
KAKAO 2018 Blind Recruitment 3rd Exam

해설자료

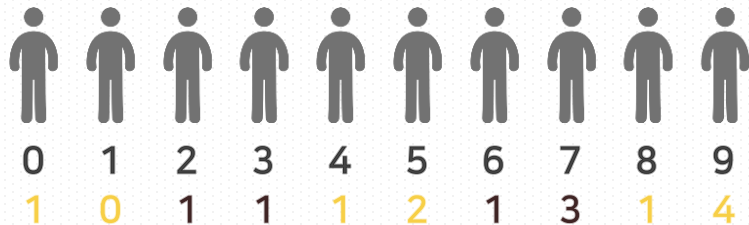
S a m p l e B y K B C

01 N진수 게임

문제 요약

- 여러 사람이 둥글게 앉아서 숫자를 **하나씩 차례대로 말하는 게임**을 진행하려고 한다. 규칙은 다음과 같다.
 - 숫자를 0부터 시작해서 **차례대로 말한다**. 첫 번째 사람은 0, 두 번째 사람은 1, ... 열 번째 사람은 9를 말한다.
 - 10 이상의 숫자부터는 한 자리 씩 끊어서 말한다**. 즉 열한 번째 사람은 10의 첫 자리인 1, 열 두 번째 사람은 둘째 자리인 0을 말한다.

참가자 10명, 10진법인 경우



참가자 10명, 2진법인 경우



- 진법이 주어졌을 때, 한 참가자가 **자신이 말해야 할 숫자를 출력하는** 프로그램을 작성하시오.
- 진법 N ($2 \leq N \leq 16$), 숫자의 개수 T ($0 < T < 1,000$), 게임에 참가하는 인원 M ($2 \leq M \leq 100$), 참가자의 순서 P ($1 \leq P \leq M$) 이 주어진다.

01 N진수 게임

접근

- (게임에 참가하는 인원) * (숫자의 개수) = 100,000 → 모든 경우를 다 따져도 시간은 충분할 것!
- 100,000자 길이의 전체 문자열에서 $P + n * M$ 번째 글자를 추출하는 방식으로 접근하자!

```
function makeBase(number, base) do
    set result = ""
    if number is 0 do
        result = "0"
    end
    while loop (number) {
        result += number % base
        number /= base
    }
    return result
end
```

- 진법이 주어졌을 때, 숫자를 변환하는 것은 나머지를 구하는 방식으로 접근하면 된다.
- 단, 0과 같은 코너케이스에 유의할 것!!!!

01 N진수 게임

접근

```
function makeBase(number, base) do
    set reference = ["0", "1", "2" ... "C", "D", "E"]
    set result = ""
    if number is 0 do
        result = "0"
    end
    while loop (number) {
        result += reference[number % base]
        number /= base
    }
    return result
end
```

```
function solution(n, t, m, p) do
    set result = ""
    set tmp = ""
    set maxSize = m * t
    set number = 0

    while loop(maxSize < size of tmp) do
        set convertNumber = makeBase(number, n)
        tmp += convertNumber
        number++
    end

    for(i -> 0 ... t) do
        result += ith char of tmp
    end
end
```

01 N진수 게임

추가 팁

- 10 ~ 15는 각각 A ~ E로 변환이 된다. → 미리 배열을 만들어서 인덱스로 접근하면 된다!

```
function makeBase(number, base) do
    set reference = ["0", "1", "2" ... "C", "D", "E"]
    set result = ""
    if number is 0 do
        result = "0"
    end
    while loop (number) {
        result += reference[number % base]
        number /= base
    }
    return result
end
```

```
function solution(n, t, m, p) do
    set result = ""
    set tmp = ""
    set maxSize = m * t
    set number = 0

    while loop(maxSize < size of tmp) do
        set convertNumber = makeBase(number, n)
        tmp += convertNumber
        number++
    end

    for(i -> 0 ... t) do
        result += ith char of tmp
    end
end
```

02 압축

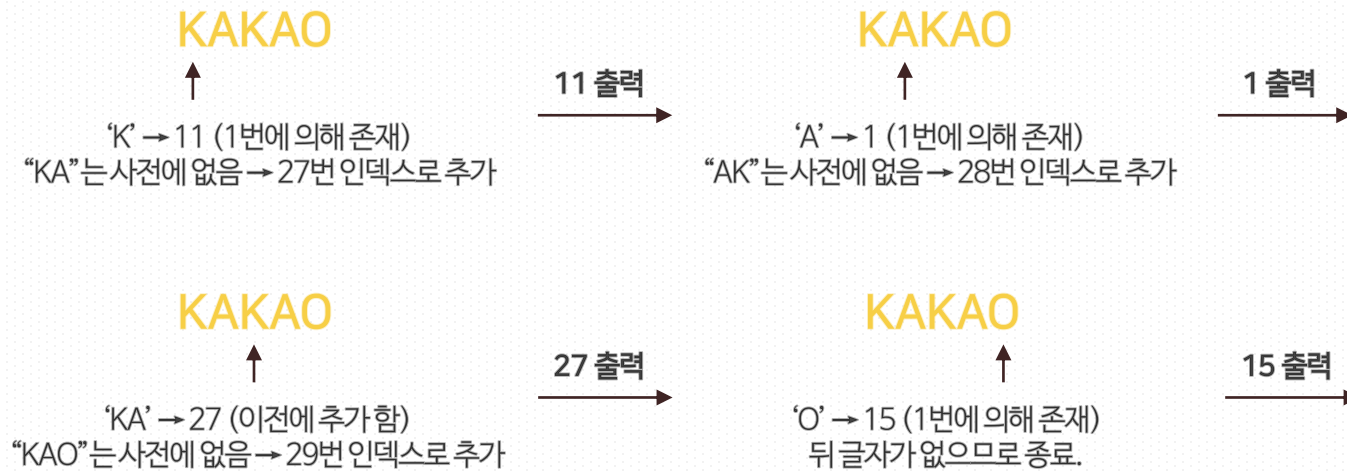


문제 요약

- LZW(Lempel-Ziv-Welch) 압축을 진행하려고 한다. 과정은 다음과 같다.
 1. 길이가 1인 모든 단어를 포함하도록 사전을 초기화한다.
 2. 사전에서 현재 입력과 일치하는 가장 긴 문자열 w 를 찾는다.
 3. W 에 해당하는 사전의 색인 번호를 출력하고, 입력에서 w 를 제거한다.
 4. 입력에서 처리되지 않은 다음 글자가 남아 있다면, (c 라고 하자.) $w + c$ 에 해당하는 단어를 사전에 등록한다.
 5. 2로 돌아간다.
- 압축을 완료했을 때, 해당 문자의 사전 색인 번호를 출력하는 프로그램을 작성시오.
- 해당하는 문자열 MSG ($1 \leq \text{len(MSG)} \leq 1,000$)이 주어진다.

02 압축

문제 요약



02 압축



접근

- 가장 긴 문자열 w 를 찾으면 이후 인덱스는 w 이후로 변경 → 시간복잡도 $O(N)$
- (사전 단어, 인덱스)를 쌍으로 저장할 수 있는 자료구조를 떠올려보자!
C++: `<map>`, `<unordered_map>` (해당 문제는 등장할 수 있는 단어가 적으니 `unordered_map`이 유리함.), Python: `Dict`
- 정해진 절차를 차근차근 따라가기만 하면 충분히 해결할 수 있음!

03 파일명 정렬

문제 요약

- 파일명에 포함된 숫자를 반영한 정렬 기능을 저장소 관리 프로그램에 구현하기로 했다.
- 파일명은 크게 **HEAD, NUMBER, TAIL**의 세 부분으로 구성된다.
 - HEAD는 숫자가 아닌 **문자**로 이루어져 있으며, 최소한 **한 글자 이상**이다.
 - NUMBER는 **1~5 글자** 사이의 **연속된 숫자**로 구성되어 있으며, **앞쪽에 0**이 올 수 있다.
 - TAIL은 **나머지 부분**이며, **0글자 이상**이다.

	HEAD	NUMBER	TAIL
foo9.txt	foo	9	.txt
foo01bar09.zip	foo	01	bar09.zip
bocho1234	bocho	1234	-

- 파일은 HEAD (대, 소문자 구분 없음), NUMBER (앞 0은 정렬 기준이 아님.) 기준으로 정렬되어야 하며, 두 조건이 모두 같을 시에는 **두 파일의 순서가 바뀌면 안된다.**
- 위 기준에 따라 **파일 이름을 정렬**하는 프로그램을 작성하시오.
- 파일명을 포함하는 배열 File (**File의 길이 <= 1,000**)이 주어지는데, 각각의 파일명은 영문 대소문자, 숫자, 공백, 마침표, '-' 부호로 구성된다. **파일명은 중복되지 않는다.**

03 파일명 정렬

접근

- 파일을 HEAD와 NUMBER, TAIL 로 분리해야 한다. → TAIL은 정렬에 영향을 끼치지 않으므로 굳이 분리하지 않아도 됨.
- “두 조건이 모두 같을 시에는 두 파일의 순서가 바뀌면 안된다.” → stable sort 라는 점이 중요함!
 - C++: std::sort 이외에도 stable_sort를 지원하는 **std::stable_sort**가 존재.
 - Python: 내부 정렬 알고리즘인 Timsort가 **stable한 정렬 기법**.

```
function solution(file) do
  set list = []

  for(f <- file) do
    set headAndNumber = makeHeadAndNumber(file)
    list <- (headAndNumber, f)
  end

  do stable sort of list
  return list
end
```

04 방금 그 곡

문제 요약

- 알고있는 멜로디를 주어졌을 때, 어떤 노래인지 출력하는 프로그램을 작성하시오.
- 단, 멜로디가 **반복재생된 노래의 앞/뒤 부분**일 수 있다.
- 주어지는 멜로디와 가능한 노래 목록은 다음 조건을 충족한다.
 - 가능한 노래 목록에선 음악 제목, 시작 시간 및 종료 시간, 멜로디를 제공한다.
 - 사용되는 멜로디는 **C, C#, D, D#, E, F, F#, G, G#, A, A#, B** 12개이다.
 - 각 음은 1분에 1개씩 재생되며, **총 재생시간 > (종료 시간 - 시작 시간)** 일 경우 반복해서 재생된다.
 - 음악이 00:00을 넘겨서 재생되는 경우는 없다.
 - 조건을 일치하는 경우가 여러 개 일 때는 **가장 재생 시간이 긴 음악 → 먼저 입력된 음악 제목** 순으로 출력한다.
 - 일치하는 음악이 없을 때는 “(None)”을 출력한다.
- 기억한 멜로디 M ($1 \leq \text{len}(M) \leq 1,439$)와 곡의 정보를 담은 배열 MusicInfos ($\text{len}(\text{MusicInfos}) \leq 100$)이 주어진다.
 - 음악의 시작 시간 및 종료 시간은 **HH:MM** 형식이다.
 - 악보 정보의 길이 조건은 **M과 동일하다.**

04 방금 그 곡

문제 요약

M	MusicInfos
ABCDEFGG	"12:00,12:14,HELLO,CDEFGAB" "13:00,13:05,WORLD,ABCDEF" CDEFG ABCDEF GAB ABCDE
CC#BCC#BCC#BCC#B	"03:00,03:30,FOO,CC#B" "04:00,04:08,BAR,CC#BCC#BCC#B" CC#BCC#BCC#BCC#B CC#BCC#BCC#BCC#BCC#B... CC#BCC#BCC#

04 방금 그 곡



접근

- 그냥 문자열을 자르고, 스캔 하기엔 #이 조금 까다로움.

CC#BCC#BCC#B

04 방금 그 곡

접근

- 그냥 문자열을 자르고, 스캔 하기엔 #이 조금 까다로움.

CC#BCC#BCC#B

- 어차피 다른 문자들은 대문자이니, #이 붙은 문자를 소문자로 치환하면 엄청나게 쉬워짐!

CcBCcBCcB

- 전체 문자를 해싱하는 것도 좋은 기법.

04 방금 그 곡



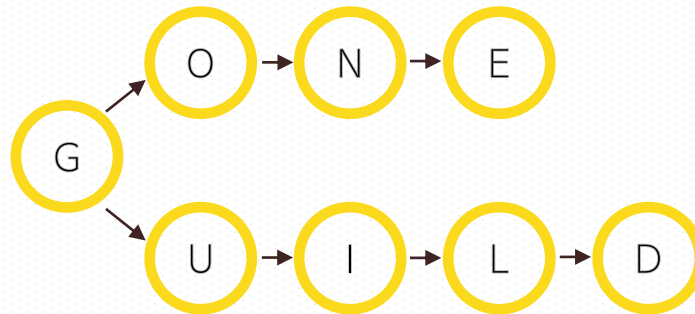
접근

- 일치하는 후보가 많은 경우, **재생 시간과 제목 순서**대로 결정하므로, **정렬은 필수!**
 - 구조체를 잘 정의해서 (**재생 시간, 제목, 멜로디**)로 묶을 수 있는 방법을 고민해보자.
 - 문제에서 필요한 정보만 골라서 묶는 것은 **코테에서 아주 중요한 습관!**

05 자동완성

문제 요약

- 문자열을 학습하여 **자동완성 기능**을 구현하려고 한다.
- 학습에 사용된 단어 중 앞부분이 같은 경우엔 **다른 문자가 나올 때 까지 입력**을 해야 하지만, 다른 문자가 나오면 **한번에 입력** 된다.
- 만약, go/gone/guild가 학습한 단어라고 한다면,



go: 2회
gone: 3회 (go 입력 후 n까지 입력)
guild: 2회 (g 입력 후 u까지 입력)

total: **7회**

- 이때, 단어가 주어지면 모든 문자를 검색할 때 **몇 번 글자를 입력해야 하는지 확인**하는 프로그램을 작성하시오.
- 단어는 **중복 없는** $N(2 \leq N \leq 100,000)$ 개가 주어지며, 단어의 길이의 총합은 $L(2 \leq L \leq 1,000,000)$ 을 넘지 않는다.

05 자동완성

접근


- 이 문제의 핵심은, “다른 문자들과 비교해서 구분되는 부분이 어디부터 인가?”를 찾는 것이다.
- 그런데, 모든 문자들과 비교할 필요가 있을까?

05 자동완성

접근

- 이 문제의 핵심은, “다른 문자들과 비교해서 구분되는 부분이 어디부터 인가?”를 찾는 것이다.
- 그런데, 모든 문자들과 비교할 필요가 있을까?
- 문자의 정렬을 떠올려보면, 앞에서 부터 비교함. → 결국 **정렬 했을 때 앞/뒤에 있는 문자들과 비교**하면 끝!

AING
BING
BOA
BOCHA
BOCHO
BOCHING
ZING



05 자동완성

접근

- 이 문제의 핵심은, “다른 문자들과 비교해서 구분되는 부분이 어디부터 인가?”를 찾는 것이다.
- 그런데, 모든 문자들과 비교할 필요가 있을까?
- 문자의 정렬을 떠올려보면, 앞에서 부터 비교함. → 결국 **정렬 했을 때 앞/뒤에 있는 문자들과 비교**하면 끝!

AING
 BING
 BOA
 BOCHA
 BOCHO
 BOCHING
 ZING

↑
●
↓

```
function solution(words) do
  set result = 0

  for(word <- words) do
    set leftNumber = min(compare(word, leftword) + 1, len(word))
    set rightNumber = min(compare(word, rightword) + 1, len(word))
    result += max(leftNumber, rightNumber)
  end

  return result
end
```

- 물론, 트라이로도 충분히 해결할 수 있는 문제.

06 유사문항

Silver 1 - 줄어드는 숫자

- 진법을 표현해 봅시다.

Silver 3 - 엑셀

- 문자열을 씹고 뜯고 맛봅시다.

Level 3 - 베스트앨범

- 이 문제도 잘 묶는 것이 포인트 입니다. 어떻게 묶어볼까요?

Gold 5 - IPv6

- 이걸 어떤 식으로 끊으면 좋을까요...

Gold 4 - 전화번호 목록

- 트라이를 안 쓴다는 전제하에 유사한 문제입니다. 어떻게 풀이를 적용할 수 있을까요?

Any Questions?