5문제를 풀어보았다.
백준 1080번: 행렬
백준 1449번: 수리공 항승
백준 1541번: 잃어버린 괄호
백준 1931번: 회의실 배정
백준 20300번: 서강근육맨

백준 1080번: 행렬

행렬 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	13388	5293	4189	39.657%

문제

0과 1로만 이루어진 행렬 A와 행렬 B가 있다. 이때, 행렬 A를 행렬 B로 바꾸는데 필요한 연산의 횟수의 최솟값을 구하는 프로그램을 작성하시오.

행렬을 변환하는 연산은 어떤 3×3크기의 부분 행렬에 있는 모든 원소를 뒤집는 것이다. (0 → 1, 1 → 0)

입력

첫째 줄에 행렬의 크기 N M이 주어진다. N과 M은 50보다 작거나 같은 자연수이다. 둘째 줄부터 N개의 줄에는 행렬 A가 주어지고, 그 다음줄부터 N개의 줄에는 행렬 B가 주어진다.

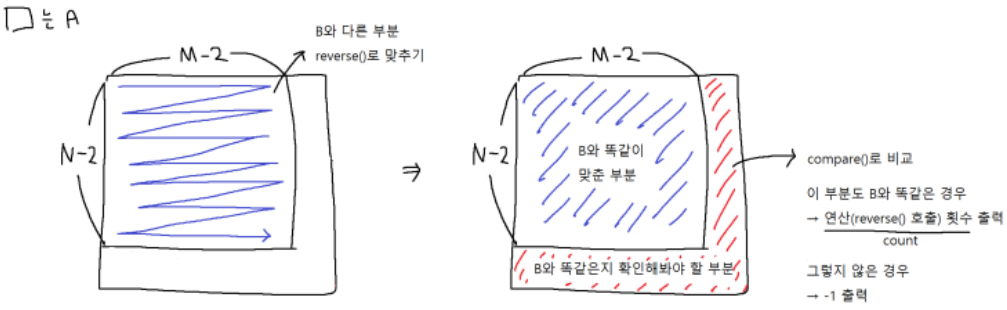
출력

첫째 줄에 문제의 정답을 출력한다. 만약 A를 B로 바꿀 수 없다면 -1을 출력한다.

https://www.acmicpc.net/problem/1080

```python
# 입력
N, M = map(int, input().split())
A = [list(map(int, list(input()))) for _ in range(N)]
B = [list(map(int, list(input()))) for _ in range(N)]

# 연산 횟수
count = 0

# 행렬을 변환하는 연산(원소 뒤집기)
def reverse(x, y):
    for i in range(x, x+3):
        for j in range(y, y+3):
            A[i][j] = 1 - A[i][j]

# A와 B가 같으면 True, 다르면 False 리턴
def compare():
    for i in range(N):
        for j in range(M):
            if N - 3 < i or M - 3 < j:
                if A[i][j] != B[i][j]:
                    return False
    return True

for i in range(N - 2):
    for j in range(M - 2):
        if A[i][j] != B[i][j]:
            reverse(i, j)
            count += 1

if compare():
    print(count)
else:
    print(-1)
```

입력 받는 방법을 떠올리지 못해서 어지러웠다.
문자열은 인덱싱이 안 되니까 숫자 리스트로 받아야하는데, 리스트 안에 리스트를 반복으로 넣는다는 걸 코드로 표현하는 게 잘 안 됐다.
N*M 행렬의 좌상단부터 총 (N-2)*(M-2)개의 인덱스만 정답과 비교해서 맞으면 그대로, 다르면 뒤집어준다.
이렇게하면 (N-2)*(M-2) 부분은 전부 정답과 똑같이 맞춰진다.
그렇기 때문에 compare()에서 A와 B를 비교할 때 (N-2)*(M-2) 이외의 부분만 비교한다.
이제 나머지 부분은 뒤집을 수 없다. 그걸 뒤집으면 정답으로 맞춰둔 부분이 뒤집혀버리기 때문이다.
오랜만에 그림판으로 소스 코드를 표현해보았다.

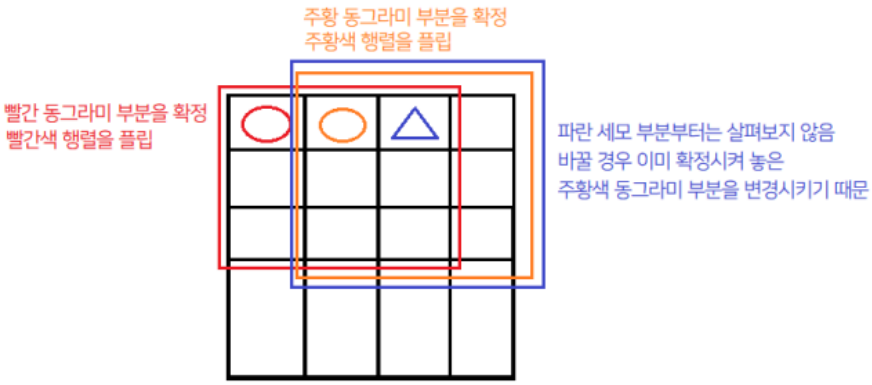


그림으로 표현한 소스 코드

정당성 분석이 참 어려웠는데 백준에 있는 답변이 이해하는 데 큰 도움이 되었다.

Q2. 이게 왜 최적의 해가 되는 건가요? 그냥 순서대로 뒤집어 보다가 단순히 맞추는 것이 아닌가요? 그것이 어떻게 최저횟수가 되는지 모르겠습니다.
→ 이 문제의 목적은 행렬 A를 행렬 B로 바꾸는 것입니다. 따라서 행렬 A에서 0인데 B에서 1인 칸이 있으면 반드시 뒤집어야 합니다. 문제는 그 연산을 몇 번 하느냐입니다. 뒤집는 연산의 특성상 2번 이상 뒤집는 것은 의미가 없습니다. 2번 뒤집으면 원래대로 돌아오고, 3번 뒤집는 것은 1번 뒤집는 것과 같습니다.
이때 최적의 선택은 "반드시 뒤집어야 하는 칸을 뒤집되, 뒤집는 횟수를 최소화하는 것" 입니다. 즉, 이미 뒤집은 칸을 2번이나 3번 뒤집지 않습니다.
이 전략은 다음과 같이 구현됩니다. 행렬을 순서대로 탐색하면서, "반드시 뒤집어야 하는 부분이 있으면" 바꿉니다. 뒤집을 필요가 없으면 확정하고 넘어갑니다.
그러다 어떤 칸을 만났는데 그 칸을 뒤집으면 이전에 확정시켜 놓은 칸이 다시 뒤집힐 경우, 살펴보지 않고 넘어갑니다.
이렇게 하면 5 × 5 행렬의 경우, 좌상단부터 3 × 3 부분에 있는 칸만 확정해도 이 행렬을 B로 변환 가능한지 아닌지를 알 수 있습니다.
이 알고리즘은 매 순간 '뒤집어야 하는 칸의 뒤집는 횟수를 최소화한다'는 점에서 그리디 알고리즘으로 분류된다고 생각합니다.

https://www.acmicpc.net/board/view/13509



https://www.acmicpc.net/board/view/13509

백준 1449번: 수리공 항승

수리공 항승 실금

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	11217	4616	3878	40.899%

문제

항승이는 품질이 심각하게 나쁜 수도 파이프 회사의 수리공이다. 항승이는 세준 지하철 공사에서 물이 샌다는 소식을 듣고 수리를 하러 갔다.

파이프에서 물이 새는 곳은 신기하게도 가장 왼쪽에서 정수만큼 떨어진 거리만 물이 샌다.

항승이는 길이가 L인 테이프를 무한개 가지고 있다.

항승이는 테이프를 이용해서 물을 막으려고 한다. 항승이는 항상 물을 막을 때, 적어도 그 위치의 좌우 0.5만큼 간격을 줘야 물이 다시는 안 샌다고 생각한다.

물이 새는 곳의 위치와, 항승이가 가지고 있는 테이프의 길이 L이 주어졌을 때, 항승이가 필요한 테이프의 최소 개수를 구하는 프로그램을 작성하시오. 테이프를 자를 수 없고, 테이프를 겹쳐서 붙이는 것도 가능하다.

입력

첫째 줄에 물이 새는 곳의 개수 N과 테이프의 길이 L이 주어진다. 둘째 줄에는 물이 새는 곳의 위치가 주어진다. N과 L은 1,000보다 작거나 같은 자연수이고, 물이 새는 곳의 위치는 1,000보다 작거나 같은 자연수이다.

출력

첫째 줄에 항승이가 필요한 테이프의 개수를 출력한다.

https://www.acmicpc.net/problem/1449

처음에 아래처럼 브루트 포스로 풀었다가, 탐욕법으로 풀지 않았다는 지적을 받고 수정했다.

```python
N, L = map(int, input().split())
location = list(map(int, input().split()))
tape = 0
count = 0

for i in range(1, 1001):
    if 0 < tape:
        tape -= 1
        continue
    if i in location:
        tape = L - 1
        count += 1

print(count)


'''
쉽게 생각하면 그냥 N을 L로 나눈 값을 올림하면 됨
그런데 테이프는 자를 수 없으므로 이걸 처리하는 게 관건
어차피 테이프는 물이 새는 곳이 보이면 바로 붙여야 됨
그 때 1 카운트하고
그 뒤로는 테이프 길이만큼 테이프가 붙어 있을 테니까 체크하지 말고 테이프 길이만큼 패스하자
카운트한 만큼 출력하면 될 듯
'''
```

아래가 탐욕법으로 푼 답안이다.
왼쪽(0)에 가까운 물이 새는 곳부터 테이프로 막고 시작한다고 보면 된다. 그래서 count를 1로 초기화하고 시작한다.
모든 물이 새는 곳을 반복문으로 돌면서 이전 위치에 붙인 테이프가 닿지 않으면 테이프를 붙이고
현재 위치로 before(가장 최근에 테이프를 붙인 위치)를 바꾼다. 테이프가 닿으면 추가로 더 붙일 필요가 없으니 넘어간다.
테이프는 항상 물이 새는 곳 왼쪽 0.5부터 시작하도록 붙인다는 걸 전제하고 있다.

```python
# 물이 새는 곳의 개수, 테이프의 길이 입력
N, L = map(int, input().split())

# 물이 새는 곳의 위치 입력
leak = list(map(int, input().split()))

# 오름차순 정렬
leak.sort()

# 사용 테이프 개수
count = 1

# 초기화시 이전 위치
before = leak[0]

# 물이 새는 곳에서
for i in leak:
    # 이전 위치의 테이프가 닿지 않으면 테이프 추가
    if i >= before + L:
        count += 1
        before = i
    # 닿으면 추가 안 함

print(count)
```

백준 1541번: 잃어버린 괄호

잃어버린 괄호 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	38272	18440	14865	48.104%

문제

세준이는 양수와 +, -, 그리고 괄호를 가지고 식을 만들었다. 그리고 나서 세준이는 괄호를 모두 지웠다.

그리고 나서 세준이는 괄호를 적절히 쳐서 이 식의 값을 최소로 만들려고 한다.

괄호를 적절히 쳐서 이 식의 값을 최소로 만드는 프로그램을 작성하시오.

입력

첫째 줄에 식이 주어진다. 식은 '0'~'9', '+', 그리고 '-'만으로 이루어져 있고, 가장 처음과 마지막 문자는 숫자이다. 그리고 연속해서 두 개 이상의 연산자가 나타나지 않고, 5자리보다 많이 연속되는 숫자는 없다. 수는 0으로 시작할 수 있다. 입력으로 주어지는 식의 길이는 50보다 작거나 같다.

출력

첫째 줄에 정답을 출력한다.

https://www.acmicpc.net/problem/1541

1541 콜렉트콜... 폰 없던 시절 도서관 다닐 때 전화하려고 많이 썼던 기억이 난다. 가끔 내가 이걸로 전화 받기도 하고 그랬는데 아무든

```
plus = 1
number = 0
result = 0
data = input()

def cal():
    global number, result
    if plus:
        result += number
    else:
        result -= number
    number = 0

for i in range(len(data)):
    if data[i] == '+' or data[i] == '-':
        cal()
        if data[i] == '+' and plus == 1:
            plus = 1
        else:
            plus = 0
    else:
        number *= 10
        number += int(data[i])

cal()
print(result)
```

무난무난하게 풀었다.
문제 해결 아이디어는 식을 최소로 만들기 위해서 플러스 다음 마이너스는 무조건 괄호로 묶어야 한다는 것이다.
그러니까 사실상 마이너스 다음 플러스는 마이너스로 처리하면 된다.
아직 Python에 익숙하지 않아서 그런지 코드가 뭔가 C 언어스럽다.

백준 1931번: 회의실 배정

회의실 배정 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	101572	30747	22028	29.157%

문제

한 개의 회의실이 있는데 이를 사용하고자 하는 N개의 회의에 대하여 회의실 사용표를 만들려고 한다. 각 회의 I에 대해 시작시간과 끝나는 시간이 주어져 있고, 각 회의가 겹치지 않게 하면서 회의실을 사용할 수 있는 회의의 최대 개수를 찾아보자. 단, 회의는 한번 시작하면 중간에 중단될 수 없으며 한 회의가 끝나는 것과 동시에 다음 회의가 시작될 수 있다. 회의의 시작시간과 끝나는 시간이 같을 수도 있다. 이 경우에는 시작하자마자 끝나는 것으로 생각하면 된다.

입력

첫째 줄에 회의의 수 N(1 ≤ N ≤ 100,000)이 주어진다. 둘째 줄부터 N+1 줄까지 각 회의의 정보가 주어지는데 이것은 공백을 사이에 두고 회의의 시작시간과 끝나는 시간이 주어진다. 시작 시간과 끝나는 시간은 2^{31}-1보다 작거나 같은 자연수 또는 0이다.

출력

첫째 줄에 최대 사용할 수 있는 회의의 최대 개수를 출력한다.

https://www.acmicpc.net/problem/1931

처음에는 회의 시간의 합, 그 다음에는 회의 종료 시간 기준으로 정렬해서 탐욕법 적용했는데 틀렸다.
sort() 함수를 잘 모르고 썼던 것이 화근이었다.
sort()로 정렬하면 같은 값일 때 해당 리스트 다든 원소를 가지고 비교하고, 그마저 같은 쪽이면 먼저 입력받은 순서대로 정렬하는 건 줄 알았다.
그런데 써보니 같은 값일 때 다든 원소는 고려하지 않고 그냥 입력받은 순서대로 정렬이 되는 것으로 보인다.

합 정렬의 반례는 아래와 같다. [2, 2], [2, 5]로 답이 2가 나와야 하나 [0, 3]이 합이 3으로 제일 작다보니 1이 나온다.

```
3
0 3
2 2
2 5
```

종료 시간 정렬은 같은 예제를 입력하더라도 입력이 들어온 순서에 따라 정답이 다르게 나올 수 있었다. 반례는 아래와 같다.
답은 [2, 4], [4, 4]로 2인데 종료 시간 기준으로 정렬해보면 둘 다 4로 똑같고, [4, 4]가 먼저 입력돼서 [4, 4]가 먼저 처리되어 1이 나온다.

```
2
4 4
2 4
```

이 때문에 시작 시간 기준으로 오름차순 정렬을 해둔 다음 종료 시간 기준으로 오름차순 정렬을 하는 것이 정답.
참고로 시작 시간 정렬은 합 정렬처럼 그냥 틀렸다. 반례는 아래와 같다.
종료 시간 반례와 비슷하다. 답은 2인데 [2, 4]가 먼저 처리되어 1이 나온다.

```
2
2 4
2 2
```

그럼 종료 시간 기준으로 오름차순 정렬해놓고 시작 시간 기준으로 오름차순 정렬하면 될까?
그럼 위 반례는 사라지지만 새로운 반례가 생긴다. 새로운 반례는 아래와 같다.
답은 [3, 4], [4, 5]로 2지만 시작 시간 기준이다보니 [2, 99]가 먼저 처리되어 1이 나온다. 아예 큰 수로 못박아버리니까 확 와닿는다.

```
3
2 99
3 4
4 5
```

정당성 분석을 해보자. 왜 종료 시간 기준일 때 탐욕법 적용이 가능할까?
일단 시작 시간, 종료 시간이 빠를수록 좋다는 건 자명하다.
그런데 반례에서 본 것처럼 시작 시간 기준이면 시작 시간이 빠르더라도 종료 시간이 무지막지하게 큰 경우 답이 없다.
하지만 종료 시간은 빠르면 시작 시간이 자연히 빨라지고(시작 시간 <= 종료 시간)
빨리 끝나야 이후 다든 회의 배정할 시간이 늘어나니까 당연한 거 같긴 한데 뭔가 논리적으로, 공식으로 설명해보려니까 잘 안 된다.
나름 설명도 충분히 했고 내가 얼추 이해한 것 같으니까 일단 여기서 마무리.

```
# 입력
N = int(input())
times = [list(map(int, input().split())) for _ in range(N)]

# 회의 시작 시간 기준 오름차순으로 정렬
times.sort(key=lambda x: x[0])
# 회의 종료 시간 기준 오름차순으로 정렬
times.sort(key=lambda x: x[1])

# 회의의 개수
count = 0

# 마지막 회의가 끝나는 시간
last_end_time = 0

for start_time, end_time in times:
    if last_end_time <= start_time:
        count += 1
        last_end_time = end_time

print(count)
```

백준 20300번: 서강근육맨

서강근육맨 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초 (추가 시간 없음)	1024 MB	1232	476	399	38.109%

문제

로니 콜먼 동영상을 보고 보디빌더가 되기로 결심한 항빈이는 PT 상담을 받으러 서강헬스클럽에 갔다. 항빈이가 서강헬스클럽을 선택한 이유는 PT를 받을 때 사용하는 운동기구를 회원이 선택할 수 있다는 점 때문이다. 하지만, 서강헬스클럽은 항상 사람이 많아서 PT를 한 번 받을 때 운동기구를 최대 두 개까지만 선택할 수 있다.

헬스장에 있는 N개의 운동기구를 한 번씩 사용해보고 싶은 항빈이는 PT를 받을 때마다 이전에 사용하지 않았던 운동기구를 선택하기로 계획을 세웠다. 그리고 비용을 절약하기 위해 PT를 받을 때 운동기구를 되도록이면 두 개를 사용하기로 했다. 예를 들어, 헬스장에 총 10개의 운동기구가 있을 경우 PT를 5번 받으면 모든 기구를 다 사용할 수 있다. 9개의 운동기구가 있는 경우에도 PT를 5번 받지만, 마지막 PT를 받을 때는 운동기구를 하나만 사용한다.

하지만 항빈이는 운동기구를 선택하다가 큰 고민에 빠졌다. 왜냐하면 운동기구마다 근손실이 일어나는 정도가 다르기 때문이다. 어떤 운동기구는 자극이 잘 안 와서 근손실이 적게 일어나는데, 어떤 운동기구는 자극이 잘 와서 근손실이 많이 일어난다. 근손실이 죽음보다 무서운 항빈이는 PT를 한 번 받을 때의 근손실 정도가 M을 넘지 않도록 하고 싶다. 이때, M의 최솟값을 구해보자. 참고로, 운동기구를 두 개 사용해서 PT를 받을 때의 근손실 정도는 두 운동기구의 근손실 정도의 합이다.

입력

첫째 줄에 서강헬스클럽에 비치된 운동기구의 개수를 나타내는 정수 N이 주어진다. ($1 \le N \le 10\,000$)

둘째 줄에는 각 운동기구의 근손실 정도를 나타내는 정수 $t_1, t_2, \cdots, t_N$가 주어진다. ($0 \le t_i \le 10^{18}$)

출력

M의 최솟값을 출력한다.

https://www.acmicpc.net/problem/20300

M의 최솟값은 각 운동기구의 근손실 정도를 나타내는 정수 리스트를 오름차순으로 정렬했을 때
N이 짝수일 때는 앞에서 i번째 원소와 뒤에서 i번째 원소를 더한 값 중 가장 작은 값이다.
N이 홀수일 때는 가장 큰 값은 더한 값으로 치고 빼므로 앞에서 i번째 원소와 뒤에서 i - 1번째 원소를 더한 값 중 가장 작은 값이다.
홀수의 운동기구가 있는 경우에는 운동기구를 하나만 쓰는 날이 생기기 때문이다.
이것도 정당성 분석이 어렵다. 큰 값들을 작게 만들어야 해서 그런 것 같은데... 이것도 얼추 이해한 것 같아 넘어간다.
제대로 이해하고 설명하기 위해서는 수학을 좀 알아야겠다 싶었다.

```python
N = int(input())
data = list(map(int, input().split()))
data.sort()
M = 0
even = 1

if len(data) % 2 == 1:
    even = 2
    M = data[len(data) - 1]

for i in range(len(data) // 2):
    result = data[i] + data[len(data) - i - even]
    if M < result:
        M = data[i] + data[len(data) - i - even]

print(M)
```

총평

파이썬 문법과 정당성 분석에 매우 약하다는 걸 느꼈다. 꾸준히, 꼼꼼히 공부해야 해결할 수 있는 문제라고 생각한다. 밑빠진 독에 물 붓고 싶지는 않다. 같은 실수를 반복하지 않도록, 이전 삽질 정리한 내용 선에서 해결되도록 기록하면서 꼼꼼히 공부해야 한다.
미래의 나든 다른 사람이든 같은 문제가 생겼을 때 똑같은 구글링 반복하지 않고 내 블로그 보는 것만으로 문제를 이해하고 해결할 수 있도록 정리하는 것이 목표다.