[Algorithm] 13 강 : 그리디 유형 문제풀이 + 백준 알고리즘 11399번 ATM문제 — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북
만든 날짜: 2020-10-26 오후 5:07
URL: https://continuous-development.tistory.com/158

Algorithm

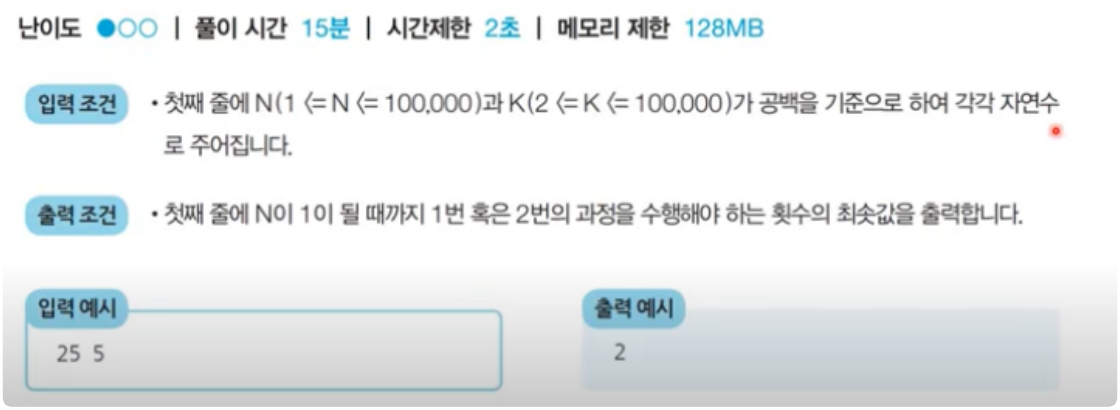
[Algorithm] 13 강 : 그리디 유형 문제풀이 + 백준 알고리즘 11399번 ATM문제

2020. 10. 26. 08:19 수정 삭제 공개

문제 1. 1이 될 때까지

〈문제〉 1이 될 때까지: 문제 설명

- 어떠한 수 **N이 1이 될 때까지** 다음의 두 과정 중 하나를 반복적으로 선택하여 수행하려고 합니다. 단, 두 번째 연산은 N이 K로 나누어 떨어질 때만 선택할 수 있습니다.
  1. N에서 1을 뺍니다.
  2. N을 K로 나눕니다.
- 예를 들어 N이 17, K가 4라고 가정합시다. 이때 1번의 과정을 한 번 수행하면 N은 16이 됩니다. 이후에 2번의 과정을 두 번 수행하면 N은 1이 됩니다. 결과적으로 이 경우 전체 과정을 실행한 횟수는 3이 됩니다. 이는 N을 1로 만드는 최소 횟수입니다.
- N과 K가 주어질 때 N이 1이 될 때까지 1번 혹은 2번의 **과정을 수행해야 하는 최소 횟수**를 구하는 프로그램을 작성하세요.



해당 문제에서 N이 25이고 K가 3일 때 문제를 가정해보자.

1단계 - 첫 번째로 N이 25 일 때는 k로 나눠지지 않는다. 이때는 1로 빼게 된다. | 25 - 1 =>24
2단계 - 24는 k로 나눠진다. 이때는 k로 나누게 된다. | 24 / 3 =>8
3단계 - 8은 k로 나눠지지 않는다. | 8-1 => 7
4단계 - 7은 k로 나눠 지지 않는다. | 7-1 => 6
5단계 - 6은 3으로 나눠진다 | 6/2 => 2
6단계 - 2는 3으로 나눠지지 않는다 | 2-1 =1
종료

이런 방식으로 진행 되게 된다.

이 방식에 대한 정당성 분석을 해봤을 때 가능한 최대한 많이 나누는 작업이 최적의 해를 항상 보장할 수 있는지 봐야 한다.

N이 아무리 큰 수여도, k로 나눈다면 -1을 하는 것보다는 더욱더 빠르게 줄일 수 있다.

그래서 가능한 최대한 많이 나누는 작업이 최적의 해를 보장한다. 이것을 코드로 구현하면 아래와 같다.

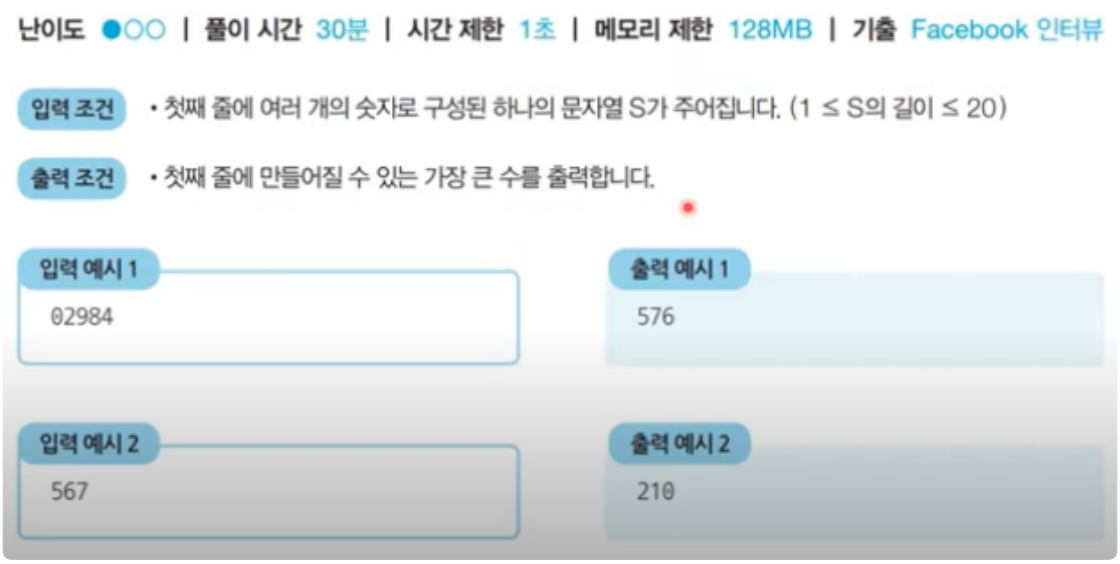
```
# N,K 공백을 기준으로 구분하여 입력 받기
n,k = map(int,input().split())

result = 0

while True:          # N이 K로 나누어 떨어지는 수가 될 때 까지 빼기
```

```
    target = (n//k)*k    # 이렇게 공식을 구성함으로 나눠지는 값을 구한다.
    result += (n-target)    # 나눠지지 않는 값을 result로 뺀다. 이건 1 만큼 단계가 증가됨으로 result에
    n = target            # result로 1을 빼는 작업을 하였으니 이제 target의 값을 n으로 변경해주고
                    # N이 k보다 작을 때 (더이상 나눌수 없을때) 반복분을 탈출한다.
    if n < k:
        break
    result += 1        # 나눴을때의 단계가 추가되는 것을 구현했다.
    n // = k           # 여기서 n을 나눈다.

result += (n - 1)        # 1이 됐을때 체크하는 부분을 위해 -1 을 해준다.
print(result)
```

문제 2, 곱하기 혹은 더하기

〈문제〉 곱하기 혹은 더하기: 문제 설명

- 각 자리가 숫자(0부터 9)로만 이루어진 문자열 S가 주어졌을 때, 왼쪽부터 오른쪽으로 하나씩 모든 숫자를 확인하며 숫자 사이에 '×' 혹은 '+' 연산자를 넣어 결과적으로 **만들어질 수 있는 가장 큰 수를 구하는 프로그램**을 작성하세요. 단, +보다 ×를 먼저 계산하는 일반적인 방식과는 달리, **모든 연산은 왼쪽에서부터 순서대로** 이루어진다고 가정합니다.
- 예를 들어 02984라는 문자열로 만들 수 있는 가장 큰 수는 ((((0 + 2) × 9) × 8) × 4) = 576입니다. 또한 만들어질 수 있는 가장 큰 수는 항상 20억 이하의 정수가 되도록 입력이 주어집니다.



해당 문제에 대해서 아이디어를 내면 대부분의 경우 + 보다는 * 가 값을 크게 만든다.
다만 예외의 경우가 있다. 0 또는 1일 때는 +이 값을 더 크게 만든다.
따라서 두 수에 대하여 연산을 할 때 두 값이 1 이하인 경우에는 + 로 1 이상인 경우에는 * 로하면
최적의 해를 구할 수 있다.

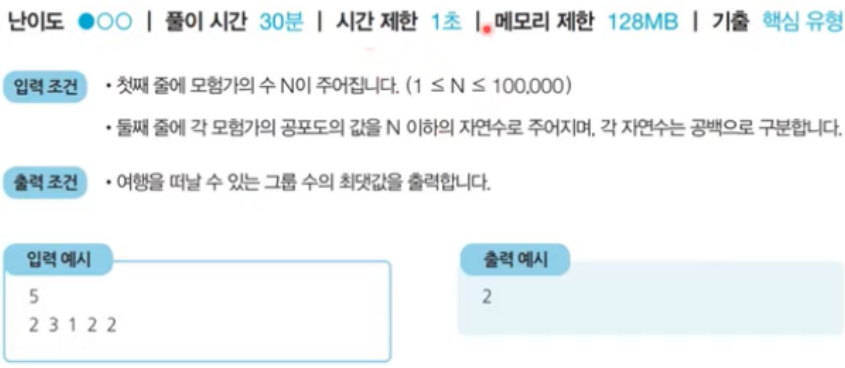
```
data = input()

# 첫번째 문자를 숫자로 변경하여 대입

result = int(data[0])

for i in range(1,len(data)):
    # 두 수 중에서 하나라도 '0' 혹은 '1'인 경우, 곱하기보다는 더하기 수행
    num = int(data[i])
    if num <= 1 or result <= 1:
        result += num
    else:
        result += num

print(result)
```

문제 3. 모험가 길드

〈문제〉 모험가 길드: 문제 설명

- 한 마을에 모험가가 N명 있습니다. 모험가 길드에서는 N명의 모험가를 대상으로 '공포도'를 측정했는데, '공포도'가 높은 모험가는 쉽게 공포를 느껴 위험 상황에서 제대로 대처할 능력이 떨어집니다.
- 모험가 길드장인 동빈이는 모험가 그룹을 안전하게 구성하고자 공포도가 X인 모험가는 반드시 X명 이상으로 구성한 모험가 그룹에 참여해야 여행을 떠날 수 있도록 규정했습니다.
- 동빈이는 최대 몇 개의 모험가 그룹을 만들 수 있는지 궁금합니다. N명의 모험가에 대한 정보가 주어졌을 때, **여행을 떠날 수 있는 그룹 수의 최댓값**을 구하는 프로그램을 작성하세요.

〈문제〉 모험가 길드: 문제 조건



위와 같이 입력 예시가 있을 때 고려해야 될 것은 공포도 이다.

공포도가 1인 모험가는 혼자서 모험이 가능하지만 공포도가 2인 경우에는 2명으로 구성되어야 한다.

이걸 고려해서 문제를 풀어야 한다.

해당 문제를 풀기 위해서

현재 그룹에 포함된 모험가의 수 >= 현재 확인하고 있는 공포도

라는 공식이 성립된다. 해당 문제에서는 공포도가 더 커서는 안되기 때문이다. 이것을 코드로 구현하면

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

result = 0 # 총 그룹의 수
count = 0 # 현재 그룹에 포함된 모험가의 수

for i in data: # 공포도를 낮은 것 부터 하나씩 확인하며
    count += 1 # 현재 그룹에 해당 모험가를 포함시키디
    if count >= i: # 현재 그룹에 포함된 모험가의 수가 현재의 공포도보다 크거나 같다면 그룹 결성
        result += 1 # 총 그룹의 수를 증가시키고
        count = 0 #모험가의 수를 초기화 한다.

print(result) # 총 그룹의 수 출력
```

11399번 문제

ATM 실버 분류

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	31857	20279	16972	65.114%

문제

인하은행에는 ATM이 1대밖에 없다. 지금 이 ATM앞에 N명의 사람들이 줄을 서있다. 사람은 1번부터 N번까지 번호가 매겨져 있으며, i번 사람이 돈을 인출하는데 걸리는 시간은 P_i분이다.

사람들이 줄을 서는 순서에 따라서, 돈을 인출하는데 필요한 시간의 합이 달라지게 된다. 예를 들어, 총 5명이 있고, $P_1 = 3$, $P_2 = 1$, $P_3 = 4$, $P_4 = 3$, $P_5 = 2$ 인 경우를 생각해보자. [1, 2, 3, 4, 5] 순서로 줄을 선다면, 1번 사람은 3분만에 돈을 뽑을 수 있다. 2번 사람은 1번 사람이 돈을 뽑을 때 까지 기다려야 하기 때문에, 3+1 = 4분이 걸리게 된다. 3번 사람은 1번, 2번 사람이 돈을 뽑을 때까지 기다려야 하기 때문에, 총 3+1+4 = 8분이 필요하게 된다. 4번 사람은 3+1+4+3 = 11분, 5번 사람은 3+1+4+3+2 = 13분이 걸리게 된다. 이 경우에 각 사람이 돈을 인출하는데 필요한 시간의 합은 3+4+8+11+13 = 39분이 된다.

줄을 [2, 5, 1, 4, 3] 순서로 줄을 서면, 2번 사람은 1분만에, 5번 사람은 1+2 = 3분, 1번 사람은 1+2+3 = 6분, 4번 사람은 1+2+3+3 = 9분, 3번 사람은 1+2+3+3+4 = 13분이 걸리게 된다. 각 사람이 돈을 인출하는데 필요한 시간의 합은 1+3+6+9+13 = 32분이다. 이 방법보다 더 필요한 시간의 합을 최소로 만들 수는 없다.

줄을 서 있는 사람의 수 N과 각 사람이 돈을 인출하는데 걸리는 시간 P_i가 주어졌을 때, 각 사람이 돈을 인출하는데 필요한 시간의 합의 최솟값을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 사람의 수 N($1 \le N \le 1,000$)이 주어진다. 둘째 줄에는 각 사람이 돈을 인출하는데 걸리는 시간 P_i가 주어진다. ($1 \le P_i \le 1,000$)

출력

첫째 줄에 각 사람이 돈을 인출하는데 필요한 시간의 합의 최솟값을 출력한다.

예제 입력 1 복사

```
5
3 1 4 3 2
```

예제 출력 1 복사

```
32
```

해당 문제에서 신경써야 할 껀 2개정도로 정리되는 것 같다.
하나는 sort를 해야 된 다는 것
두번째는 값을 저장하고 누적해서 더해야 된 다는 것 이였다.

```
N = int(input())
nums = list(map(int, input().split()))

result = 0
late = 0
if N == 1:
    print(nums[0])
else:
    nums.sort()
    for i in range(N):
        result += (nums[i] + late)
        late += nums[i]
    print(result)
```

위의 내용을 코드로 구현하였다.

'**Algorithm**' 카테고리의 다른 글

[Algorithm] 13 강 : 그리디 유형 문제풀이 + 백준 알고리즘 11399번 ATM문...

[Algorithm] 12 강 : 그리디 알고리즘 개요(탐욕법)

[Algorithm] 11 강 : 자주 사용하는 라이브러리(유용한 라이브러리)

[Algorithm] 10 강 : 파이썬 문법 - 함수

[Algorithm] 9 강 : 파이썬 문법 - 반복문

[Algorithm] 8 강 : 파이썬 문법 - 조건문

Greed 문제풀이 Greed 유형 문제 풀이 greedy 문제 풀이 그리디 문제풀이 그리디 유형 문제풀이 백준 11399 백준 알고리즘 11399

나무늘보스

혼자 끄적끄적하는 블로그 입니다.