21/01/05

1. 멀쩡한 사각형

**문제 설명**

가로 길이가 Wcm, 세로 길이가 Hcm인 직사각형 종이가 있습니다. 종이에는 가로, 세로 방향과 평행하게 격자 형태로 선이 그어져 있으며, 모든 격자칸은 1cm x 1cm 크기입니다. 이 종이를 격자 선을 따라 1cm × 1cm의 정사각형으로 잘라 사용할 예정이었는데, 누군가가 이 종이를 대각선 꼭지점 2개를 잇는 방향으로 잘라 놓았습니다. 그러므로 현재 직사각형 종이는 크기가 같은 직각삼각형 2개로 나누어진 상태입니다. 새로운 종이를 구할 수 없는 상태이기 때문에, 이 종이에서 원래 종이의 가로, 세로 방향과 평행하게 1cm × 1cm로 잘라 사용할 수 있는 만큼만 사용하기로 하였습니다.  
가로의 길이 W와 세로의 길이 H가 주어질 때, 사용할 수 있는 정사각형의 개수를 구하는 solution 함수를 완성해 주세요.

제한사항

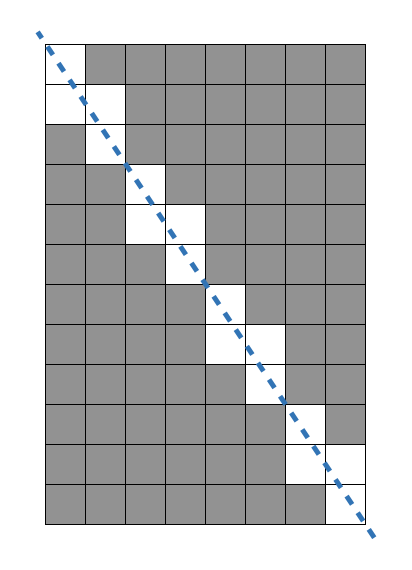
* W, H : 1억 이하의 자연수

**입출력 예**

| **W** | **H** | **result** |
| --- | --- | --- |
| 8 | 12 | 80 |

입출력 예 설명

입출력 예 #1  
가로가 8, 세로가 12인 직사각형을 대각선 방향으로 자르면 총 16개 정사각형을 사용할 수 없게 됩니다. 원래 직사각형에서는 96개의 정사각형을 만들 수 있었으므로, 96 - 16 = 80 을 반환합니다.



* 내 답

using namespace std;

long long solution(int w,int h) {

long long answer = 0;

for(int i=1;i<w;i++)

answer+=(long long)(h\*i)/w;

answer\*=2;

return answer;

}

처음에는 answer+=(long long)((double)h/w)\*i);

처럼 했는데, 캐스팅과정에서 무언가 오류가 있는 것 같아서

아예 연산 순서를 바꾸었다. 캐스팅은 최소화하여 사용해야 할 것 같다.

* 남의 답

#include <iostream>

using namespace std;

int GCD(int a, int b){

if(a == 0) return b;

return GCD(b % a, a);

}

long long solution(int w,int h)

{

long long answer = 1;

int gcd = GCD(w, h);

cout << w + h - gcd;

answer = ((long)w \* h) - ((long)w + h - gcd);

return answer;

}

이걸 O(1)시간에 풀어내는 것도 대단한 것 같다.

출처: [https://leedakyeong.tistory.com/entry/프로그래머스-멀쩡한-사각형-in-python](https://leedakyeong.tistory.com/entry/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%EB%A9%80%EC%A9%A1%ED%95%9C-%EC%82%AC%EA%B0%81%ED%98%95-in-python) [슈퍼짱짱]

1. 프린터

###### 문제 설명

일반적인 프린터는 인쇄 요청이 들어온 순서대로 인쇄합니다. 그렇기 때문에 중요한 문서가 나중에 인쇄될 수 있습니다. 이런 문제를 보완하기 위해 중요도가 높은 문서를 먼저 인쇄하는 프린터를 개발했습니다. 이 새롭게 개발한 프린터는 아래와 같은 방식으로 인쇄 작업을 수행합니다.

1. 인쇄 대기목록의 가장 앞에 있는 문서(J)를 대기목록에서 꺼냅니다.

2. 나머지 인쇄 대기목록에서 J보다 중요도가 높은 문서가 한 개라도 존재하면 J를 대기목록의 가장 마지막에 넣습니다.

3. 그렇지 않으면 J를 인쇄합니다.

예를 들어, 4개의 문서(A, B, C, D)가 순서대로 인쇄 대기목록에 있고 중요도가 2 1 3 2 라면 C D A B 순으로 인쇄하게 됩니다.

내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 알고 싶습니다. 위의 예에서 C는 1번째로, A는 3번째로 인쇄됩니다.

현재 대기목록에 있는 문서의 중요도가 순서대로 담긴 배열 priorities와 내가 인쇄를 요청한 문서가 현재 대기목록의 어떤 위치에 있는지를 알려주는 location이 매개변수로 주어질 때, 내가 인쇄를 요청한 문서가 몇 번째로 인쇄되는지 return 하도록 solution 함수를 작성해주세요.

##### 제한사항

* 현재 대기목록에는 1개 이상 100개 이하의 문서가 있습니다.
* 인쇄 작업의 중요도는 1~9로 표현하며 숫자가 클수록 중요하다는 뜻입니다.
* location은 0 이상 (현재 대기목록에 있는 작업 수 - 1) 이하의 값을 가지며 대기목록의 가장 앞에 있으면 0, 두 번째에 있으면 1로 표현합니다.

##### 입출력 예

| **priorities** | **location** | **return** |
| --- | --- | --- |
| [2, 1, 3, 2] | 2 | 1 |
| [1, 1, 9, 1, 1, 1] | 0 | 5 |

##### 입출력 예 설명

예제 #1

문제에 나온 예와 같습니다.

예제 #2

6개의 문서(A, B, C, D, E, F)가 인쇄 대기목록에 있고 중요도가 1 1 9 1 1 1 이므로 C D E F A B 순으로 인쇄합니다.

* 내 답

#include <string>

#include <vector>

#include <map>

#include <deque>

#include <iostream>

#include <algorithm>

using namespace std;

int solution(vector<int> priorities, int location) {

int answer = 0;

int size = priorities.size();

vector<int> prior\_t;

prior\_t.assign(priorities.begin(),priorities.end());

deque<int> array;

for(int i=0;i<size;i++)

array.push\_back(i);

while(!array.empty()) {

if(priorities[array[0]] == \*max\_element(prior\_t.begin(),prior\_t.end())) {

prior\_t[array[0]] = 0;

answer++;

if(array[0] == location)

return answer;

array.pop\_front();

continue;

}

array.push\_back(array[0]);

array.pop\_front();

}

return answer;

}

전의 다른 사람의 풀이에서 봤던 index 활용을 해보았다. 그리고 여기서 새로 알게 된 것은

벡터,큐 등에서 최대값을 구할 때 #include <algorithm> 알고리즘 헤더를 추가한 후

\*max\_element(vector.begin(),vector.end())구간을 넣어주면 최대, 최소 값을 구할 수 있는 것이다.

다른 사람의 best 풀이도 나와 비슷하였다.