9월 30일 (목)

1. 정보처리기사 실기 공부 - 3장 데이터 입출력

2. 백준 알고리즘 풀이 - 브루트 포스 및 엘리스 SW 코딩테스트 문제풀이

사탕 게임 출처 다국어

☆ 한국어

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	128 MB	20306	6427	4504	31.077%

문제

상근이는 어렸을 적에 "봄보니 (Bomboni)" 게임을 즐겨했다.

가장 처음에 N×N크기에 사탕을 채워 놓는다. 사탕의 색은 모두 같지 않을 수도 있다. 상근이는 사탕의 색이 다른 인접한 두 칸을 고른다. 그 다음 고른 칸에 들어있는 사탕을 서로 교환한다. 이제, 모두 같은 색으로 이루어져 있는 가장 긴 연속 부분(행 또는 열)을 고른 다음 그 사탕을 모두 먹는다.

사탕이 채워진 상태가 주어졌을 때, 상근이가 먹을 수 있는 사탕의 최대 개수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에 보드의 크기 N이 주어진다. (3 ≤ N ≤ 50)

다음 N개 줄에는 보드에 채워져 있는 사탕의 색상이 주어진다. 빨간색은 C, 파란색은 P, 초록색은 Z, 노란색은 Y로 주어진다.

사탕의 색이 다른 인접한 두 칸이 존재하는 입력만 주어진다.

출력

첫째 줄에 상근이가 먹을 수 있는 사탕의 최대 개수를 출력한다.

예제 입력 1 복사

```
3
CCP
CCP
PPC
```

예제 출력 1 복사

```
3
```

예제 입력 2 복사

```
4
PPPP
CYZY
CCPY
PPCC
```

예제 출력 2 복사

```
4
```

예제 입력 3 복사

```
5
YCPZY
CYZZP
CCPPP
YCYZC
CPPZZ
```

예제 출력 3 복사

```
4
```

힌트

예제 3의 경우 4번 행의 Y와 C를 바꾸면 사탕 네 개를 먹을 수 있다.

- 골드바흐의 추측 (엘리스SW 엔지니어 트랙 코딩테스트 준비)

골드바흐의 추측

골드바흐의 추측이란 2보다 큰 짝수는 두 소수의 합으로 나타낼 수 있다는 추측입니다.

예를 들어 4, 6, 8은 다음과 같이 나타낼 수 있습니다.

4 = 2 + 2

6 = 3 + 3

8 = 3 + 5

현재 이 추측은 참인지 거짓인지 아직 증명되지 않았으나 충분히 크지 않은 수에 대해서는 컴퓨터를 통해 참으로 결론을 내린 상태입니다.

여러분도 한번 짝수를 두 소수의 합으로 나타내어 봅시다!

입력

N개의 짝수가 공백으로 구분되어 입력됩니다.

각 짝수는 모두 최대 10,000을 넘지 않습니다.

출력

줄마다 합계가 짝수가 되는 두 소수를 작은 소수부터 차례대로 출력합니다.

만약 해당 짝수를 만드는 소수의 쌍이 여러 개라면 가장 두 소수의 차이가 적은 것을 출력합니다.

시간제한

각 실행 케이스마다 1.5초의 시간이 주어집니다.

입력 예시

```
4 6 8
```

Copy

입력 예시

```
4 6 8
```

출력 예시

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

부분점수

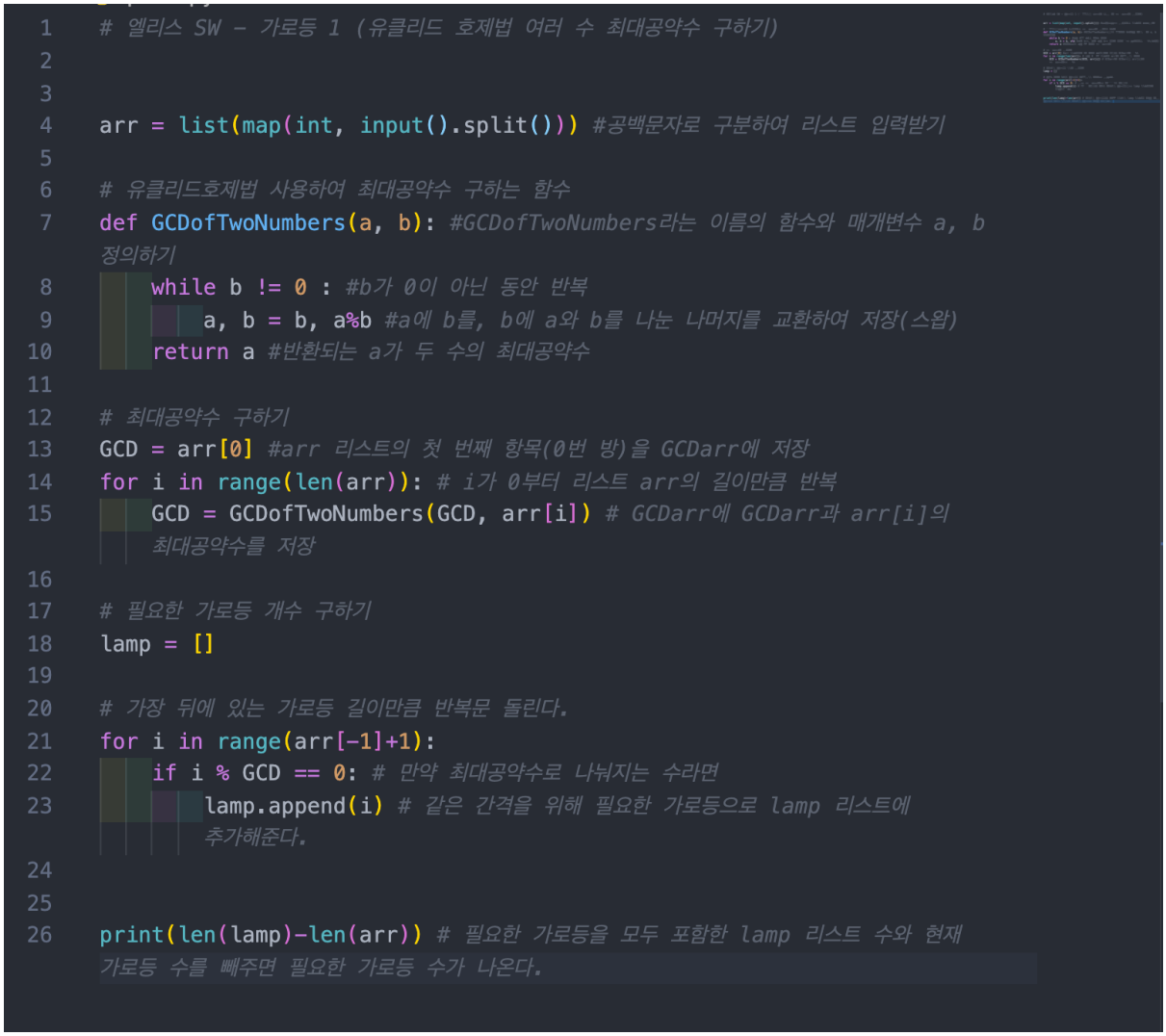
- $1 \leq N \leq 5$: 30점
- 5 < N \leq 500 : 70점

```
from prime import prime_list
# prime_list는 1부터 10000사이의 소수가 오름차순으로 저장된 리스트 입니다.

# 소수리스트 얻는 함수
def getPrimes(n):
    n = int(n)
    primes = []
    for i in prime_list:
        if i <= n:
            primes.append(i)
        else:
            break
    return primes


# 두 소수의 합 구하는 함수
def getAddedPrimes(n):
    li = getPrimes(n)  # 소수리스트 얻는 함수 이용
    i = 0 # 리스트 첫번째값 포인트
    j = -1 # 리스트 마지막값 포인트
    arr = [] # 정답 리스트

    # 탐색시작
    while li[j] >= li[i]:
        sum_value = li[j] + li[i]
        if n > sum_value: # 더한 값이 입력값보다 크기가 작으면 앞 포인터 오른쪽으로 이동
            i += 1
```

```
        elif n < sum_value: # 더한 값이 입력값보다 크기가 크면 뒤 포인터 왼쪽으로 이동
            j += -1

        elif n == sum_value: # 더한 값이 입력값과 같고
            if li[i] != li[j]: # 두 값이 같지않다면
                arr.append([li[i], li[j]])
                i += 1  # 차이가 더 작은 수를 찾기 위해 계속 탐색
                j -= 1  # 두개 모두 이동
            else:
                arr.append([li[i], li[j]]) # 두 값이 같으면 정답 리스트에 추가 후 반복중지
                break

    # 리스트 제거후 출력
    return ' '.join(map(str, arr[-1]))



# 정답 출력

evens = input().split() # 입력받기 TIL: .split() 는 리스트를 반환해낸다.

for even in evens:
  even = int(even)
  print(getAddedPrimes(even))
```

- (엘리스SW) 가로등 1 (최대공약수 구하기) - 첫 번째 문제들

기존 가로등에서 같은 간격으로 맞추려고 할 때 필요한 가로등 수 구하기



```python
# 엘리스 SW - 가로등 1 (유클리드 호제법 여러 수 최대공약수 구하기)


arr = list(map(int, input().split())) #공백문자로 구분하여 리스트 입력받기

# 유클리드호제법 사용하여 최대공약수 구하는 함수
def GCDofTwoNumbers(a, b): #GCDofTwoNumbers라는 이름의 함수와 매개변수 a, b 정의하기
    while b != 0 : #b가 0이 아닌 동안 반복
        a, b = b, a%b #a에 b를, b에 a와 b를 나눈 나머지를 교환하여 저장(스왑)
    return a #반환되는 a가 두 수의 최대공약수

# 최대공약수 구하기
GCD = arr[0] #arr 리스트의 첫 번째 항목(0번 방)을 GCDarr에 저장
for i in range(len(arr)): # i가 0부터 리스트 arr의 길이만큼 반복
    GCD = GCDofTwoNumbers(GCD, arr[i]) # GCDarr에 GCDarr과 arr[i]의 최대공약수를 저장

# 필요한 가로등 개수 구하기
lamp = []

# 가장 뒤에 있는 가로등 길이만큼 반복문 돌린다.
for i in range(arr[-1]+1):
    if i % GCD == 0: # 만약 최대공약수로 나눠지는 수라면
        lamp.append(i) # 같은 간격을 위해 필요한 가로등으로 lamp 리스트에 추가해준다.


print(len(lamp)-len(arr)) # 필요한 가로등을 모두 포함한 lamp 리스트 수와 현재 가로등 수를 빼주면 필요한 가로등 수가 나온다.
```

최대공약수 구하는 유클리드 호제법 식

```
# 유클리드호제법 사용하여 최대공약수 구하는 함수
def GCDofTwoNumbers(a, b): #GCDofTwoNumbers라는 이름의 함수와 매개변수 a, b 정의하기
    while b != 0 : #b가 0이 아닌 동안 반복
        a, b = b, a%b #a에 b를, b에 a와 b를 나눈 나머지를 교환하여 저장(스왑)
    return a #반환되는 a가 두 수의 최대공약수
```

```
print(2%8)
# print : 2
```

- (엘리스SW) 여기서부터 여기까지 - 첫번째 문제

```
# 여기서부터 여기까지 문제


# 현재 가진 돈 입력
money = int(input())

# 상점의 물품 가격리스트 입력
items = list(map(int, input().split()))

# TIL: 이중 반복문 중단스위치
isbreak = False
# 더하는 값
total = 0

# 리스트 내 원소 하나하나 더한다.
for i in range(len(items)):
    if items[i] != items[-1]: # 현재 원소가 마지막 원소가 아니라면
        total += items[i] # 더해나간다.
        for j in range(i+1, len(items)): # 실수한 부분: i+1 이 아니라 1로 썼다.
            total += items[j]
            if total > money:
                total = 0
                break
            elif total == money:
                start_idx, end_idx = i, j
                isbreak = True # 실수한 부분: isbreak == True 로 적었다.
                break
            else:
                continue
        if isbreak == True: # 이중 for문 break 스위치
            print('와 '.join(map(str, items[start_idx : end_idx+1])) + '을 가르키며...')
            print('여기서부터 여기까지 다 주세요!')
            break

    # 현재원소가 마지막 원소이고 마지막 원소가 내 예산과 같다면
    elif (items[i] == items[-1]) and (items[i] == money):
        print(str(items[i]) + '을 가르키며...')
        print('이거 주세요!')
        break

    # 현재 원소가 마지막 원소인데 마지막 원소까지 내 예산과 같지 않다면
    else:
        print('다음에 올게요...')
        break
```