2023 Algorithm Study Week 5 - 과제

cheonwon · 방금 전 · 비공개

2023 Algorithm Study smarcle

사전

알고리즘 시간에 나온 '사전'(사전 ADT)은 key들을 통해 value를 찾는 일종의 dictionary 자료형과 같았으나, 이번 문제의 경우 이름은 사전이지만 알파벳 순으로 사전식 정렬을 한다는 것을 제외하고는 사전ADT를 활용하는 문제는 아닌것 같아 보였다.

사전 <백준 1256>

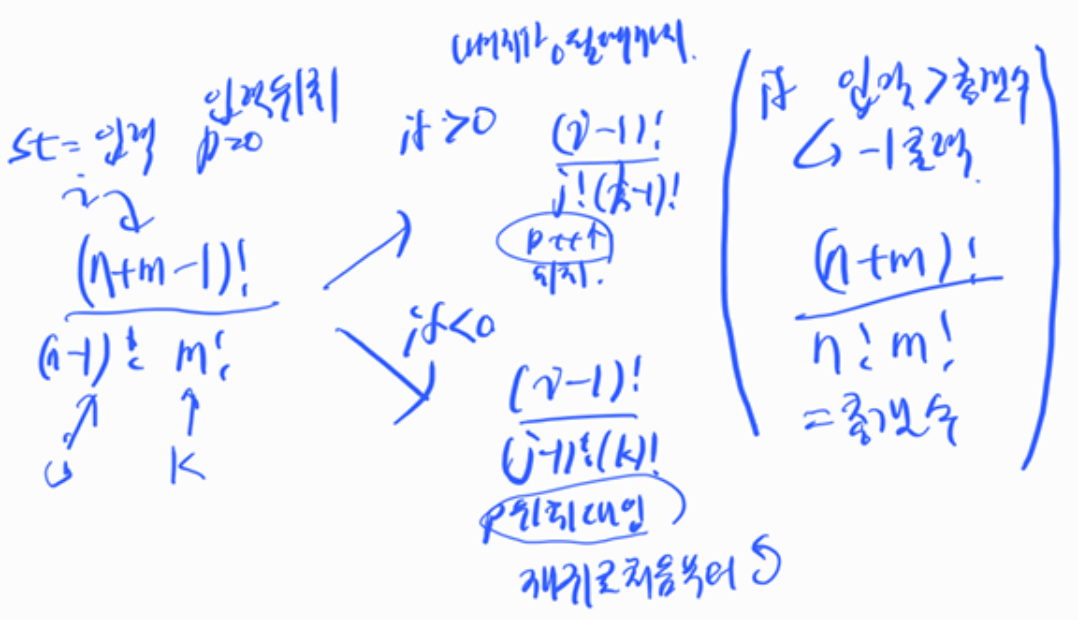
a가 n개, z가 m개 들어간 모든 문자열을 사전식으로 정렬한다고 했을때, k번째의 문자열을 구하는 문제인데... 처음 생각했을때 이 문제의 쟁점은 이 2가지였다.

1. a가 n개, z가 m개 들어간 모든 문자열을 어떻게 만들 것인지.

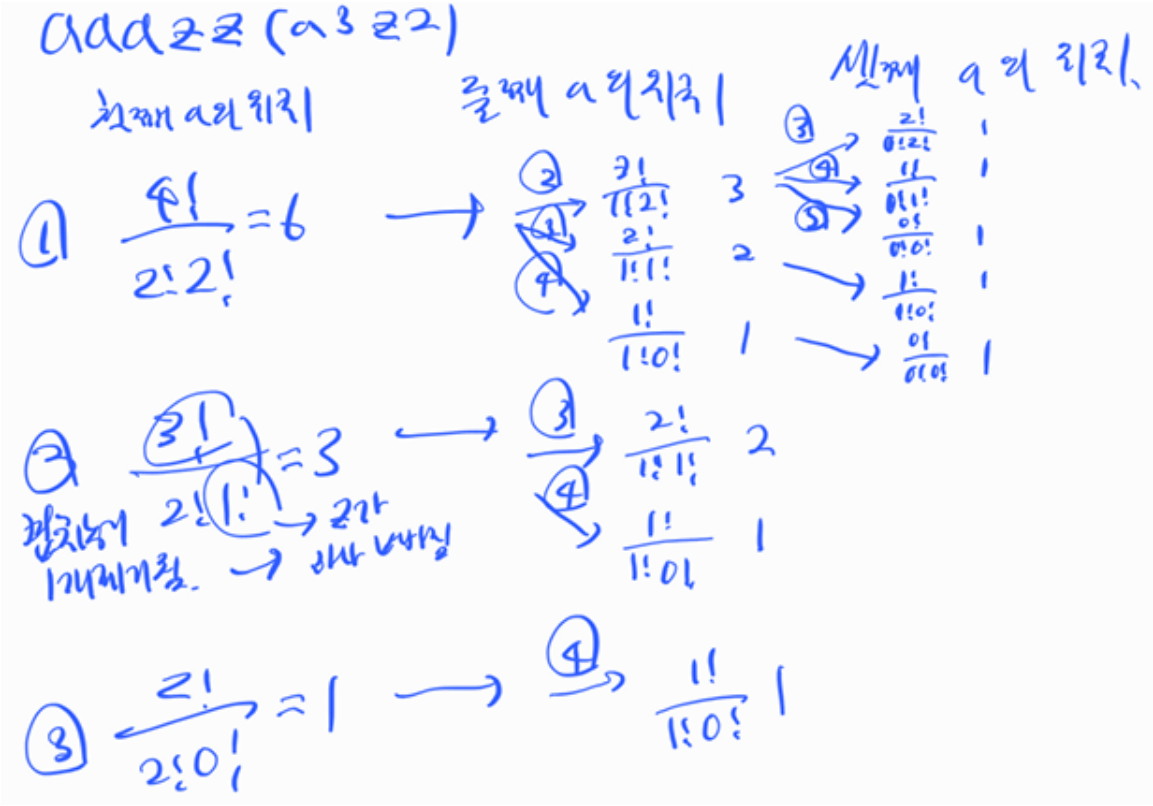
2. 그것들을 어떻게 사전식으로 정렬할 것인지.

이름도 '사전'이라서 인덱스랑 키값을 묶어 정렬하는 사전ADT를 이용한 방식을 생각했던 것이다.
일단 2번은 비교적 쉽게 보였다. 1학년때 했던 것 처럼, 앞에서부터 한 문자씩 비교해서 a가 먼저 나오는 걸 앞으로 스왑하는 방식을 취해서 정렬해도 되고, if의 조건문만 문자열을 서로 비교하는 것으로 바꿔서 퀵정렬이든, 선택정렬이든, 삽입정렬이든, 합병정렬이든 사용해서 정렬을 하면 풀릴것 같았다.
문제는 1번이다. a가 n개, z가 m개 들어간 모든 문자열을 어떻게 만들 것인가. 처음에는 a가 2개, z가 2개인 예시를 보고 2중for문으로 풀려고 했으나, 생각해보니 a가 많아지면 많아질수록 n중, m중 for문이 되기 때문에 풀지 못하게 되며, 부문제를 나눠서 재귀로 풀려고 해도 딱히 잘 알고리즘이 떠올려지질 않았다.
그래서 반대로 생각해보았다. 모든 문자열을 생성한 뒤 정렬하여 k번째의 문자열을 구하는게 아니라 그냥 k번째의 문자열을 직접 구하는 것이다.
아주 옛날 학교 수학시간에 사전식 나열법 문제 풀듯이 푸는 것이다. 첫번째에 a가 오는 경우의 수보다 k가 크면 두번째에 a가 오고 k에서 그만큼의 경우의 수를 제외하고, 두번째에 a가 오는 경우의수 보다

그만큼이 또 크면 세번째에 a가 오는 경우가 되는 방식이다.
a랑 z만 들어가므로 두 문자간의 정렬이 정해져 있으며, 따라서 사전식 나열법에 따르면 첫번째에 a가 오는 경우가 무조건적으로 앞에 오게 되는데, k번째를 구할때 k가 첫번째에 a가 오는 경우의 수보다 크다면 k번째는 무조건 a가 두번째 이후부터 나오게 된다는 방식을 역이용한것이라고 보면 된다.
또한, 첫번째에 a가 오는 경우의 수는 우리가 확통프 시간에 배웠던 것을 응용해 a와 a,z와 z의 차이가 있을때 나열하는 경우의 수 / a간에 나열하는 경우의수 / z간에 나열하는 경우의 수로 구해서 중복되는 값이 있을때 나열하는 경우의 수를 구하는 방법을 사용할 것이다.
이를 그림으로 표현하면 다음과 같다.
<그림 1>



블로그에 올릴용도로 그린 그림이 아니라 내가 그냥 알고리즘 짜다가 그린거라 좀 그림이 더럽다.
aaazz(a3z2)일때를 예시로 한번 그려보았다. 첫째 a가 2에 올때, 3에 올때는 일단 그냥 생략했다.
<그림 2>



근데 결론부터 말하자면 실패했다. 런타임 에러 (IntegerOverflow)가 났기 때문.

```
#pragma warning(disable:4996)
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

long fac(long f) {
    long k = 1;
    for (long i = f; i > 0; i--) {
        k = k * i;
    }
    return k;
}
```

먼저 factorial을 계산하는 코드이다. 경우의수를 계산하기 위해 사용해야하는데, 매번 여러번 입력하기 귀찮아서 사용했다. 재귀문을 이용하는 경우도 있으나 그냥 반복문을 이용해서 구현해봤다.

```
long cal(long i, long j, long k) {
    return fac(i) / fac(j) / fac(k);
}
```

경우의 수를 구하는 방법이다. 전체를 나열하는 방법(n + m)! / 중복되는 갯수(n!, m!)이다.

```
void makech(char ch[], long n, long p, long i, long j, long k) { //p, i, j, k는 반복문 돌리기 위한 변수(
    long tmp = n / (cal(i, j, k) + 1);

    while (tmp > 0) {
```

```
            n = n - cal(i, j, k);
            p++; i--; k--;
            tmp = n / (cal(i, j, k) + 1);
        }
        i--; j--;

        ch[p] = 'a';
        p++;

        if (i < 0 || j < 0|| k < 0) //j < 0만 있어도 됨(총 갯수가 더 큰 경우는 미리 걸러짐)
            return;

        makech(ch, n, p, i, j, k);
    }
```

처음 문자열을 z로 모두 초기화해놓고 a를 넣는 방식으로 코드를 짰다. 이 알고리즘에서 p, i, j, k의 움직임은 <그림 1>과 <그림 2>를 보면 이해할 수 있다. 맨 처음 <그림 1>이 이 알고리즘을 설명해주는 그림인데, <그림 2>를 보면서 일반화하면 <그림 1>이 된다. tmp = n / (cal(i, j, k) + 1)를 한 이유는 그냥 cal(i, j, k)를 하면 딱 그 번째 수가 됐을때 tmp가 1이 되서 반복문이 돌아가버리기 때문이다.(ex - cal = 6일때 n이 6이면 원래는 반복문 안돌아가고 그 자리에 바로 대입을 해야 하는데 반복문이 돌아가서 그 다음자리에 대입되게 되기 때문)
첫번째 a가 끝나면 그 경우의수를 제외하기 위해 n = n - cal(i, j, k);하고 다시 tmp = n / (cal(i, j, k) + 1); 를 갱신했다.

```
int main() {

    long n, m, k;
    scanf("%ld%ld%ld", &n, &m, &k);

    char *ch = (char*)malloc(sizeof(char)*(n + m) + 1);

    for (long i = 0; i < n + m; i++) {
        ch[i] = 'z';
    }
    ch[n + m] = '\0';

    if (k <= cal(n + m, n, m)) {
        makech(ch, k, 0, n + m - 1, n - 1, m);
        printf("%s", ch);
    }
    else {
        printf("-1");
    }

    free(ch);

    return 0;
}
```

입출력만 맞춰주는 main함수이다. makech를 호출할때 인자들은 <그림 1>에서 i, j, k를 초기화 하는 걸 보면 이해할 수 있다.

<제출 화면>

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이	제출한 시간
67737580	cheonwon	1256	런타임 에러 (IntegerOverflow)			C99	999 B	2시간 전
67737284	cheonwon	1256	런타임 에러 (IntegerOverflow)			C99	980 B	2시간 전

찾아보니 런타임 에러 (IntegerOverflow)가 정수가 저장할 수 있는 가장 큰 값보다 더 큰 값, 또는 가장 작은 값보다 더 작은 값을 저장하려고 할 때 발생한다고 해서 혹시 몰라 int를 long으로 바꿔봤는데도 문제가 계속 발생했다.
일단 해결방안을 계속해서 찾아봐야 겠다.

천승원

뭐든지 한걸음씩

이전 포스트
2023 Algorithm Study Week 4 - 과제

0개의 댓글

댓글을 작성하세요

댓글 작성


Powered by
Stellate