1. 그리디 알고리즘

:지금 당장 좋은 것만 고르는 방법

특징: 암기 없이도 풀 수 있음.

‘가장 큰 순서대로’ ‘가장 작은 순서대로’의 기준을 몰래 제시.

예제3-1 500,100,50,10원으로 N을 거슬러 줄 수 있는 최소 개수

나의 풀이:

N=int(input())

cnt=0

if N/500 !=0:

        cnt+=N/500

        N%=500

if N/100 !=0:

        cnt+=N/100

        N%=100

if N/50 !=0:

        cnt+=N/50

        N%=50

if N/10 !=0:

        cnt+=N/10

        N%=10

print(cnt)

결과:

1260

7.32

리뷰: 하나하나 비교하지 말고 반복되므로 동전들을 리스트에 넣으면 된다.

결과값을 정수로 받고싶으면 /말고 //사용.

정답

N=int(input())

cnt=0

coin=[500,100,50,10]

for c in coin:

        cnt+=N//c

        N%=c

print(cnt)

포인트: 알고리즘 해법이 정당한지 검토하기.

(만약 거스름돈이 500 400 100 70원이었다면,,, )

3-2)큰 수 법칙

다양한 수의 배열이 있을 때 주어진 수를 M번 더하여 가장 큰 수를 만들기.

단, 배열의 특정 인덱스가 연속하여 K번을 초과하여 더할 수 없음.

풀이1)

6 6 6 5 6 6 6 5 6 이라면

무언가 하나 처리할때마다 m-=1을 해준다. m==0이면 반복문을 빠져나온다.

그 안에서 6은 k만큼 반복해서 더해주고, 반복문을 빠져나와 5를 한번 더해준다.

틀린 답:

n,m,k=map(int,input().split())

a=list(map(int,input().split()))

cnt=0

sum=0

a.sort()

first=a[n-1]

second=a[n-2]

while True:

        if m==0:

                break <<틀렸음

        for i in range(k):

                sum+=first

                m-=1

        sum+=second

        m-=1

print(sum)

-리뷰:

if m==0:

                break

이것은 first를 더하기 전에, second를 더하기 전에 두번 써야 함.

m==0이 될때까지 중간중간 확인하는 과정.

문제 이해도 잘못했음. 최대한 큰 수를 만들기 위해선

5 5 5 4 4 4 5 5 5 이런식이 아니라

5 5 5 4 5 5 5 4 이런식.

올바른 답:

n,m,k=map(int,input().split())

a=list(map(int,input().split()))

cnt=0

sum=0

a.sort()

first=a[n-1]

second=a[n-2]

while True:

        if m==0:

                break

        for i in range(k):

                sum+=first

                m-=1

        if m==0:

                break

        sum+=second

        m-=1

print(sum)

풀이2)반복되는 수열

k=3 m=9라고 했을 때

6 6 6 5 6 6 6 5 6

수열 하나가 6 6 6 5이다

이런식으로 반복. 반복횟수는 k+1이 된다

수열 반복 횟수는 m/(k+1)이다.

6이 등장한 횟수: (m/(k+1))\*k + m%(k+1)

5가 등장한 횟수는 수열 끝에만 한번씩 등장하기 때문에 수열반복횟수와 같다.

-먼저 반복되는 규칙이 있는지 확인하고, 수열로 풀어도 된다.

-m/k+1 아니고 m/(k+1)괄호 빼먹지 말자

n,m,k=map(int,input().split())

a=list(map(int,input().split()))

a.sort()

first=a[n-1]

second=a[n-2]

count=int(m/(k+1)) #수열 반복 횟수

f=count\*k+(m%(k+1)) #first등장 횟수

s=count #second 등장 횟수

re= f\*first + s\*second

print(re)

96p 3.숫자 카드 게임

N\*M배열에서 각 행마다 작은 숫자들을 고름.

그 숫자중 가장 큰 숫자를 고름

ex) 3 3

3 1 2 ->1

4 1 4 ->1

2 2 2 ->2 =>2

나의 풀이(정답)-2중 반복문으로 풀었다.

n,m=map(int,input().split())

min\_list=[]

for i in range(n):

        min=10001

        a=list(map(int,input().split()))

        for ain in a:

                if ain <min:

                        min=ain

        min\_list.append(int(min))

print(max(min\_list))

-min=10001을 전역변수로 설정해놨더니 답이 1이 나왔음. 이유는 for ain in a에서 min의 값이 행이 바뀔때마다 갱신되지 않았기 때문이다. 따라서 for문 안에 선언해주어야 한다.

 min=10001

**-min/max갱신**

계속 갱신해가는 과정을

        for ain in a:

                if ain <min:

                        min=ain

(전,c++에서 사용했던 방법)

        for ain in a:

                min\_a=min(ain,min\_a)

**(후)**

**3줄에서 2줄로 간단하게 바꿀 수 있다.**

max갱신도 마찬가지.

리스트에 넣어 비교하는 것

min\_list=[]

..

        min\_list.append(int(min))

(전)

대신

        for ain in a:

                min\_a=min(ain,min\_a)

        re=max(re,min\_a) <<행 하나가 끝날때마다 max 갱신

(후)

답1) min()이용

3 1 2에서 1을 고르고

max(0,1)에서 1쪽이 re에 들어간다.

그걸 행렬을 흝으면서 반복한다.

전 행에 남아있던 max값은 다음 행에서 또 비교한다.

n,m=map(int,input().split())

min\_list=[]

re=0

for i in range(n):

        a=list(map(int,input().split()))

        min\_a=min(a) #행 중 가장 작은 수들의 모임

        re=max(re,min\_a) #?, 가장 큰 수 찾기

print(re)

답2)2중 반복문

-행마다 min이 무엇인지 비교하니까 min은 반복문 안에서 매번 갱신해준다.

max는 전체에서 비교하므로 갱신이 필요 없다.

n,m=map(int,input().split())

re=0

for i in range(n):

        a=list(map(int,input().split()))

        min\_a=1001 #min, max를 행마다 갱신, min은 행마다 초기화

        for ain in a:

                min\_a=min(ain,min\_a)

        re=max(re,min\_a)

print(re)

4. 1이 될 때까지

n이 1이 될 때까지 다음 두 과정 중 하나를 반복적으로 선택하여 수행한다.

1. N-1

2. N나누기 K

만약 n=17, k=4라면

1. 17-1=16

2. 16/4 =4

3. 4/4=1

총 3번이다.

큰 수로 나누어서 답을 빠르게 최소화하는 그리디 문제.

1조건보다 2조건이 더 빠르게 최소화 할 수 있으므로

2조건을 쓸 수 있는지 확인하고 (나눌시 나머지 0) 되면 나눈다.

안되면 -1을 해준다.

내 답안(정답)

n,k=map(int,input().split())

cnt=0

while True:

        if n%k==0:

                n/=k

                cnt+=1

        else :

                n-=1

                cnt+=1

        if n==1:

                break

print(cnt)

수행시간 출력하기

import time

start\_time=time.time() #측정시작

end\_time=time.time() #종료

print("time: ",end\_time-start\_time) #수행시간 출력

수행시간이 0초가 나온다..

암기 필요 알고리즘

정렬 라이브러리/플로이드 워셜/다익스트라 알고리즘