김다인

알고리즘 스터디 4주

목차

- 투포인터 알고리즘의 이해
- 어떤 문제가 투포인터로 풀 수 있는 문제인가?
- 슬라이딩 윈도우와 차이
- 투포인터 문제 풀이
- 슬라이딩 윈도우 문제 풀이
- 숙제!

투포인터 알고리즘의 이해

- 리스트에 순차적으로 접근해야 할 때, 두 포인터의 위치를 기록하며 처리하는 알고리즘
- 병합 정렬(merge sort)과 퀵정렬(quick sort)에서 이 투 포인터를 사용하기도 한다 (low, high)
- 주로 숫자 리스트가 주어지고, 연속합이 가장 큰 구간을 고르라거나 수의 합 등이 최저/최고가 되도록 하라는 문제에서 쓸 수 있다
- 정렬을 해서 풀 수 있는 문제가 있고 정렬을 못 하는 문제가 있으므로 두 경우에 모두 대비하는 것이 현명하다
- 투 포인터 문제는 두 포인터가 처음과 끝을 가리키는 경우 그리고 한 쪽에서 두 개의 포인터가 이동하는 경우가 있음
- 시간복잡도는 O(N)
- https://butter-shower.tistory.com/226

병합 정렬에서의 투포인터 사용

- L, h라는 변수를 두고 비교하면서 두 포인터를 움직여 병합에 이용한다
- https://www.daleseo.com/sort-merge/

```python
def merge_sort(arr):
    if len(arr) < 2:
        return arr

    mid = len(arr) // 2
    low_arr = merge_sort(arr[:mid])
    high_arr = merge_sort(arr[mid:])

    merged_arr = []
    l = h = 0
    while l < len(low_arr) and h < len(high_arr):
        if low_arr[l] < high_arr[h]:
            merged_arr.append(low_arr[l])
            l += 1
        else:
            merged_arr.append(high_arr[h])
            h += 1
    merged_arr += low_arr[l:]
    merged_arr += high_arr[h:]
    return merged_arr
```

어떤 문제가 투포인터로 풀 수 있는 문제인가?

- 2470 두 용액 (정렬해도 되는 문제)

두 용액

성공 스페셜 저지

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초 (추가 시간 없음)	128 MB	23082	7002	5153	30.582%

문제

KOI 부설 과학연구소에서는 많은 종류의 산성 용액과 알칼리성 용액을 보유하고 있다. 각 용액에는 그 용액의 특성을 나타내는 하나의 정수가 주어져있다. 산성 용액의 특성값은 1부터 1,000,000,000까지의 양의 정수로 나타내고, 알칼리성 용액의 특성값은 -1부터 -1,000,000,000까지의 음의 정수로 나타낸다.

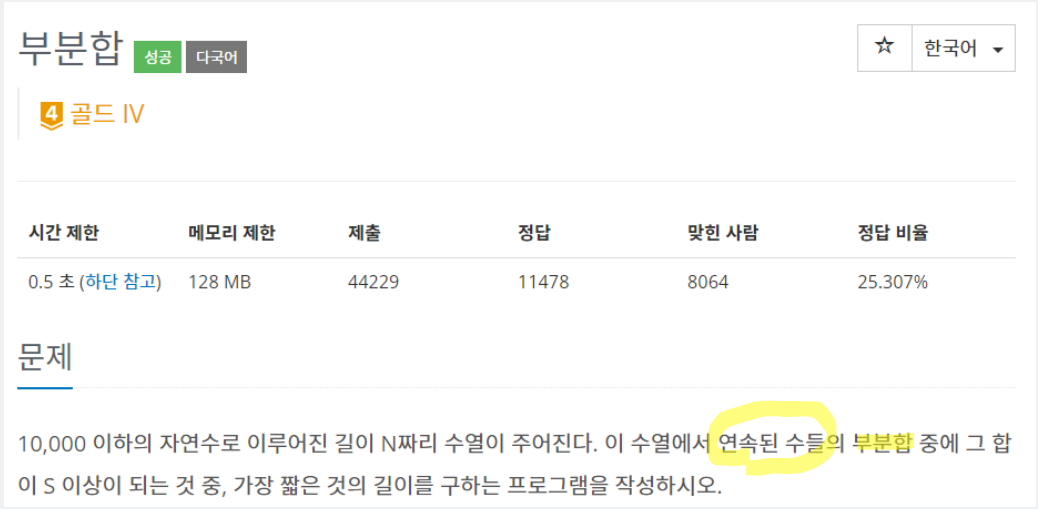
같은 양의 두 용액을 혼합한 용액의 특성값은 혼합에 사용된 각 용액의 특성값의 합으로 정의한다. 이 연구소에서는 같은 양의 두 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들려고 한다.

예를 들어, 주어진 용액들의 특성값이 [-2, 4, -99, -1, 98]인 경우에는 특성값이 -99인 용액과 특성값이 98인 용액을 혼합하면 특성값이 -1인 용액을 만들 수 있고, 이 용액이 특성값이 0에 가장 가까운 용액이다. 참고로, 두 종류의 알칼리성 용액만으로나 혹은 두 종류의 산성 용액만으로 특성값이 0에 가장 가까운 혼합 용액을 만드는 경우도 존재할 수 있다.

산성 용액과 알칼리성 용액의 특성값이 주어졌을 때, 이 중 두 개의 서로 다른 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들어내는 두 용액을 찾는 프로그램을 작성하시오.

어떤 문제가 투포인터로 풀 수 있는 문제인가?

- 1806 부분합 (정렬을 못 하는 문제)
- 사실 저는 이 문제를 포인터 1개로 풀긴 했지만 대표적 투포인터 문제 중 하나입니다



부분합 성공 다국어

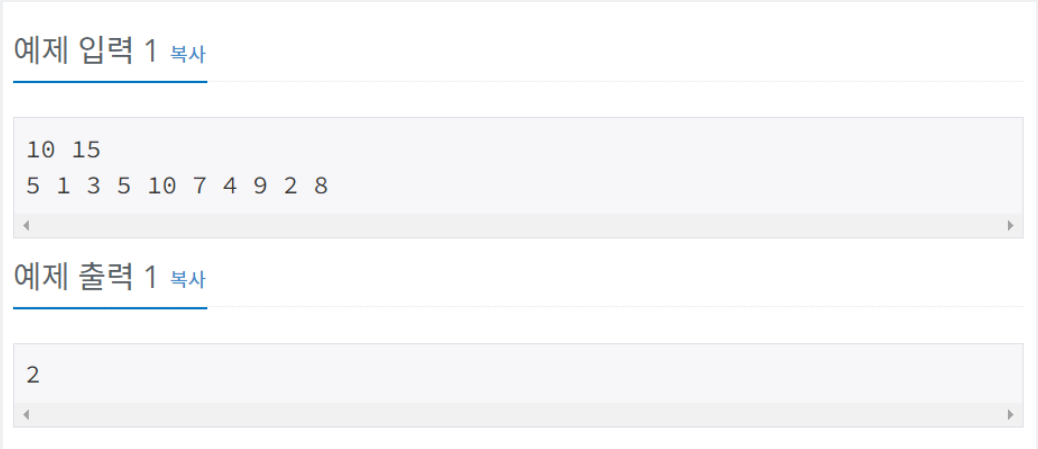
☆ 한국어

골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.5 초 (하단 참고)	128 MB	44229	11478	8064	25.307%

문제

10,000 이하의 자연수로 이루어진 길이 N짜리 수열이 주어진다. 이 수열에서 연속된 수들의 부분합 중에 그 합이 S 이상이 되는 것 중, 가장 짧은 것의 길이를 구하는 프로그램을 작성하시오.



예제 입력 1 복사

```
10 15
5 1 3 5 10 7 4 9 2 8
```

예제 출력 1 복사

```
2
```

슬라이딩 윈도우와 차이

- 투포인터와 유사하지만, 투포인터는 포인터의 이동 범위가 리스트 전체에 걸쳐 있다면 슬라이딩 윈도우는 특정 범위(고정적인 크기)를 제한해 이동한다는 특징이 있다고 함
- 투포인터는 부분 배열의 길이가 가변적이지만, 슬라이딩 윈도우는 부분 배열의 길이가 고정
- 근데 정확히 뭔 말인지는 모르겠고 이것도 예제 문제를 풀면서 틀려보고 모범 코드를 봐야 이해가 될 듯합니다
- 처음과 끝에 대한 삽입, 삭제가 빈번하기 때문에 덱이나 힙 등을 사용한다고 하지만 항상 그런 건 아닌 듯합니다 (투 포인터는 주로 그냥 인덱스를 가리키는 정수형 변수를 사용할 때가 많음)
  - 크기를 고정할 방법은, 사이즈를 체크하면 됨
  - https://www.youtube.com/watch?v=uH9VJRIpIDY

두 용액

- 백준 2470
- 정렬할 수 있는 문제를 풀어봅시다
- 투포인터 클래식 문제!

두 용액 성공 스페셜 저지

☆

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초 (추가 시간 없음)	128 MB	23082	7002	5153	30.582%

문제

KOI 부설 과학연구소에서는 많은 종류의 산성 용액과 알칼리성 용액을 보유하고 있다. 각 용액에는 그 용액의 특성을 나타내는 하나의 정수가 주어져있다. 산성 용액의 특성값은 1부터 1,000,000,000까지의 양의 정수로 나타내고, 알칼리성 용액의 특성값은 -1부터 -1,000,000,000까지의 음의 정수로 나타낸다.

같은 양의 두 용액을 혼합한 용액의 특성값은 혼합에 사용된 각 용액의 특성값의 합으로 정의한다. 이 연구소에서는 같은 양의 두 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들려고 한다.

예를 들어, 주어진 용액들의 특성값이 [-2, 4, -99, -1, 98]인 경우에는 특성값이 -99인 용액과 특성값이 98인 용액을 혼합하면 특성값이 -1인 용액을 만들 수 있고, 이 용액이 특성값이 0에 가장 가까운 용액이다. 참고로, 두 종류의 알칼리성 용액만으로나 혹은 두 종류의 산성 용액만으로 특성값이 0에 가장 가까운 혼합 용액을 만드는 경우도 존재할 수 있다.

산성 용액과 알칼리성 용액의 특성값이 주어졌을 때, 이 중 두 개의 서로 다른 용액을 혼합하여 특성값이 0에 가장 가까운 용액을 만들어내는 두 용액을 찾는 프로그램을 작성하시오.

예제 입력 1 복사

```
5
-2 4 -99 -1 98
```

예제 출력 1 복사

```
-99 98
```

해야 할 일

- (1) 먼저 주어지는 용액을 전부 받고 정렬합니다
- (2) **연속된 용액** 중에서 0에 가장 가까운 두 개를 고르라고 한 것이 아니라 그냥 **전체 용액들 중 0에 가장 가까운 두 개**를 고르라고 한 것이기 때문에, 정렬이 가능합니다.
- 정렬합니다.
- (3) 포인터 2개를 선언해 용액 리스트의 **처음과 끝 인덱스**를 가리키도록 합니다
- (4) 이 두 인덱스가 가리키는 용액의 합을 구해줍니다.
  - (만약 이게 0보다 크다면 오른쪽 포인터를, 0보다 작다면 왼쪽 포인터를 이동하면 될 것입니다)

```
n = int(input())
resolves = list(map(int, input().split()))
resolves.sort()

l = 0
r = n-1
attr = resolves[l] + resolves[r]
i1, i2 = l, r
```

투포인터 알고리즘 부분

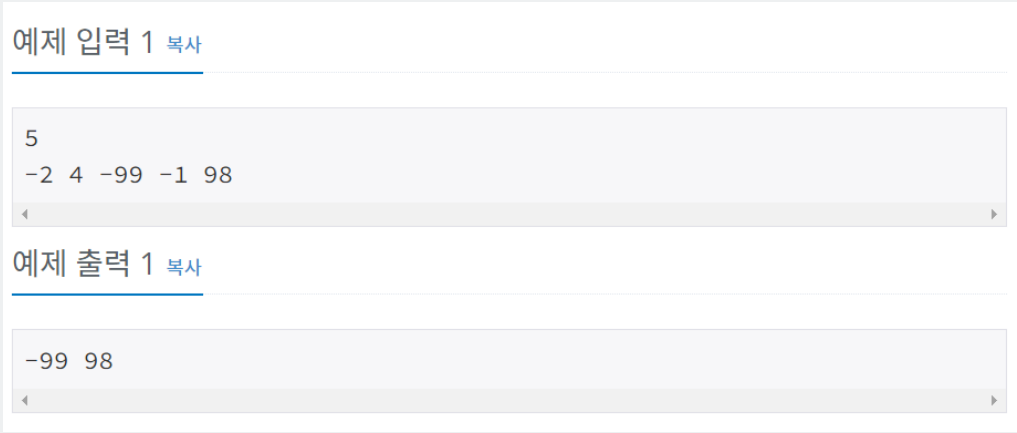
- (5) 왼쪽 포인터가 오른쪽 포인터보다 작은 동안, 매 루프 시작마다 새로운 혼합물 mix를 만들기
- (6) 0과 같거나 기존 혼합물보다 0과 가까운지(음수일 수도 있기 때문에 절댓값으로) 비교
- (7) 만약 새로 만든 혼합물이 0과 같거나 더 적다면, 기존 혼합물을 새로 만든 혼합물로 바꾸고 i1, i2에 l, r 인덱스 저장

```
while l < r:
    mix = resolves[l] + resolves[r]
    if mix == 0 or abs(mix) < abs(attr):
        attr = mix
        i1 = l
        i2 = r

    if mix == 0:
        break
    elif mix < 0:
        l += 1
    else:
        r -= 1

print(resolves[i1], resolves[i2])
```

-99	-2	-1	4	98

i1, i2 포인터는 왜 필요한가?

- l1, i2 포인터가 없으면 에러가 날 수 있음
- 예제의 리스트는 정렬했을 때 [-99, -2, -1, 4, 98]
- 그래서 attr(-1), l(0), r(4)
- Attr이 0보다 작으므로 왼쪽 포인터를 이동
- -2 + 98 -> 96은 절댓값 1보다 크므로 attr은 -1로 유지, l(1), r(4)
- Mix(96)이 0보다 크므로 오른쪽 포인터를 이동
- -2 + 4 -> 2은 절댓값 1보다 크므로 attr은 -1로 유지, l(1), r(3)



- Mix(2)이 0보다 크므로 오른쪽 포인터 이동
- -2 + -1 -> -3은 절댓값 1보다 크므로 attr은 -1로 유지, l(1), r(2)
- Mix(-3)이 0보다 작으므로 왼쪽 포인터 이동
- 이 때, l이 2가 되면서 r과 같아짐.
  - while 루프를 탈출하게 됨
- 이 l, r을 그대로 답으로 쓰면 l도 -1, r도 -1을 가리키기 때문에 오답임
- 따라서 정상범위의 l, r을 추적하는 i1, i2 변수가 필요한 것

전체 코드

```python
n = int(input())
resolves = list(map(int, input().split()))
resolves.sort()

l = 0
r = n-1
attr = resolves[l] + resolves[r]
i1, i2 = l, r

while l < r:
    mix = resolves[l] + resolves[r]
    if mix == 0 or abs(mix) < abs(attr):
        attr = mix
        i1 = l
        i2 = r

    if mix == 0:
        break
    elif mix < 0:
        l += 1
    else:
        r -= 1

print(resolves[i1], resolves[i2])
```

N번째 큰 수

- 백준 2075
- 메모리 제한이 아주 빡센 문제
  (시간/메모리 제한만 없다면 쉬울 문제)
- 최대 1500*1500 사이즈 입력이므로 한번에 리스트에 저장해서 풀 수 없음
- 한 줄 한 줄마다 N번째 큰 수를 구해야 함

N번째 큰 수 성공

☆

골드 V

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	12 MB (하단 참고)	12314	4964	3459	39.531%

문제

N×N의 표에 수 N^2개 채워져 있다. 채워진 수에는 한 가지 특징이 있는데, 모든 수는 자신의 한 칸 위에 있는 수보다 크다는 것이다. N=5일 때의 예를 보자.

12	7	9	15	5
13	8	11	19	6
21	10	26	31	16
48	14	28	35	25
52	20	32	41	49

이러한 표가 주어졌을 때, N번째 큰 수를 찾는 프로그램을 작성하시오. 표에 채워진 수는 모두 다르다.

일반 리스트 쓰기

일반 리스트를 사용해서 풀기

Nth에 항상 N개의 요소들만 있도록 하고, 새로운 값들과 비교해 추가하고 매번 정렬하는 방법

하지만 이 방법은 O(N^4) 시간복잡도

(?? N^3인가요 N^4인가요? ㅋㅋㅋ)

```python
N = int(input())
nth = []
for i in range(N):
    if i == 0:
        nth = list(map(int, input().split()))
    else:
        tmp = list(map(int, input().split()))
        for j in range(N):
            if tmp[j] >= min(nth):
                nth.sort()
                nth.pop(0)
                nth.append(tmp[j])
nth.sort()
print(nth[0])
```

힙 + 슬라이딩 윈도우로 구현하기

- 비슷한 로직이지만 힙으로 구현하기
- 이 때 포인트는 힙에 든 요소의 개수가 항상 N개로 고정되어 있다는 것 (즉, 슬라이딩 윈도우)
- 이 문제는 힙 문제로 구분될 수도 있지만 고정되어 있다는 점에서 슬라이딩 윈도우로 구분될 수도 있음
- 시간 복잡도는 O(N^2logN)

```python
import heapq

N = int(input())
heap = []
for _ in range(N):
    if not heap:
        heap = list(map(int, input().split()))
        heapq.heapify(heap)
        continue
    else:
        tmp = list(map(int, input().split()))
        for j in range(N):
            if tmp[j] >= heap[0]:
                heapq.heappop(heap)
                heapq.heappush(heap, tmp[j])
print(heap[0])
```

수업 문제

- 지금 풀어봅시다!
- 백준 2531 회전 초밥

공통 문제

- 백준 2230 수 고르기
- 백준 2096 내려가기 (슬라이딩 윈도우+DP)
- 백준 1806 부분합

- 그리디
- 완전탐색+구현 -> 수지님 (2/14)
- 백트랙킹 -> 수현님 (2/21)