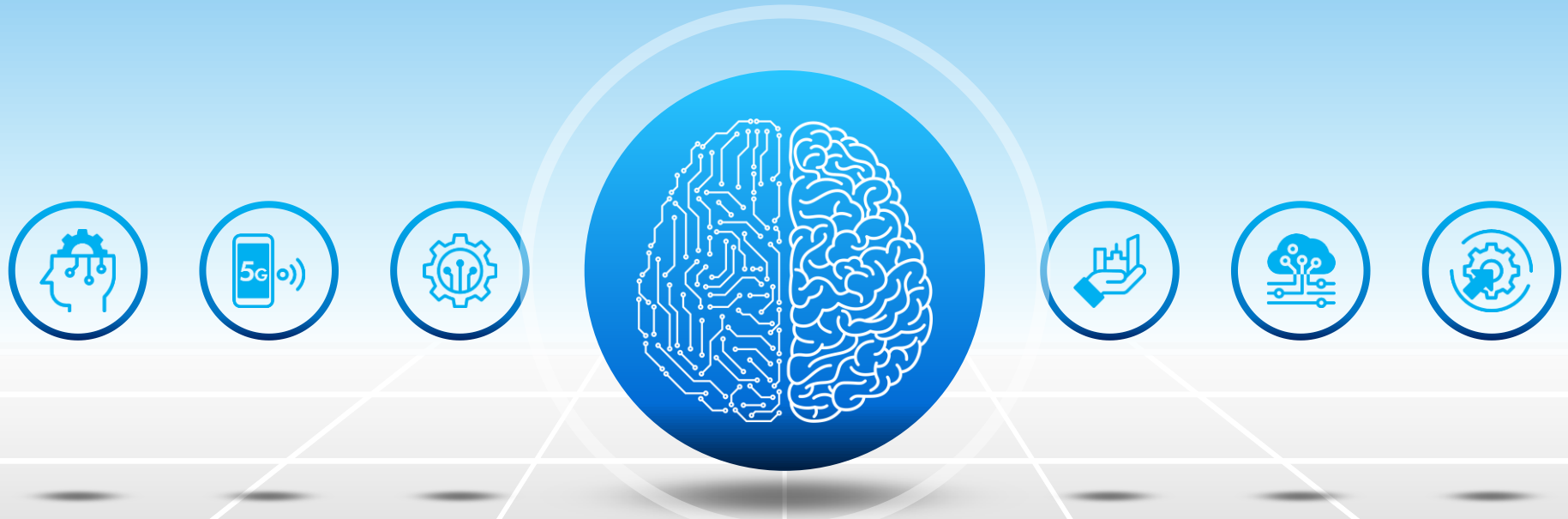
CIS3034 문제해결프로젝트

# 연산량과 시간복잡도



# 목 차
CONTENTS

# 예제 풀이

# 문제2A-도토리 키재기

반복문을 활용하여 문제를 풀어봅시다.

그 과정에서 불필요한 탐색을 제거하여 성능을 개선하는 방법을 고민해봅시다.

실습 내용

도토리 나라의 도토리들이 다니는 도토리 고등학교에서는 매 달 재미있는 이벤트를 한다. 도토리 나라에서는 다른 도토리들 보다 큰 키를 가지는 것이 큰 경쟁력 중 하나였기에, 도토리 고등학교에서는 매 달마다 그 달에 생일이 있는 도토리들 중 가장 큰 키를 가진 도토리에게 상장을 수여한다.

$N$개의 도토리에 대한 키와 생일 정보가 주어진다. 이 때 이번 달에 생일이 있는 도토리들 중 가장 키가 큰 도토리의 키를 출력하는 프로그램을 작성하자.

## 입력 형식

- 첫 줄에는 도토리의 수를 나타내는 $N$이 1이상 10만 이하의 자연수로 주어진다.
- 두 번째 줄에는 $N$개의 도토리의 키가 공백으로 구분된 1이상 10억 이하의 자연수로 주어진다.
  - 도토리들은 현재 키에 대해 오름차순으로 서 있다고 가정한다.
  - 모든 도토리의 키는 서로 다르다. 나노 미터 단위로 측정하였기 때문에 그럴 수 있다.
  - $i$번째에 나오는 숫자가 $i$번째 도토리의 키이다.
- 세 번째 줄에는 $N$개의 도토리들의 생일이 속한 달을 나타내는 1이상 12이하의 자연수가 공백으로 구분되어 주어진다.
  - $i$번째에 나오는 숫자가 $i$번째 도토리의 생일이 속한 달이다.
- 네 번째 줄에는 현재 달을 나타내는 1이상 12이하의 자연수 $M$이 주어진다

## 출력 형식

문제의 답을 공백없이 정수로 출력한다. 생일에 해당 달에 속한 도토리가 없는 경우 `-1`을 출력한다.

# 문제2A-도토리 키재기

입/출력 예시

␣ : 공백 ↲ : 줄바꿈

예시 1

입력
```
5↲
12␣13␣14␣15␣16↲
1␣5␣7␣2␣3↲
2
```

출력
```
15
```

예시 2

입력
```
5↲
12␣13␣14␣15␣16↲
1␣5␣7␣2␣3↲
12
```

출력
```
-1
```

예시 3

입력
```
20↲
10␣11␣12␣13␣14␣15␣16␣17␣18␣19␣20␣21␣22␣23␣24␣25␣26␣27␣28␣29↲
1␣11␣8␣10␣7␣11␣8␣12␣8␣6␣2␣10␣10␣1␣3␣12␣3␣5␣5␣5↲
1
```

출력
```
23
```

# 문제2A-도토리 키재기

```cpp
#include<cstdio>
#include<iostream>

using namespace std;

/**
* 생일이 m월인 가장 큰 키의 도토리를 찾는 함수
* @param height  각 도토리의 키
* @param month   각 도토리의 출생 월
* @param n   도토리의 수
* @param m   찾고자 하는 달
* @return    month[k] == m인 가장 큰 height[k]
*/
int getMaximumHeight(int height[], int month[], int n, int m)
{

}
```

```cpp
int main()
{
  int n, m;
  int* height;
  int* month;

  scanf("%d", &n);
  height = new int[n];
  month = new int[n];

  for (int i = 0; i < n; i++)
  {
    scanf("%d", &height[i]);
  }

  for (int i = 0; i < n; i++)
  {
    scanf("%d", &month[i]);
  }

  scanf("%d", &m);

  int answer = getMaximumHeight(height, month, n, m);

  printf("%d\n", answer);

  delete[] height;
  delete[] month;
  return 0;
}
```

# 문제2B-오름차수인가?

항상 데이터 전체가 특정 조건을 만족할 지 검사해보아야 할 필요는 없습니다. 그 점을 생각해보며 최대한 효율적인 풀이를 생각해봅시다.

실습 내용

그룹을 관리하는 리더라는 직위는 절대 쉬운 것이 아니다. 외국의 한 유명한 여자 대학교인 질레보르(Zylevol)대학교에는 한국인 유학생 수정이와 예인이가 다니고있다. 수정이는 학생회장으로서 학교 행사마다 학생들을 통솔하는 역할을 맡고 있다. 이번 학교 행사에서도 학생회장 수정이는 학생들을 통솔하는 역할을 맡게 되었다. 이번 행사에서는 안무 준비를 위하여 각 학생들은 키가 작은 학생부터 큰 순서대로 일렬로 서야 한다. 하지만 전 날에 먹은 우유의 탓으로 수정이가 몸져 눕게되자, 평소 자신도 학생 회장을 잘 할 수 있다는 자신감에 가득 차 있던 예인이가 대행을 자원했다.

예인이는 이리저리 뛰어다니며 학생들을 일렬로 세웠으나, 이 많은 학생들이 정말로 키 순으로 서 있는지 확신이 서지 않았다. 당신은 예인이를 도울 수 있도록 학생들이 실제로 키에 대하여 오름차순으로 서 있는지 판별해주는 프로그램을 작성해주려고 한다.

## 입력 형식

첫 줄에는 학생의 수 $N$이 주어진다. $N$은 1이상 10만 이하의 자연수이다.

두 번째 줄에는 공백으로 구분된 학생들의 키가 총 $N$개 주어진다. 각 학생의 키는 마이크로미터 단위로 측정하여 1이상 2,000,000 이하의 자연수이다.

## 출력 형식

첫 줄에 학생들의 키가 오름차순으로 정렬되어 있는지 여부를 `YES` 혹은 `NO`로 출력한다.

- `YES` - 각 학생들의 키가 오름차순으로 정렬되어있다.
- `NO` - 그렇지 않다

# 문제2B-오름차순인가?

입/출력 예시

␣ : 공백 ↵ : 줄바꿈

예시 1

입력

```
5↵
1␣2␣3␣5␣4
```

출력

```
NO
```

```
#include<cstdio>
#include<iostream>

using namespace std;

/**
* 주어진 배열이 오름차순인지 검사하는 함수
* @param data
* @param n      데이터의 수
* @return       data[0] ~ data[n-1]이 오름차순이라면 true, else false
*/
bool isOrdered(int data[], int n)
{

}
```

```
int main()
{
  int n;
  int* data;

  scanf("%d", &n);
  data = new int[n];

  for (int i = 0; i < n; i++)
  {
    scanf("%d", &data[i]);
  }

  bool result = isOrdered(data, n);

  if (result)
  {
    printf("YES");
  }
  else{
    printf("NO");
  }

  delete[] data;
  return 0;
}
```

# 문제2C-다양성

정렬된 데이터는 다양한 특성을 가집니다. 정렬된 데이터의 특성을 생각하며 문제를 해결해봅시다.

실습 내용

가수의 앨범은 팬들에게는 중요한 존재이다. 긴 기다림의 끝에 마주친 오아시스 같은 존재이기 때문이다. 그렇기에 최근의 가수들은 자신의 앨범에 기념품이나 손편지 등을 추가하여 팬들에게 고마운 마음을 전하고있다.

인기 많은 아이들 트와이스의 열렬한 팬인 민규는 이번에도 앨범을 사기 위하여 신촌의 한 백화점에 줄을 서서 대기하고있다. 이 백화점에서 판매하는 앨범은 한정판으로서, 각 앨범에는 임의의 화보가 한 장 들어있다. 이미 입대가 얼마남지 않은 민규는 자신의 모든 재산을 털어 트와이스 앨범을 구매하여 모든 종류의 화보를 수집하고자 한다.

각 화보들은 종류별로 고유한 번호를 가지고있으며 이 번호를 통해 같은 화보인지 아닌지 구분할 수 있다. 물론 여러 앨범들에는 같은 화보가 들어있을 수 있다. 민규는 총 $N$개의 앨범을 구매하여 화보들을 번호 순서대로 정렬하여 보관하려고 한다. 이 때 민규는 자신이 총 몇 종류의 화보를 획득하였는지 궁금해졌고, 코딩을 잘하는 당신에게 이를 계산하는 프로그램을 작성해주기를 부탁하였다. 민규의 마지막 소원을 들어주기 위하여 프로그램을 작성하시오.

## 입력 형식

첫 줄에는 화보의 수 $N$이 주어진다. $N$은 1이상 10만 이하의 자연수이다.

두 번째 줄에는 각 화보의 고유번호가 공백으로 구분되어 주어진다. 각 화보의 고유번호는 32비트 정수형 중 하나로 주어지며, $N$개의 고유번호는 오름차순으로 주어진다.

## 출력 형식

한 줄에 중복을 제외한 화보의 종류의 수를 공백없는 자연수로 출력한다.

# 문제2C-다양성

입/출력 예시

␣ : 공백 ↲ : 줄바꿈

예시 1

입력
5↲
1␣2␣3␣4␣5

출력
5

예시 2

입력
5↲
1␣2␣3␣4␣4

출력
4

예시 3

입력
5↲
-1␣-1␣-1␣-1␣-1

출력
1

# 문제2C-다양성

```cpp
#include<cstdio>
#include<iostream>

using namespace std;

/**
* 중복을 제외한 숫자의 종류의 수를 계산하는 함수
* @param data  원본 배열
* @param n     원본 배열의 크기
* @return  숫자의 종류의 수
*/
int getElementTypeCount(int data[], int n)
{
  int countType = 0;

  return countType;
}
```

```cpp
int main()
{
  int n;
  int* data;

  scanf("%d", &n);
  data = new int[n];

  for (int i = 0; i < n; i++)
  {
    scanf("%d", &data[i]);
  }

  int answer = getElementTypeCount(data, n);

  printf("%d\n", answer);

  delete[] data;
  return 0;
}
```

# 문제2D-문자열의 비교

프로그래밍 언어에서는 단어나 문장과 같은 두 글자 이상의 텍스트 데이터를 문자열의 형태로 관리합니다. 문자열이란 각 문자를 배열로 저장한 구조를 말합니다. 아래의 그림이 문자열의 구조를 설명해줍니다.


최근의 언어들은 문자열 또한 string과 같은 기본 타입으로서 제공하기에 많은 분들이 모르고 넘어가는 것이 있습니다. 바로 문자열은 단순히 하나의 값이 아니라 문자 값의 배열이기 때문에 숫자와 같이 단순한 비교가 불가능하다는 점입니다. 이를 해결하기 위하여 해싱과 같은 기법들이 등장하였으나, 원리를 모르고 사용하는 해싱은 독이 될 수 있습니다.

두 문자열을 사전순으로 비교하거나 서로 같은 문자열인지 판별하려면 어떻게 해야할까요? 또 어떻게 비교해야 효율적으로 비교할 수 있을까요? 이런 점들을 고민하며 문제를 풀어봅시다.

실습 내용

두 문자열을 사전순으로 비교하는 함수를 작성해봅시다.

- 언어의 string타입에 내장된 비교 기능을 사용하지 말고 직접 구현해보자.

## 입력 형식

한 줄에 하나씩 공백없이 영어 소문자로 구성된 문자열이 두 개 주어진다. 문자열의 길이는 10만 글자를 넘지 않는다.

## 출력 형식

두 문자열을 사전 순으로 비교한 결과에 따라서 아래와 같이 출력한다.

- 첫 줄에 입력된 문자열이 사전순으로 앞서는 문자열이라면 `-1`을 출력한다.
- 두 번째 줄에 입력된 문자열이 사전으로 앞서는 문자열이라면 `1`을 출력한다.
- 두 문자열이 일치한다면 `0`을 출력한다

# 문제2D-문자열의 비교

입/출력 예시

␣ : 공백 ↲ : 줄바꿈

예시 1

입력
```
algorithm↲
allergy
```

출력
```
-1↲
```

예시 2

입력
```
algorithm↲
algorithm
```

출력
```
0↲
```

예시 3

입력
```
allergy↲
algorithm
```

출력
```
1↲
```

# 문제2D-문자열의 비교

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

const int MAX_LENGTH = 100000;

class MyString{
private:
  char *characters;
  int length;

public:
  MyString(const char * str)
  {
    this->length = strnlen(str, MAX_LENGTH);
    this->characters = new char[this->length];
    for (int i = 0; i < this->length; i += 1)
    {
      this->characters[i] = str[i];
    }
  }

  MyString(const string & original)
  {
    this->length = original.length();
    this->characters = new char[this->length];
    for (int i = 0; i < this->length; i += 1)
    {
      this->characters[i] = original[i];
    }
  }

  ~MyString()
  {
    delete[] characters;
  }
```

```cpp
  /**
   * @param o ..... 비교 할 문자열 (오른쪽 항)
   * @return true .. this가 o보다 사전순으로 앞선다면 true
   * @return false else
   */
  bool operator < (const MyString & o) const{
    int n = min(this->length, o.length);

  }

  /**
   * @param o ..... 비교 할 문자열 (오른쪽 항)
   * @return true .. o가 this보다 사전순으로 앞선다면 true
   * @return false else
   */
  bool operator > (const MyString & o) const{
    int n = min(this->length, o.length);

  }

  /**
   * @param o ..... 비교 할 문자열 (오른쪽 항)
   * @return true .. 두 문자열이 같다면
   * @return false 두 문자열이 다르다면
   */
  bool operator == (const MyString& o) const{

  }

};
```

```cpp
int main()
{
  string s1;
  string s2;
  cin >> s1 >> s2;

  MyString mys1(s1);
  MyString mys2(s2);

  if (mys1 < mys2)
  {
    printf("-1");
  }
  else if (mys1 > mys2)
  {
    printf("1");
  }
  else{
    printf("0");
  }

  return 0;
}
```

# 단원 정리

- **연산량과 시간복잡도 개념 이해하기**
- **실제 연산량을 줄이기 위한 방법 활용하여 문제 해결하기**
  - 도토리 키재기
  - 오름차순인가?
  - 다양성
  - 문자열의 비교