알고리즘 스터디 풀이 - 1회차

권민준

9440번 - 숫자 더하기 풀이

먼저 생각해야할 것은 가장 큰 숫자가 가장 오른쪽에 오고, 가장 작은 숫자일수록 왼쪽에 있는 것이 이상적입니다.

그러면 여기서 2개의 숫자를 만들 배열에 주어준 숫자들을 큰 수부터 번갈아 넣으면 된다고 생각할 수 있습니다.

예시로 준 1, 2, 7, 8, 9 를 예시로 들어보겠습니다.

1. 9가 가장 크므로 1번 배열에 9를 넣습니다.

   1번 배열 : [9], 2번 배열 : []

2. 그다음 8이 크므로 2번 배열에 8을 넣습니다.

   1번 배열 : [9], 2번 배열 : [8]

3. 같은 순서로 계속 넣으면 아래와 같이 됩니다.

   1번 배열 : [1, 7, 9], 2번 배열 : [2, 8]

4. 이 두 배열을 숫자로 표현해서 더해보면 **179 + 28 = 207** 이라는 수가 가장 작다는 것을 알 수 있습니다.

여기까지만 하면 상당히 쉬운 문제지만..

입력값으로 0이 들어오지만 불행하게도 0은 맨 오른쪽에 위치할 수가 없다고 합니다.

그러면 저렇게 배열을 만들고 나서 0을 적절히 배치해서 넣어줘야합니다.

각 배열에 0을 몇개 넣을지가 상당히 중요합니다. 그러려면 **기준을 명확히 정해야합니다.**

1. 숫자가 적게 들어간 배열부터 0을 넣어줍니다.
2. 두 배열에 들어간 숫자들의 개수가 동일할 때, 각 배열의 가장 작은 값을 비교해서 더 작은 값이 있는 배열에 0을 넣어줍니다.

이 두 가지 기준을 적용해서 코드를 작성하면 풀 수 있습니다 :)

13718번 - Taven 풀이

문제는 간단합니다.

문자열에 #이 M개 섞여있고, #에 들어갈 알파벳 후보들을 K개가 주어진다고 합니다. 모든 후보들이 들어갈 수 있는 경우의 수를 다 따져서 사전순으로 정렬했을 때 X번째에 있는 경우의 수가 정답이라고 합니다.

시간제한이 1초에 메모리 제한은 32MB 입니다. 즉, 모든 경우의 수를 일일이 계산하고 앉아있으면 100% Time Out 뜹니다.

그러면 우리는 이제 알고리즘을 간소화시킬 필요가 있다는 것을 알았습니다.

문제를 살펴보면 우리가 필요한 것은 #에 들어갈 문자들입니다. 모든 경우의 수를 사전순으로 정렬한다고 하면 분명 알파벳 후보들이 일종의 순서로 순회하는 형태를 띌 것입니다.

이것을 힌트로 우리는 **K진법 계산** 을 통해 어떤 문자가 들어가는지 단번에 알 수 있습니다.

예시로 주어준 테스트케이스를 봅시다:

```
9 2 3 7
po#olje#i
sol
znu
```

알파벳 후보들을 사전순 정렬해보면 **['l', 'o', 's'], ['n', 'u', 'z']** 입니다. **X는 7**이구요.

알파벳 후보들이 들어갈 수 있는 경우의 수를 전부 따져보면 다음과 같습니다.

pololjeni, pololjeui, pololjezi, poooljeni, poooljeui, poooljezi, **posoljeni**, posoljeui, posoljezi

후보군에서 **s**와 **n**이 선택되는데 각각은 각 배열의 **3번째, 1번째**입니다.

K진법으로 X를 변환했을 때 각 자리수가 정렬된 후보군을 가리키는 index 임을 알 수 있습니다.

이 규칙을 도출해냈다면 통해서 쉽게 풀수가 있습니다. 다음은 이를 구체화한 python 코드입니다.

```python
N, M, K, X = map(int, input().split())
L = input()
candidates = []
for _ in range(M):
    candidates.append(sorted(input()))

X = X - 1
result = ''
for i in range(M):
    result += candidates[M-i-1][X // K ** i % K]
for x in result[::-1]:
    L = L.replace('#', x, 1)
print(L)
```