
11
12         char arr1[102][62] = { 0, };      입력 받기 위해 배열 생성
13         bool arr2[102] = { 0, };
14
15         for (int i = 0; i < n; i++)
16         {
17             fgets(arr1[i], sizeof(arr1[i]), stdin);      한 줄 받기
18         }
19
20         for (int i = 0; i < 2 * n - 1; i++)
21         {
22             scanf("%d %c", &a, &b);
23             arr2[a - 1] = !arr2[a - 1];
24         }      짝수 번이면, 후보가 아니니까 온오프!
25
26         for (int i = 0; i < n; i++)
27         {
28             if (arr2[i] == true) printf("%d %s", cnt++, arr1[i]);
29         }      2N-1번 했으니 true는 하나겠지, 그거 출력!
30     }
31 }
```

## C-style 주의사항

입력은 다양하게 받을 수 있는데, **개행문자 남는거 주의!**

scanf, fgets 등등.. 사실 gets도 있었는데 (없었습니다)  
(C버전이 올라가면서 사라짐)

Char 배열은 **0-based**이고, **마지막 값은 NULL**인 거 생각하자

동적으로 할당하지 않으니, **배열 크기도 신경 써야됨!**

C++ style 문법은 어떨까?

## C++ style

입출력도 cin, cout을 이용하며,  
**string** class를 이용한다!

```
1  ∨ #include <iostream>
2    #include <string>
3    using namespace std;
4
5  ∨ int main()
6    {
7        int a;
8        cin >> a;
9        cout << a;
10   }
```

입출력이구요.

```
1  ∨ #include <iostream>
2    #include <string>
3    using namespace std;
```

사용하는 헤더예요



## C++ style

```
1  ✓ #include <iostream>
2    #include <string>
3    using namespace std;
```

좀 특이한건,

using namespace std;

namespace요?

## C++ style

namespace는  
함수나, 객체 등등의  
일종의 소속을 밝히는 키워드와 같다.

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

std에 속해 있다고 미리 알리지 않으면,  
왼쪽처럼 매번 알려줘야 한다.

(그러니까 항상 써줍시다.)

## C++ style

Q. 그럼 String? Class는 또 뭐예요?

```
class Point
{
    int x,y;
    // 구조체처럼 멤버 변수도 갖구요.
public:
    Point(int x, int y) { this->x = x, this->y = y; }
    // 생성자라고 생성할때 호출하는 함수도 만들수 있구
    int get_xpos() { return x; }
    void set_xpos(int x) { this->x = x; }
    // 이렇게 내부 함수도 가져요.
    // STL때 배운거 그대로입니다
    // 히히
};
```

A. **class** 는 멤버 함수, 변수를 갖는 틀이고  
구조체를 선언하고  
객체를 찍어냈던 것처럼 사용 가능하다!

구조체 확장 ver.

# C++ style

```
int main()
{
    string str = "Hello world";
    // 생성은 다양하게 가능해요.
    str = str + str;
    // +, - 연산을 지원해요.
    str[0] = 'h';
    // 배열처럼 접근해서 값의 수정도 할 수 있어요.

    int length = str.length();
    // 길이를 반환받아서 for문 돌릴 수도 있지요.
    str.push_back('w');
    // push, pop도 지원하구요.
    str.c_str();
    // 이렇게 const char * (c_style)처럼 변신도 가능!
}
```

나름 정리해 봤어요

사실 **string** class는 STL중 하나이다!

자세한 내용은

레퍼런스 사이트를 참고하자!

## C++ style

아는 걸 정리해보는다면,

```
int main()
{
    string str1 = "Hello world";
    string str2;
    str2 = "HiHi";
}
```

입력

`cin >> str1 >> str2;`



꺾새 방향 주의!!

출력

`cout << str1 << str2 << '\n';`



여러 번 쓸 수 있다!

수정

`str[5] = 'A'` 혹은, string의 함수, 연산자 사용.

삭제

- 연산자나, `pop_back()` 등 매우 다양한 방법이 있다.

매우 편리하고 쉽다! 뒤편.....

## C++ style

좀 더 확장해본다면?

△ 입력

`cin >> str1 >> str2;`

△ 출력

`cout << str1 << str2;`

수정

`str[5] = 'A'` 혹은, `string`의 함수, 연산자 사용.

삭제

- 연산자나, `pop_back()` 함수 등 매우 다양한 방법이 존재.

```
int main()
{
    string str1 = "Hello world";
    string str2;
    str2 = "HiHi";
}
```

한 줄 전체 입력

`cin.getline(str, 'Wn');`

`getline(cin, s, 'Wn')`



뒤에 'Wn'는 제한자를 의미한다.  
제한자는 'Wn' 말고 다른것도 된다!

문자열 비교

`str1 == str2` (혹은, `>`, `<` 연산자)

## C++ style

제한자는 또 뭐야?

어디 전까지 받아와주세요를 나타내는거임!

`getline(cin, s, '\n')`

string인 s에 '\n' 직전까지 입력!

`cin.getline(s, '#');`

string인 s에 '#' 직전까지 입력!

(얘네는 '\n' 을 남기지 않아서, 귀찮게 `getchar()` 쓰는거 안해도 됨!!)

C++ style 문법  
활용해보자



**1380 귀걸이**

# 한 줄 입력 받기  
# 배열관리



## C++ style 문법 활용해보자



**1380 귀걸이**

# 한 줄 입력 받기  
# 배열관리

편하게 **cin**으로 받고,  
**vector**<char>으로 관리하자!

vector는, STL이고, 7주차 강의자료에 나와있다!

# C++ style 문법 활용해보자



## 1380 귀걸이

# 한 줄 입력 받기  
# 배열관리

```
1  #include <iostream>
2  #include <string>
3  #include <vector>
4  using namespace std;
5
6  int main()
7  {
8      int t;
9      int cnt = 1;
10     while (cin >> t && t)
11     {
12         vector<char> ggi[105]; //[i] 번째 사람이 몇번 체크를 받았는가를 표시
13         string prn[105]; //[i] 번째 사람의 이름을 담아둠.
14
15         getchar();
16         // 버퍼 비우기용도.
17         // cin으로 숫자를 받고 문자열을 받을때 한번 해줘야됨.
18
19         for(int i=1;i<=t;i++)
20         {
21             getline(cin, prn[i]);
22         }
23         // 이름이 길수도 있으므로 한줄을 다 받음.
24
25         for(int i=1;i<=2 * t-1;i++)
26         {
27             int a; cin >> a;
28             char b; cin >> b;
29             ggi[a].push_back(b);
30         }
31         // A가 2번 혹은 B가 2번 들어오는 경우는 없으므로, 일단 넣어줌.
32
33         for(int i=1;i<=t;i++)
34         {
35             if (ggi[i].size() != 2)
36             {
37                 // 각 사람은 무조건 2개아님 1개의 체크를 받음.
38                 // 그러나 2N-1번 입력이므로 1번 받은사람이 정답
39                 {
38                     cout << cnt++ << ' ' << prn[i] << '\n';
39                     break;
40                 }
41             }
42         }
43     }
44 }
```

C++ style 문법  
활용해보자



**1874 스택 수열**

# 입출력  
# STACK

# C++ style 문법 활용해보자



## 1874 스택 수열

# 입출력  
# STACK

```
1  #include <iostream>
2  #include <vector>
3  #include <stack>
4  using namespace std;
5
6  int main()
7  {
8      vector<int> arr;
9      vector<char> ans;
10     stack<int> stk;
11     int N; cin >> N;
12
13     for (int i = 0; i < N; i++)
14     {
15         int tmp; cin >> tmp;
16         arr.push_back(tmp);
17     }
18
19     int j = 0;
20     for (int i = 1; i <= N; i++)
21     {
22         stk.push(i);
23         ans.push_back('+');
24
25         while (!stk.empty() && stk.top() == arr[j])
26         {
27             stk.pop();
28             ans.push_back('-');
29             j++;
30         }
31     }
32
33     if (!stk.empty()) cout << "NO" << '\n';
34     else {
35         for (int i = 0; i < ans.size(); i++)
36         {
37             cout << ans[i] << '\n';
38         }
39     }
40
41 }
```

cin  
cout을  
자유자재로  
사용해봅시다!

## C++ style

근데 사실!  
cin은  
조금 치명적인  
단점이 있다

수행시간이 너무 차이 난다는 것!

언어	입력방식	수행시간(초)
C/C++	scanf	0.798
C/C++	getchar(*)	0.39
C++	std::cin	2.051
C++	std::ios::sync_with_stdio(false) + std::cin	0.796
Java	java.util.Scanner	6.068
Java	java.io.BufferedReader(*)	0.934
golang	fmt.Scan	44.557
golang	bufio.Reader + fmt.Fscan	9.899
golang	bufio.Scanner(*)	1.299

C++ style 문법  
활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

## C++ style 문법 활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

혹시 안 풀었던 사람들을 위해,  
문제 풀이 방식은

값을 받을때, 스택을 체크,

스택이 **비었으면**, 왼쪽에 탑이 없다는 얘기니까 0출력!

스택이 **차있으면**, 왼쪽에 탑이 있다는 얘기니까

**현재 입력** 받은 탑의 높이랑, 스택의 **top**이랑 비교!

값의 크기가 현재 최고 **탑보다 작으면** pop해서 다음 값 비교

Top의 값이 **높이보다 크거나 같을 경우**, 스택이 빌 경우까지 반복

Top이 같거나 크면, top의 인덱스를 출력, 비었으면 0출력

그 다음 현재 입력 받은 값을 인덱스랑 같이 스택에 넣어준다.

# C++ style 문법 활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

```
1  #include <iostream>
2  #include <stack>
3  using namespace std;
4
5  const int MAX = 5e5 + 5;
6
7  stack<pair<int,int>> T;  <높이, 인덱스> 를 페어로 해서, 스택에 담는다.
8
9  int main() {
10
11     int N; cin >> N;
12     for (int i = 1; i <= N; i++)
13     {
14
15         int k; cin >> k;
16         높이를 입력으로 받는다.
17         while (!T.empty())
18         {
19             if (T.top().first > k)
20             {
21                 cout << T.top().second << ' ';
22                 break; 비교해주는 과정. 작다면 출력하고 break
23             }
24             else
25                 T.pop();  크다면 stack pop
26         }
27         if (T.empty()) cout << 0 << ' ';
28         T.push({ k,i });  Stack이 비었으면 0출력
29         체크 다해줬으면 본인 탑 높이 넣어줌.
30     }
31 }
```



# C++ style 문법 활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

문제 번호	결과	메모리	시간	언어	코드 길이
5 2493	시간 초과			C++14 / 수정	382 B

아니;; 풀었자나요..... 왜 TLE?

## C++ style 문법 활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

언어	입력방식	수행시간(초)
C/C++	scanf	0.798
C/C++	getchar(*)	0.39
C++	std::cin	2.051
C++	std::ios::sync_with_stdio(false) + std::cin	0.796
Java	java.util.Scanner	6.068
Java	java.io.BufferedReader(*)	0.934
golang	fmt.Scan	44.557
golang	bufio.Reader + fmt.Fscan	9.899
golang	bufio.Scanner(*)	1.299



문제는, 입력을 cin으로 많이 받으면서,  
엄청나게 딜레이가 생겼다는 것!

# C++ style 문법 활용해보자

해결 방법은?



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = 5e5 + 5;
5
6 stack<pair<int,int>> go;
7
8 int main() {
9
10     int N; cin >> N;
11     for (int i = 1; i <= N; i++)
12         {
```



```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = 5e5 + 5;
5
6 stack<pair<int,int>> go;
7
8 int main() {
9     ios_base::sync_with_stdio(0);
10    cin.tie(0);
11    cout.tie(0);
12
13    int N; cin >> N;
14    for (int i = 1; i <= N; i++)
```

# C++ style 문법 활용해보자



**2493 답**

# 입출력  
# STACK  
# 7주차 과제였었다!

```
1 #include<bits/stdc++.h>
2 using namespace std;
3
4 const int MAX = 5e5 + 5;
5
6 stack<pair<int,int>> go;
7
8 int main() {
9     ios_base::sync_with_stdio(0);
10    cin.tie(0);
11    cout.tie(0);
12
13    int N; cin >> N;
14    for (int i = 1; i <= N; i++)
```

이게 뭔데요?

사실 cin이 오래 걸리는 이유는

cpp의 iostream과 c의 stdio가  
동기화를 하면서,  
stdio랑 iostream의 버퍼를 모두 사용하기 때문이다!  
(그냥 cin 부스터라고 알고만 있자)

그래서 그것을 해제해 주는 구문이다.

(해제해도 기능은 똑같다.)

대신, c\_style의 입출력이랑 같이 쓰게되면 오류 발생 가능!

# C++ style 문법 활용해보자



**2493 탑**

# 입출력  
# STACK  
# 7주차 과제였었다!

넣고 돌려주시면,

문제 번호	결과	메모리	시간	언어	코드 길이
5 2493	맞았습니다!!	6264 KB	104 ms	C++14 / 수정	440 B

(뽀로롱)

cin 부스트는 선택이 아닌 필수!

## 결론

C style은,

문자열 관리가 조금 어려울 수 있지만, 익숙해지면 편리하다!

딜레이 면에서 고민할 필요가 없다.

대신, format을 일일이 지정한다 던가 하기때문에 코드가 길다

C++ style은,

문자열 관리가 매우 편리하다. (이미 모두 재정의되어있음.)

cin, cout은 알아서 형식따라 저장해주기 때문에 편리하다.

대신, 딜레이를 고려해야하고, 실수표현에 있어서 조금 불편하다.

**근데, 딜레이도 사실 고칠 수 있고, 실수표현은 c랑 섞어쓰면 된다!**

그니까 본인 취향 맞춰서 사용하면 되는데, C++쓰는게 사실 편하긴 하다.

# 시간복잡도

---

시간은 금!

시간복잡도

영어로 Time Complexity

왜 중요할까?



## 시간복잡도

우리는 항상 시공간과 물질에 대해 제약을 받기 때문이다!  
어떤 문제가 생겼다면, 복잡도를 고려해야한다.

사실은 다양한 복잡도가 존재하는데, (공간복잡도 등등..)  
이번에는, **시간에 대한 복잡도**만을 알아보자!

# 시간복잡도

## Big-O: functions ranking

BETTER



WORSE

- $O(1)$  constant time
- $O(\log n)$  log time
- $O(n)$  linear time
- $O(n \log n)$  log linear time
- $O(n^2)$  quadratic time
- $O(n^3)$  cubic time
- $O(2^n)$  exponential time

시간복잡도는  
Big-O 표기법으로 나타낸다

예를 들면, `int a = 1;`  
이렇게 한번의 연산을  $O(1)$ 이라고 적는다.

근데, 모든 연산을 다 적는게 아니라  
대략적으로 영향력 있는 항만 계산한다!

## 시간복잡도

```
int N; cin >> N;
for(int i=1;i<=N;i++)
{
    cin >> A[i];
}
```

Q 처음 입력 1번, For문으로 N번  
이거는 그럼  $O(N + 1)$ 인가요?

A 아뇨, 상수항은 무시합니다.  $O(N)$

```
int N; cin >> N;
for(int i=1;i<=N;i++)
{
    for(int j=1;j<=N;j++)
    {
        cin >> A[i][j];
    }
}

for(int i=1;i<=N;i++)
{
    cin >> B[i];
}
```

Q 아 상수항은 무시군요, 그럼  
얘는  $O(N^2 + N)$ 인가요?

A 아뇨, 계수도 무시하구요,  
영향력 있는 것만 계산해요.  $O(N^2)$

# 시간복잡도

## 대표적인 알고리즘의 시간복잡도

Binary Search

$O(\log N)$



Log의 밑은 2이다!

Qsort

$O(N \log N)$



최악의 경우엔,  $O(N^2)$

DFS/BFS

$O(V + E)$  ,  $O(V^2)$



각각, 인접리스트, 인접행렬

BackTracking

$O(N^N)$

보통은 **본인이 쓴 코드의 시간복잡도를 계산**해서, 시간안에 되는지 **확인**한다.

## 시간복잡도

### TIP

PS에서, 시간제한 1초는 보통 for문 2.5억 번을 의미한다.  
입력이  $N = 20000$ 이고, 본인이 짠 코드가  $O(N^2)$ 라면  
 $N^2 = 4$ 억 이므로, 시간초과 오류를 받게 됨.

코드 작성 이전에, 본인이 떠올린 풀이의 시간복잡도가  
시간안에 돌아가는 것인지 체크하는게 항상 우선이 되어야한다!!

## Big-O: functions ranking

BETTER



WORSE

- $O(1)$  constant time
- $O(\log n)$  log time
- $O(n)$  linear time
- $O(n \log n)$  log linear time
- $O(n^2)$  quadratic time
- $O(n^3)$  cubic time
- $O(2^n)$  exponential time

시간복잡도는!

지금보다는 앞으로에 있어서 매우 중요하니,

자료구조라던가  
자료구조라던가  
자료구조라던가

효율을 위해 본인의 코드에 대한 시간복잡도를  
자주 계산해보는 연습을 하자!

수고하셨습니다!

잘 쉬시고, 담주에 만나요~~!