스터디(Study)

[문제] 학과 친구들과 요즘 생성형 AI에 관한 유명한 책을 골라 이 책으로 그룹 스터디를 한다. 팀원은 모두 k명이며 각각 맡은 Chapter를 설명하고 해당 데모 코드까지 시연을 보이기로 했다. 그런데 각 Chapter마다 페이지수 와 데모할 code의 수가 달라서 그냥 Chapter 단위로 배분하면 챕터별 분량이 달라 분량의 공평성에 문제가 생긴다. 예를 5개의 Chapter로 구성된 책의 각 챕터 별 페이지가 다음과 같다고 하자.

Book = [35, 27, 67, 13, 45]

단 세미나의 원활한 진행을 위하여 각 사람에게는 연속된 장을 배정한다. 예를 들어 반드시 이어지는 연속된 장, 예를 들어 {C3, C4, C5}장을 한 사람에게 지정한다. 특정인에게 서로 연속되지 않은 장, 예를 들어 서로 떨어진 {C2, C7, C9} 장이나 {C1, C3}과 같은 식으로는 배정하지는 않는다.[1] 아래는 위 Book책의 각 챕터를 3 사람에게 배정하는 5가지 경우를 보여주고 있다. 물론 더 많은 방법도 가능하지만 예로 5개만 제시한다. Ci는 i−번째 장(chapter)를 의미한다.

만일 M4 배분과 같이 P2 에게 {C2, C3, C4} 장을 몰아주면 불공평하다고 난리가 날 것이다. 따라서 아래 5가지 배분 방법 중에서는 한 사람에게 배분된 최대 분량이 67로 가장 적은 M1을 선택해야 한다. 즉 각 3 사람에게 차례대로 {C1, C2 }장, {C3}장 {C4, C5}장을 배분한다.

표−1. 5장으로 구성된 책 [35, 27, 67, 13, 45]을 3사람에게 배정하는 몇 가지 방법

배분 방법	P1	P2	P3	최대 배정 분량
M1	35, 27	**67**	13, 15	67
M2	35,	**27, 67**	13, 15	27+67 = 94
M3	35,	27	**67,13,15**	67+13+15 = 95
M4	35,	**27,67,13**	15	27+67+13= 109
M5	**35,27,67**	13	15	35+27+67= 129

이 문제를 형식적으로 정의하면 다음과 같다. N개의 장으로 구성된 책을 k 명의 스터디 회원

1) 서로 떨어진 장을 특정인에게 배정하는 배정받은 장들의 내용상 영속성이 떨어져 준비하는 사람이 힘들게 된다.

들에게 배정을 하려고 한다. ($k < N$). 배분에는 한가지 제한 조건이 있는데 특정인에게는 반드시 연속된 장으로, 즉 즉 $[i,\, i+1,\, i+2,\, \ldots j-1,\, j]$장을 모두 배정해야 한다. 배분의 최종 목표는 가장 많은 페이지를 배당 받은 사람의 양이 최소가 되도록, 즉 최대 배분 페이지의 양을 최소화 불만을 최소화시켜 공평을 유지하는 것이다. 즉 이러한 문제는 min max 문제라고 부른다. 어떤 구성에서 나타나는 max 값을 minimize하는 것으로 최적화 문제의 대표적인 형식이다. [2)]

[입출력] 입력과 출력은 모든 표준 입출력을 사용한다. 입력파일 **stdin**의 첫 줄에는 챕터의 수를 나타내는 N과 스터디 그룹의 회원수 k가 공백을 두고 주어진다. 단 $k < N < 100$ 이다. 그 다음 두 번째 줄에는 각 챕터의 페이지 p_i가 정수로 한 줄에 모두 주어진다. 단 $1 \le p_i \le 200$. 여러분은 앞에서 설명한 방법으로 최대한 "공평하게" 배분했을 때 한 사람에게 배정된 최대분량의 페이지 수를 정수로 **stdout**에 출력해야 한다.

[예제]

stdin	**stdout**
`11 5` `45 10 13 65 12 34 88 45 13 67 90`	`125`
`9 4               //N=9, k=4` `11 12 13 21 22 23 31 32 33`	`63`
`8 4               //N=8, k=4` `11 12 13 21 22 23 31 32`	`54`

[제한조건] 제출 프로그램은 **study.{c, cpp, py}**이다. 제출 횟수는 최대 15회, 각 데이터 당 수행 제한 시간은 1초이다. 여러분의 코드에서 사용할 수 있는 token의 최대 갯수는 600이다.

2) 보통 min max optimization이라고 부른다. 예를 들어 100개의 점을 연결하는 도로망을 구성한다고 할 때 도로망 구성 이후에 그 도로상으로 볼 때 가장 먼 두 점이 존재한다. 이 점을 farthest point pair라고 한다. 즉 이 점은 max { dist(x,y), for x,y V}이다. 이렇게 했을 때 가장 좋은 도로망은 바로 이 가장 먼 점의 거리가 최소가 되는 도로망이라고 할 수 있다. 이것을 찾는 문제가 바로 min max optimization 문제이다.