1. N과 M(11)

**문제**

N개의 자연수와 자연수 M이 주어졌을 때, 아래 조건을 만족하는 길이가 M인 수열을 모두 구하는 프로그램을 작성하시오.

* N개의 자연수 중에서 M개를 고른 수열
* 같은 수를 여러 번 골라도 된다.

**입력**

첫째 줄에 N과 M이 주어진다. (1 ≤ M ≤ N ≤ 7)

둘째 줄에 N개의 수가 주어진다. 입력으로 주어지는 수는 10,000보다 작거나 같은 자연수이다.

**출력**

한 줄에 하나씩 문제의 조건을 만족하는 수열을 출력한다. 중복되는 수열을 여러 번 출력하면 안되며, 각 수열은 공백으로 구분해서 출력해야 한다.

수열은 사전 순으로 증가하는 순서로 출력해야 한다.

**풀이**

N개의 수에서 M개의 수를 고르는 문제지만 특이한 조건이 있다.

1. 같은 수를 여러 번 골라도 된다.
2. 중복된 수열을 여러 번 출력하면 안된다.
3. 사전 순으로 출력해야 한다.

3번 조건은 N개의 수를 정렬한 뒤, M개의 수를 뽑을 때, 1개의 수를 뽑을 때 마다 앞에서부터 순서대로 수를 뽑으면 된다.

즉, M=2 일 때, N개의 수 A(N) = [4,4,5]라 했을 때, 1번 조건을 만족하는 조합 C에 대하여 C에 존재한 조합의 원소들에 대응되는 A의 원소의 인덱스를 모두 나타내면

[1,1], [1,2], [1,3], [2,1], [2,2], [2,3], [3,1], [3,2], [3,3]

이 된다. 이를 원소의 값으로 표현하면

[4,4], [4,4], [4,5], [4,4], [4,4], [4,5], [5,4], [5,4], [5,5]

이 된다. 이 상황에서 중복된 수열을 뽑지 않는 방법은 뭐가 있을까?

1. 최종 상태를 저장한 뒤, 중복된 상태를 제거하기

가장 무식하면 방법이지만 확실한 방법이다. 그러나 나올 수 있는 경우의 수는 N­M 이다. 이를 모두 저장하려면 최대 77 가지 경우를 모두 저장해야 한다. 이는 220 언저리로 매우 많은 공간을 사용하여 좋지 못한 방법이 된다. 이를 보안하기 위해 다음과 같은 방법을 사용하자.

2. 중복된 조합이 나오는 경우를 모두 배제

만약 위의 예시를 모두 탐색하는 경우를 생각해보자. 먼저, A의 첫 번째 원소를 뽑는다. 그 다음 구간을 탐색해보면 [4,4,5] 를 삽입해야 하는데 4가 두 번 존재한다. 이는 같은 조합이 나올 수 있음을 의미한다. 따라서 우리는 두 번의 4중 하나만 선택하여 조합을 만들어야 한다. 그리하여 4,5를 한 번씩 뽑게 되면 [4,4], [4,5]라는 조합이 만들어진다. 이제 조합의 첫 원소를 뽑는 과정으로 돌아가면 마지막 원소를 뽑았을 때와 같은 상황이 발생한다. A의 첫 원소를 뽑은 뒤, A의 두 번째 원소는 첫 원소인 4와 같다. 그렇다면 다음 원소들을 뽑을 때, 처음 조합을 뽑을 때와 같은 상황이 나온다. 따라서, 이 경우는 탐색을 진행하지 않는다. 이를 모든 원소에 대해서 반복하게 된다면 뽑는 조합이 중복되지 않음을 보장된다. 따라서 K번째 원소를 뽑을 때, 이미 뽑은 원소가 아닌 원소를 뽑으면 됨을 의미한다. (K는 M보다 작거나 같은 자연수)

2. 연산자 끼워 넣기

## 문제

N개의 수로 이루어진 수열 A1, A2, ..., AN이 주어진다. 또, 수와 수 사이에 끼워 넣을 수 있는 연산자가 주어진다. 연산자는 덧셈(+), 뺄셈(-), 곱셈(×), 나눗셈(÷)으로만 이루어져 있다. 연산자의 개수는 N-1보다 많을 수도 있다. 모든 수의 사이에는 연산자를 한 개 끼워 넣어야 하며, 주어진 연산자를 모두 사용하지 않고 모든 수의 사이에 연산자를 끼워 넣을 수도 있다.

우리는 수와 수 사이에 연산자를 하나씩 넣어서, 수식을 하나 만들 수 있다. 이때, 주어진 수의 순서를 바꾸면 안 된다.

예를 들어, 6개의 수로 이루어진 수열이 1, 2, 3, 4, 5, 6이고, 주어진 연산자가 덧셈(+) 3개, 뺄셈(-) 2개, 곱셈(×) 1개, 나눗셈(÷) 1개인 경우에는 총 250가지의 식을 만들 수 있다. 예를 들어, 아래와 같은 식을 만들 수 있다.

* 1+2+3-4×5÷6
* 1÷2+3+4-5×6
* 1+2÷3×4-5+6
* 1÷2×3-4+5+6
* 1+2+3+4-5-6
* 1+2+3-4-5×6

식의 계산은 연산자 우선 순위를 무시하고 앞에서부터 진행해야 한다. 또, 나눗셈은 정수 나눗셈으로 몫만 취한다. 음수를 양수로 나눌 때는 C++14의 기준을 따른다. 즉, 양수로 바꾼 뒤 몫을 취하고, 그 몫을 음수로 바꾼 것과 같다. 이에 따라서, 위의 식 4개의 결과를 계산해보면 아래와 같다.

* 1+2+3-4×5÷6 = 1
* 1÷2+3+4-5×6 = 12
* 1+2÷3×4-5+6 = 5
* 1÷2×3-4+5+6 = 7
* 1+2+3+4-5-6 = -1
* 1+2+3-4-5×6 = -18

N개의 수와 연산자가 주어졌을 때, 만들 수 있는 식의 결과가 최대인 것과 최소인 것을 구하는 프로그램을 작성하시오.

## 입력

첫째 줄에 수의 개수 N(2 ≤ N ≤ 11)가 주어진다. 둘째 줄에는 A1, A2, ..., AN이 주어진다. (1 ≤ Ai ≤ 100) 셋째 줄에는 합이 N-1보다 크거나 같고, 4N보다 작거나 같은 4개의 정수가 주어지는데, 차례대로 덧셈(+)의 개수, 뺄셈(-)의 개수, 곱셈(×)의 개수, 나눗셈(÷)의 개수이다.

## 출력

첫째 줄에 만들 수 있는 식의 결과의 최댓값을, 둘째 줄에는 최솟값을 출력한다. 연산자를 어떻게 끼워넣어도 항상 -10억보다 크거나 같고, 10억보다 작거나 같은 결과가 나오는 입력만 주어진다. 또한, 앞에서부터 계산했을 때, 중간에 계산되는 식의 결과도 항상 -10억보다 크거나 같고, 10억보다 작거나 같다.

**풀이**

단순한 백트랙킹 문제이다. 맨 처음 숫자부터 시작하여, 해당 단계에서 사용 가능한 연산자를 사용하며 해당 연산자의 사용 횟수를 늘려가며 다음 숫자와 연산을 한 뒤, 다음 단계로 넘어가는 과정을 모두 진행하면 된다. 마지막 숫자까지 연산을 완료했다면 최댓값, 최솟값을 계속 바꿔주면 된다.

## 3. 감소하는 수

## 문제

음이 아닌 정수 X의 자릿수가 가장 큰 자릿수부터 작은 자릿수까지 감소한다면, 그 수를 감소하는 수라고 한다. 예를 들어, 321과 950은 감소하는 수지만, 322와 958은 아니다. N번째 감소하는 수를 출력하는 프로그램을 작성하시오. 0은 0번째 감소하는 수이고, 1은 1번째 감소하는 수이다. 만약 N번째 감소하는 수가 없다면 -1을 출력한다.

## 입력

첫째 줄에 N이 주어진다. N은 1,000,000보다 작거나 같은 자연수 또는 0이다.

## 출력

첫째 줄에 N번째 감소하는 수를 출력한다.

**풀이**

먼저 모든 감소하는 수를 찾기 위해선 어떻게 해야 할까? 현재 수의 1의 자리 숫자를 k라 하자. 그렇다면 k보다 작은 숫자를 현재 수보다 뒤에 붙이면 하나의 감소한 수가 된다. 또한 그 수에서 위의 과정을 반복하게 된다면 결국 모든 감소하는 수를 찾을 수 있다. 따라서 이 모든 경우를 탐색하여 저장한 뒤, 정렬을 하면 i번째 (i는 음이 아닌 정수) 감소하는 수를 찾을 수 있다. 감소하는 수는 총 1023개로 i가 1023이상이라면 존재하지 않는 수이므로 n이 1023 이상이면 -1을 출력하면 된다.

|  |
| --- |
|  |
|  |

4. 도도의 음식준비

|  |
| --- |
| 도도는 주방장이다. 총K개의 요리가 준비되는 최소 시간을 구해야 한다. 각각의 요리사는 자신만의 음식 조리 시간이 있다. 음식 조리 시간은 음식 하나를 만들 때 걸리는 시간이다. 도도는 요리사에게 격려를 해줄 수 있다. 격려 받은 요리사는 영구적으로 음식 조리 시간이 1초 감소한다. 도도는 한 요리사에게 여러 번 격려할 수 있고, 요리사의 음식 조리 시간을 1초 미만으로 줄일 수는 없다. 도도를 위해 요리에 걸리는 최소 시간을 출력하는 프로그램을 만들어 보자. |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **입력** |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 첫째 줄에 요리사의 수N (1<=N<=10), 만들어야 할 음식의 개수 K(1<=K<=1000000), 격려해줄 수 있는 횟수 C (0<=C<=5)가 주어진다. |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| 둘째 줄에 길이가 N인 정수 수열 A가 주어진다. i번째로 주어지는 수 Ai는 i번째 요리사의 음식 조리 시간이다. (1<=i<=N,1<=Ai<=1000000) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| **출력** |
|  |

|  |
| --- |
|  |
|  |

첫째 줄에 K개의 음식 조리가 완료되는 최소 시간을 출력한다.

**풀이**

백트랙킹 + 이분탐색 문제이다.

문제를 두 문제로 바꾸자.

* 1. C번의 격려를 누구에게 할 지
  2. K개의 요리를 만드는데 걸리는 시간

2번의 과정은 K개의 요리를 만드는 최소 시간을 T라 하면 t<T인 경우엔 K개의 요리를 절대 만들 수 없다. 반대의 경우엔 반드시 K개 이상의 요리를 만들 수 있으므로 K개의 요리를 만드는 시간에 대해서 이분 탐색을 진행하면 최소시간을 T를 구할 수 있다.

1번의 과정은 C번의 격려가 가능한 상황에서 격려가 가능한 사람에게 1번 격려를 해주고 해당 요리사들의 상태에서 C-1번 격려 가능한 상황으로 넘어가며 다시 격려 가능한 사람 중 한 명을 격려해준다. 과정을 격려할 수 있는 모든 경우에 대하여 탐색을 한다. 이렇게 탐색을 진행하며 더 이상으로 격려할 수 없는 상황이 된다면 2번 과정을 수행하여 최소 시간 T를 갱신한다.

5. 신기한 소수

## 문제

수빈이가 세상에서 가장 좋아하는 것은 소수이고, 취미는 소수를 가지고 노는 것이다. 요즘 수빈이가 가장 관심있어 하는 소수는 7331이다.

7331은 소수인데, 신기하게도 733도 소수이고, 73도 소수이고, 7도 소수이다. 즉, 왼쪽부터 1자리, 2자리, 3자리, 4자리 수 모두 소수이다! 수빈이는 이런 숫자를 신기한 소수라고 이름 붙였다.

수빈이는 N자리의 숫자 중에서 어떤 수들이 신기한 소수인지 궁금해졌다. N이 주어졌을 때, 수빈이를 위해 N자리 신기한 소수를 모두 찾아보자.

## 입력

첫째 줄에 N(1 ≤ N ≤ 8)이 주어진다.

## 출력

N자리 수 중에서 신기한 소수를 오름차순으로 정렬해서 한 줄에 하나씩 출력한다.

풀이

1. 완전 탐색

N자리 수 중에서 주어진 조건을 만족하는 경우를 찾는 것이라면 탐색할 수의 개수는 9N이고 조건에 맞는지 확인하는데 각 (1~걸리는 시간은 이다. 따라서 총 시간 복잡도는 으로 굉장히 오랜 시간이 걸린다.

2. 분기 한정

현재 K 자리 수에서 뒤에 숫자를 삽입한다고 생각해보자. (K는 N보다 작은 음이 아닌 정수) 우리는 최종 조건에 맞는 수를 찾기 위해선 K자리 수에 숫자를 삽입할 때, 소수가 되지 않는다면 우리는 해당 숫자를 삽입한 이후의 수들에 대해선 탐색을 할 필요가 없다. 즉, 매번 숫자를 삽입할 때 마다, 해당 수가 소수면 탐색을 더 진행하고 아니면 해당 수에 대해선 더 이상 탐색하지 않는 방법으로 진행하면 조건을 만족하는 N자리 소수를 모두 구할 수 있다.

따라서 탐색이 가능한 수 , 각 케이스 마다 한 번만 소수 판정을 진행하면 되므로

소수 판정 알고리즘의 시간 복잡도는 이므로 총 시간 복잡도는 으로 여전히 크긴 하지만 분기 한정 조건에 의해 2,4,6,8과 같은 짝수는 반드시 빠지게 되므로 상당히 줄어든다.