



기하 활용 문제 해결 1

1

기하 관련 지식 활용해 SW 문제 풀이하는 방법 익히기

01. 문제에서 사용하는 정의 이해: 정수 좌표 가지는 직각삼각형

02. 첫 번째 알고리즘

03. 두 번째 알고리즘

04. 중간고사 점수 확인

05. 실습 문제 풀이 & 질의 응답

<왜 기하/수학식의 활용 예를 보는가? >

- SW/HW 시스템 설계 시 수학식 활용하여 문제 해결되는 경우 많음
- 적절한 수학식 활용 (기하 포함) → (i) 좋은 성능의 시스템 설계 + (ii) 설계 시간 단축
- 기하 관련 지식은 컴퓨터 그래픽 분야 및 수학 활용 SW 개발에 많이 활용됨 (예: Octave, Matlab)

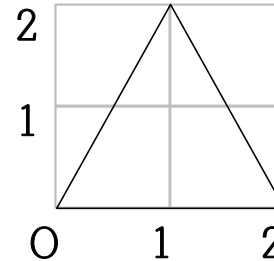
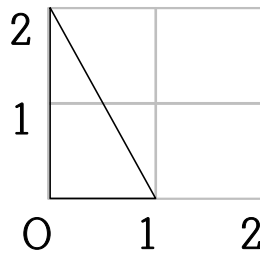


문제 정의: 정수 좌표 가지는 직각 삼각형

2

이 문제는 다음 조건 ① ~ ⑤ 만족하는 삼각형 $\triangle OAB$ 에 대한 문제이다.

- ① 점 O 는 **원점 $(0,0)$** 이다. 즉 삼각형의 세 점 중 하나는 반드시 원점이어야 한다.
- ② 점 $A(x_1, y_1)$ 와 $B(x_2, y_2)$ 는 **정수 좌표**를 가진다. 즉 x_1, y_1, x_2, y_2 는 모두 정수이다.
- ③ 양의 정수 N 이 입력으로 주어질 때, A 와 B 의 좌표는 $0 \sim N$ 사이에 있어야 한다. 즉 $0 \leq x_1, y_1, x_2, y_2 \leq N$ 이다.
- ④ ($\triangle OAB$ 가 삼각형이 되기 위해) **OAB 세 점은 서로 다른 좌표**를 가져야 한다.
- ⑤ 삼각형 $\triangle OAB$ 의 각 중 **하나는 직각**이다. 즉 $\angle A (= \angle OAB)$, $\angle B (= \angle OBA)$, $\angle O (= \angle AOB)$, 중 하나는 직각이다.



(Q) $N = 2$ 일 때,
왼쪽의 두 삼각형은
조건 ① ~ ⑤를
만족하는가?

양의 정수 N 이 입력으로 주어질 때, 조건 ① ~ ⑤를 만족하는 **서로 다른 삼각형** $\triangle OAB$ 의 개수를 세는 방법을 설계 하시오. ※ 같은 3개 점을 다른 순서로 나열한 삼각형은 같은 삼각형으로 본다. 즉 $\triangle Oab$ 와 $\triangle Oba$ 는 같다. 예를 들어 $(0,0)-(0,2)-(1,0)$ 는 $(0,0)-(1,0)-(0,2)$ 와 같으므로 하나의 삼각형으로 count해야 한다.



문제 풀이 과정

- ① 이번 시간 문제를 잘 이해하기 위해 간단한 경우부터 시작해 문제 이해하고 답 이끌어내 보기



- ② 이번 시간 문제에 대한 풀이 방법 생각



- ③ 생각한 풀이 방법을 보다 효율적으로 개선

문제 명확히 이해하는데 도움

이 과정에서 문제에 대한 해법도
생각해 낼 수 있음

조건 ① ~ ⑤ 만족하는 서로 다른 삼각형의 개수를 세는 방법을 설계하시오.

① 점 O는 **원점 (0,0)**이다. 즉 삼각형의 세 점 중 하나는 반드시 원점이어야 한다.

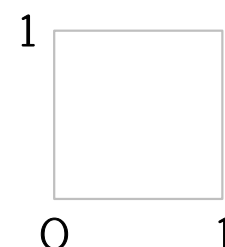
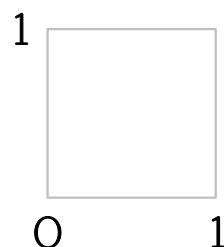
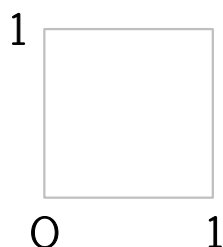
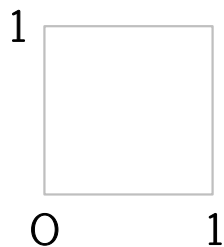
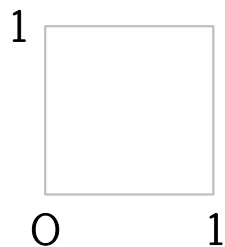
② 점 A(x1,y1)와 B(x2,y2)는 **정수 좌표**를 가진다. 즉 x1,y1,x2,y2 는 모두 정수이다.

③ 양의 정수 N이 입력으로 주어질 때, A와 B의 좌표는 0~N 사이에 있어야 한다. 즉 $0 \leq x1, y1, x2, y2 \leq N$ 이다.

④ ($\triangle OAB$ 가 삼각형이 되기 위해) **OAB 세 점은 서로 다른 좌표**를 가져야 한다.

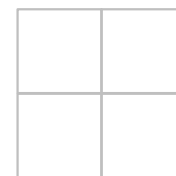
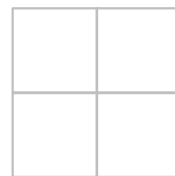
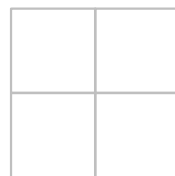
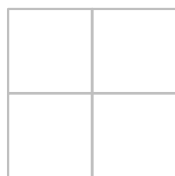
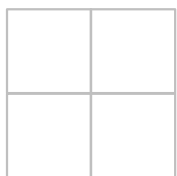
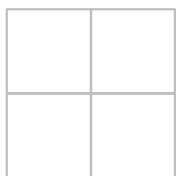
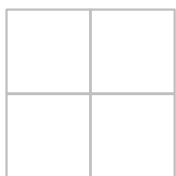
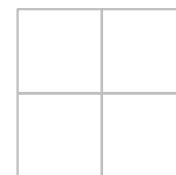
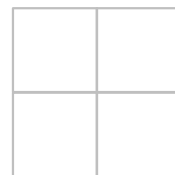
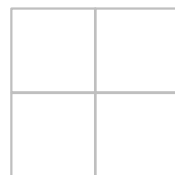
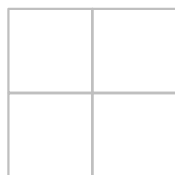
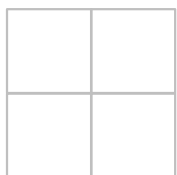
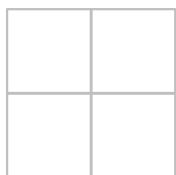
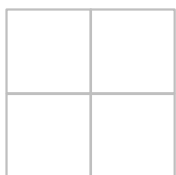
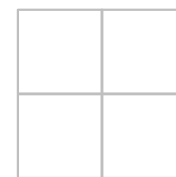
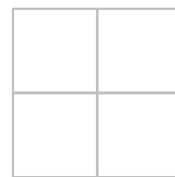
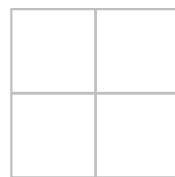
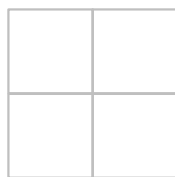
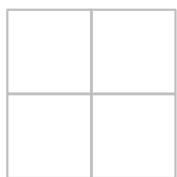
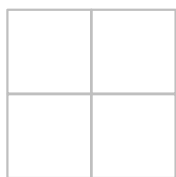
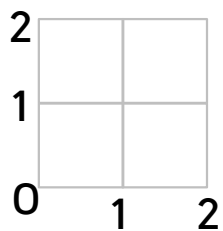
⑤ 삼각형 $\triangle OAB$ 의 각 중 **하나는 직각**이다. 즉 $\angle A (= \angle OAB)$, $\angle B (= \angle OBA)$, $\angle O (= \angle AOB)$, 중 하나는 직각이다.

(1) $N = 1$ 일 때, 조건을 만족하는 서로 다른 직각 삼각형 $\triangle OAB$ 를 모두 그려보고 몇 개인지 세어 보시오.



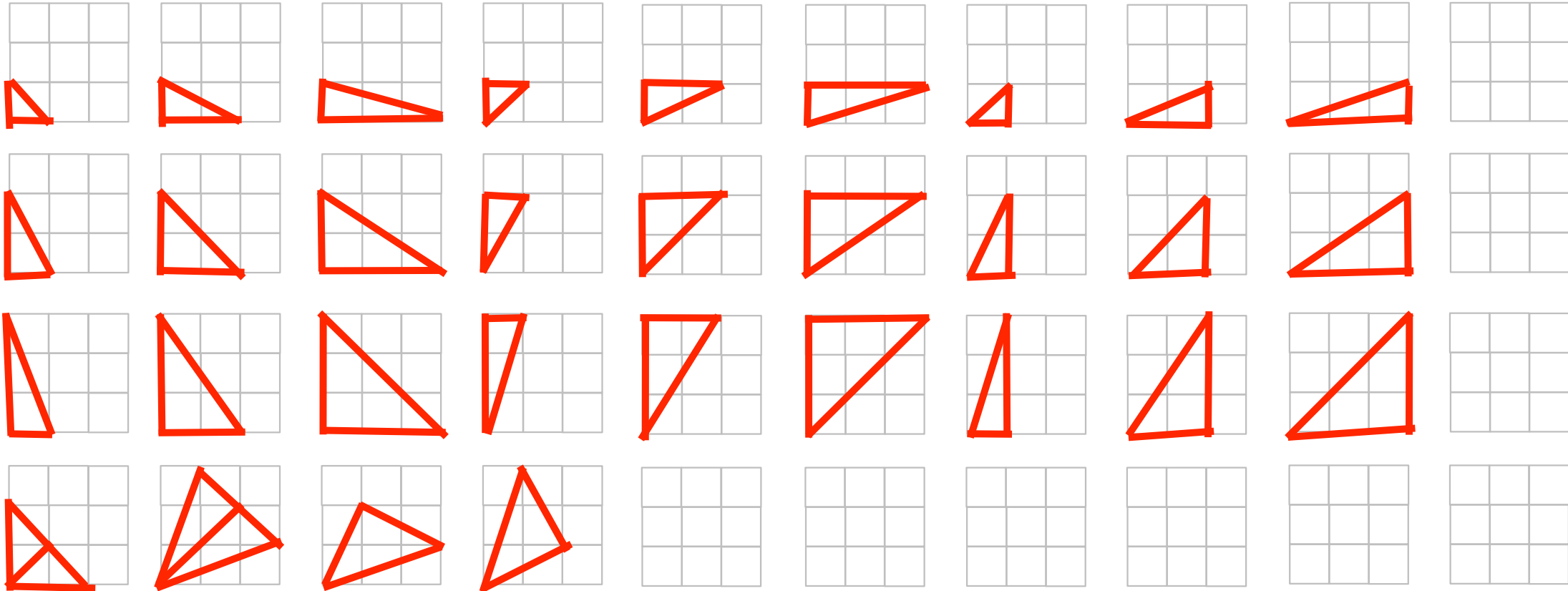
0~N의 정수 좌표 가지는 서로 다른 직각삼각형 $\triangle OAB$ 의 개수를 세는 방법을 설계하시오.

(2) $N = 2$ 일 때, 조건을 만족하는 서로 다른 직각 삼각형 $\triangle OAB$ 를 모두 그려보고 몇 개인지 세어보시오. 특히 앞 문제에서 구한 $N = 1$ 인 경우의 해는 $N = 2$ 인 경우의 해에도 포함됨을 고려하시오.



0~N의 정수 좌표 가지는 서로 다른 직각삼각형 $\triangle OAB$ 의 개수를 세는 방법을 설계하시오.

(3) $N = 3$ 일 때, 조건을 만족하는 서로 다른 직각 삼각형 $\triangle OAB$ 를 모두 그려보고 몇 개인지 세어보시오. 또한 일반적으로 임의의 N 값에 대해 어떻게 답을 구해야 할지도 생각해 보시오.



0~N의 정수 좌표 가지는 서로 다른 직각삼각형 $\triangle OAB$ 의 개수를 세는 방법을 설계하시오.

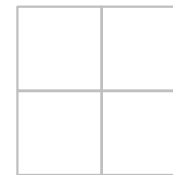
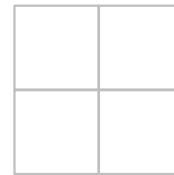
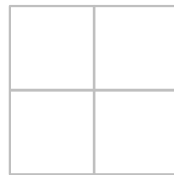
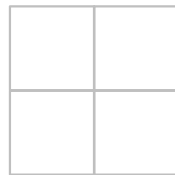
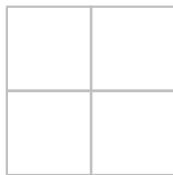
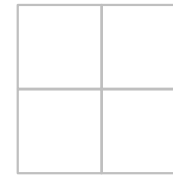
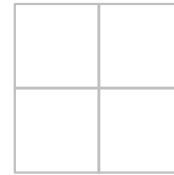
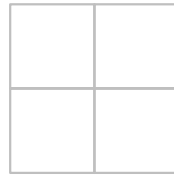
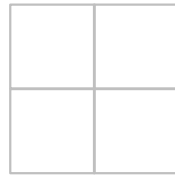
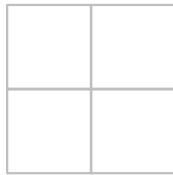
- (4) 임의의 N 값에 대해 답을 구하는 방법을 생각한 대로 기술해 보시오.
제안한 방법이 어떤 큰 N(>3) 값에 대해서도 정확한 답을 낼 수 증명할 수 있겠는가?



첫 번째 방법 (완전 탐색, brute-force)

8

For ① $0 \leq x_1, y_1, x_2, y_2 \leq N$ 범위에서 원점 아닌 서로 다른 두 점의 쌍 $A(x_1, y_1)$ 와 $B(x_2, y_2)$ 각각에 대해
② $\triangle OAB$ 의 세 각 $\angle A$, $\angle B$, $\angle O$ 중 직각인 것이 있다면 count



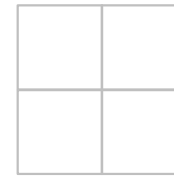
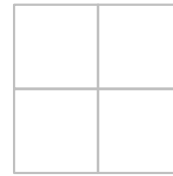
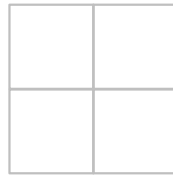
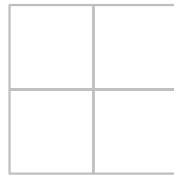
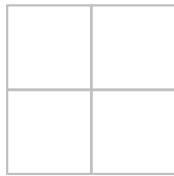


첫 번째 방법 (완전 탐색, brute-force)

9

For ① $0 \leq x_1, y_1, x_2, y_2 \leq N$ 범위에서 원점 아닌 서로 다른 두 점의 쌍 $A(x_1, y_1)$ 와 $B(x_2, y_2)$ 각각에 대해

② $\triangle OAB$ 의 세 각 $\angle A, \angle B, \angle O$ 중 직각인 것이 있다면 count



(5) 위 방법에서 ①은 for loop 등을 사용해 구현할 수 있다. 그렇다면 ②를 어떻게 구현할지 생각해 보자.

①에서 3개의 점 $O(0, 0), A(x_1, y_1), B(x_2, y_2)$ 를 정했다면 $\triangle OAB$ 가 직각삼각형임을 어떻게 확인하겠는가?



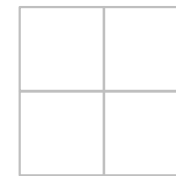
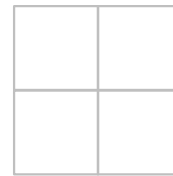
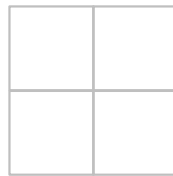
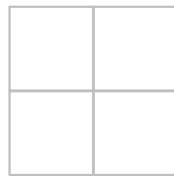
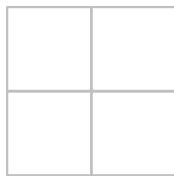
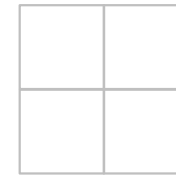
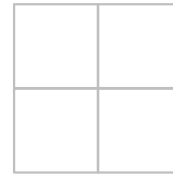
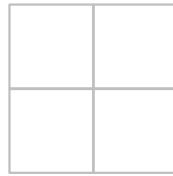
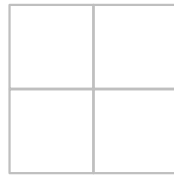
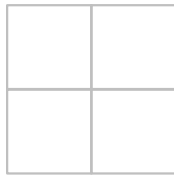
첫 번째 방법 (완전 탐색, brute-force): ①을 더 자세히 기술

10

for each $A(x_1, y_1)$ in $0 \leq x_1, y_1 \leq N$, such that $A \neq 0$
for each $B(x_2, y_2)$ in $0 \leq x_2, y_2 \leq N$, such that $B \neq 0$
if $A \neq B$ then

For ① $0 \leq x_1, y_1, x_2, y_2 \leq N$ 범위에서 원점 아닌 서로 다른 두 점의 쌍 $A(x_1, y_1)$ 와 $B(x_2, y_2)$ 각각에 대해

② $\triangle OAB$ 의 세 각 $\angle A, \angle B, \angle O$ 중 직각인 것이 있다면 count





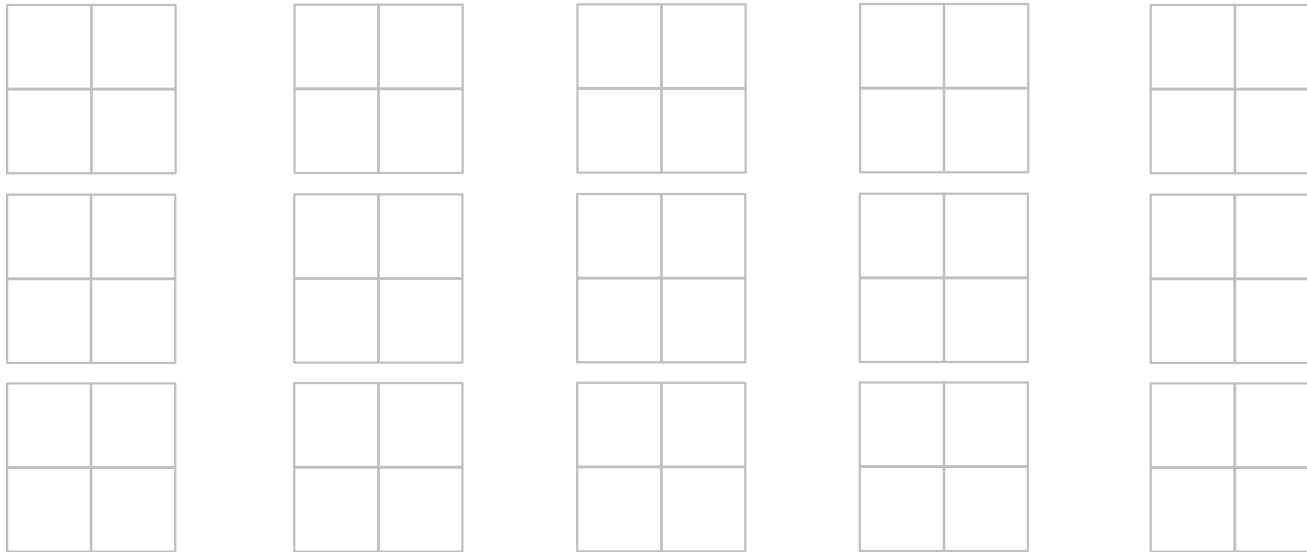
첫 번째 방법 (완전 탐색, brute-force): ①을 더 자세히 기술

11

for each $A(x_1, y_1)$ in $0 \leq x_1, y_1 \leq N$, such that $A \neq 0$
for each $B(x_2, y_2)$ in $0 \leq x_2, y_2 \leq N$, such that $B \neq 0$
if $A \neq B$ then

For ① $0 \leq x_1, y_1, x_2, y_2 \leq N$ 범위에서 원점 아닌 서로 다른 두 점의 쌍 $A(x_1, y_1)$ 와 $B(x_2, y_2)$ 각각에 대해

② $\triangle OAB$ 의 세 각 $\angle A, \angle B, \angle O$ 중 직각인 것이 있다면 count



(6) 위와 같은 방법을 사용한다면 같은 삼각형을 중복해서 2번 count할 수도 있다. 예를 들어 $A(3,3), B(4,4)$ 를 count한 후 $A(4,4), B(3,3)$ 을 다시 count하게 될 수도 있을 것이다. 이러한 중복 counting을 막기 위해서는 방법을 어떻게 수정해야 할까?



첫 번째 알고리즘의 Python 구현

12

```
def rightTriangle1(N):  
    count = 0  
    for A in range(1, (N+1)**2 - 1): # Choose A > 0  
        x1, y1 = A // (N+1), A % (N+1)  
        for B in range(A + 1, (N+1)**2): # Choose B > A  
            x2, y2 = B // (N+1), B % (N+1)  
            lmax, l1, l2 = maxNrest(x1**2 + y1**2, x2**2 + y2**2, (x1-x2)**2 + (y1-y2)**2)  
            if lmax == l1 + l2: count += 1  
    return count
```

```
def maxNrest(v1, v2, v3):  
    if v1 >= v2 and v1 >= v3: return v1, v2, v3  
    elif v2 >= v1 and v2 >= v3: return v2, v1, v3  
    else: return v3, v1, v2
```

먼저 위 코드가 무엇을 하는지 앞에서 본 수도코드와 비교하며 이해해 보시오.



첫 번째 알고리즘의 Python 구현

13

```
def rightTriangle1(N):
    count = 0
    for A in range(1, (N+1)**2 - 1): # Choose A > 0
        x1, y1 = A // (N+1), A % (N+1)
        for B in range(A + 1, (N+1)**2): # Choose B > A
            x2, y2 = B // (N+1), B % (N+1)
            lmax, l1, l2 = maxNrest(x1**2 + y1**2, x2**2 + y2**2, (x1-x2)**2 + (y1-y2)**2)
            if lmax == l1 + l2: count += 1
    return count

def maxNrest(v1, v2, v3):
    if v1 >= v2 and v1 >= v3: return v1, v2, v3
    elif v2 >= v1 and v2 >= v3: return v2, v1, v3
    else: return v3, v1, v2

if __name__ == "__main__":
    functions = [rightTriangle1]
    for f in functions:
        for i in range(1,5):
            print(i, f(i))
        print()
```

위 코드를 이해한 후에는 이를 실행해 보자. 이 함수는 RightTriangle.py에 있다. 먼저 **N=1~4 사이의 값에 대해 계산한 값이 올바른지** 비교하여 **확인**해 보시오. 그리고 그 아래의 코드로 시간도 측정해 보시오.

```
def rightTriangle1(N):
```

```
    count = 0
```

```
    for A in range(1, (N+1)**2 - 1): # Choose A > 0
```

```
        x1, y1 = A // (N+1), A % (N+1)
```

```
        for B in range(A + 1, (N+1)**2): # Choose B > A
```

```
            x2, y2 = B // (N+1), B % (N+1)
```

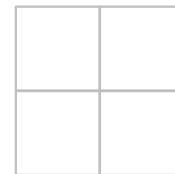
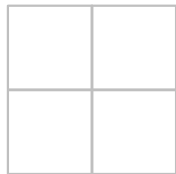
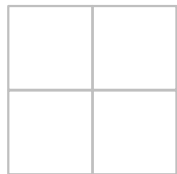
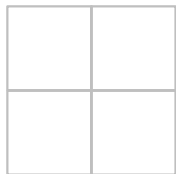
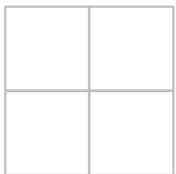
```
            lmax, l1, l2 = maxNrest(...)
```

```
            if lmax == l1 + l2: count += 1
```

```
    return count
```

A	B	개수
1	2 ~ (N+1)^2 - 1	(N+1)^2 - 1
2	3 ~ (N+1)^2 - 1	.
3	.	.
.	.	.
.	.	.
.	.	.
(N+1)^2 - 2	(N+1)^2 - 1	1

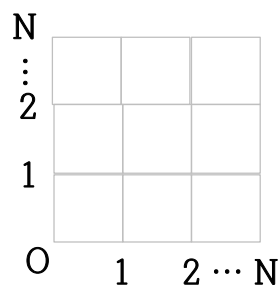
(7) (중복되는 경우를 제거하도록 최적화한) 첫 번째 알고리즘의 시간 복잡도에 대해 생각해 보자. 예를 들어 N=2라면 대략 얼마나 많은 서로 다른 A(x1, y1), B(x2, y2)의 쌍을 고려할 것인가?



$$\frac{(N^2 + 2N - 1)(N^2 + 2N)}{2}$$

=>

(8) 문제 (7)의 답을 임의의 N에 대해 일반화해 보자.



$$\frac{N^4}{2}$$

=>

$$N^4$$



(9) 첫 번째 방법의 수행 시간을 개선할 방법을 제안해 보시오.

for i = 1 ~ 3

for j = 1 ~ 2

이러면
 $3 * 2 = 6$

for i = 1 ~ 3

for j = 1 ~ f(i)

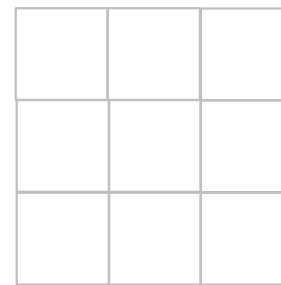
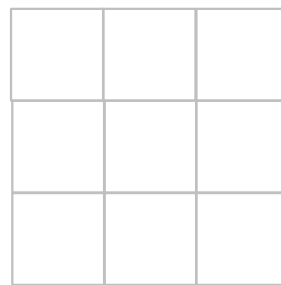
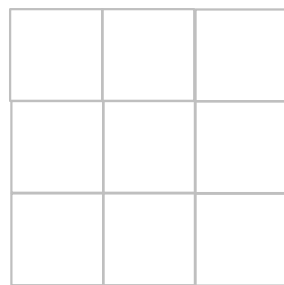
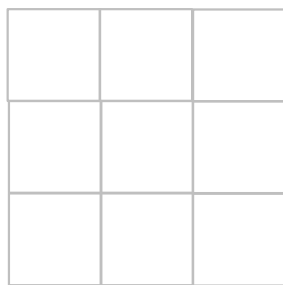
이러면
안쪽 j의 모든 연산 개수를 더해주는 것이
시간 복잡도가 된다!!



두 번째 방법 개요

16

- ① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count
- ② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq 0$
- ③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count



① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count

② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq 0$

③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

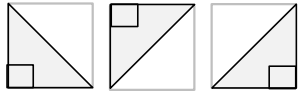
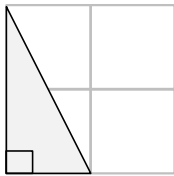
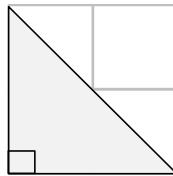
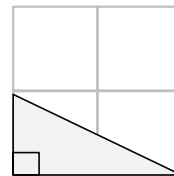
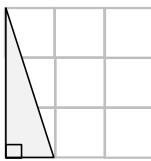
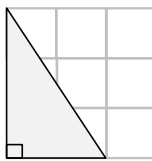
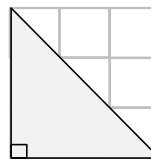
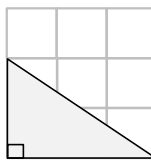
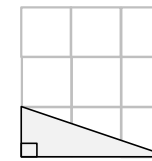
N	유형 ①: 직각이 x축 or y축에 인접한 경우 (누적해 더함)					유형 ②: 직각이 x/y축에 인접하지 않은 경우	
1							
	[1×1] 격자를 활용하는 3가지 경우						
2							
	3가지 경우 [1×2]	3가지 경우 [2×2]	3가지 경우 [2×1]				y=x에 대해 대칭
3							
	3 cases [1×3]	3 cases [2×3]	3 cases [3×3]	3 cases [3×2]	3 cases [3×1]	y=x에 대해 대칭	y=x에 대해 대칭

① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count

② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq O$

③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

(10) 임의 입력 N이 주어졌을 때, 유형 ①에 속하는 경우는 몇 가지라고 할 수 있는가?

N	유형 ①: 직각이 x축 or y축에 인접한 경우 (누적해 더함)				
1					
	[1×1] 격자를 활용하는 3가지 경우				
2					
	3가지 경우 [1×2]	3가지 경우 [2×2]	3가지 경우 [2×1]		
3					
	3 cases [1×3]	3 cases [2×3]	3 cases [3×3]	3 cases [3×2]	3 cases [3×1]

① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count

② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq O$

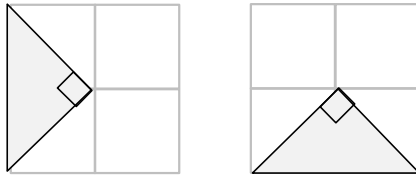
③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

N

유형 ②: 직각이 x/y축에 인접하지 않은 경우

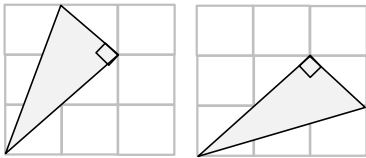
1

2

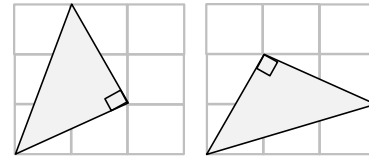


y=x에 대해 대칭

3

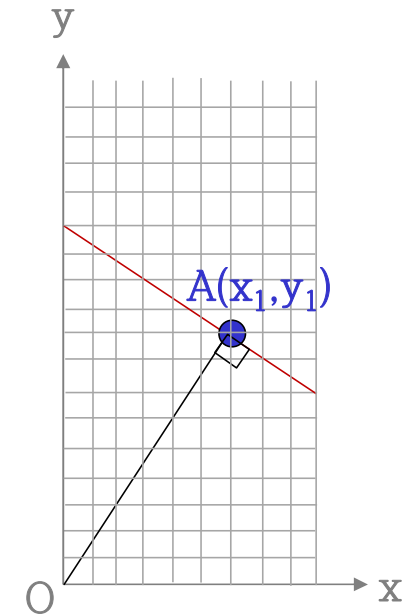
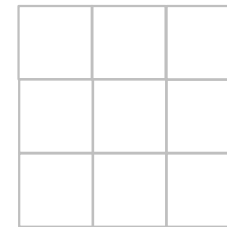
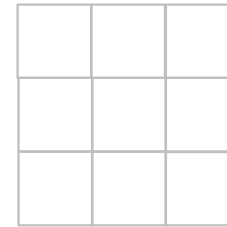


y=x에 대해 대칭



y=x에 대해 대칭

직각이 있는 점 $A(x_1, y_1)$ 는 $1 \leq x_1, y_1 \leq N$ 범위에서 선택한다. $x_1=0$ 이거나 $y_1=0$ 이면 유형 ①에 속하기 때문이다. 이렇게 A를 선택한 후 $\angle OAB$ 가 직각이 되도록 B를 선택한다.

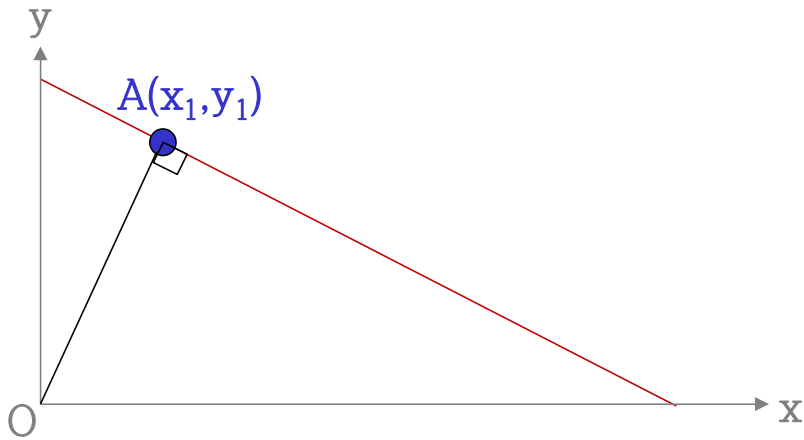


① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count

② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq O$

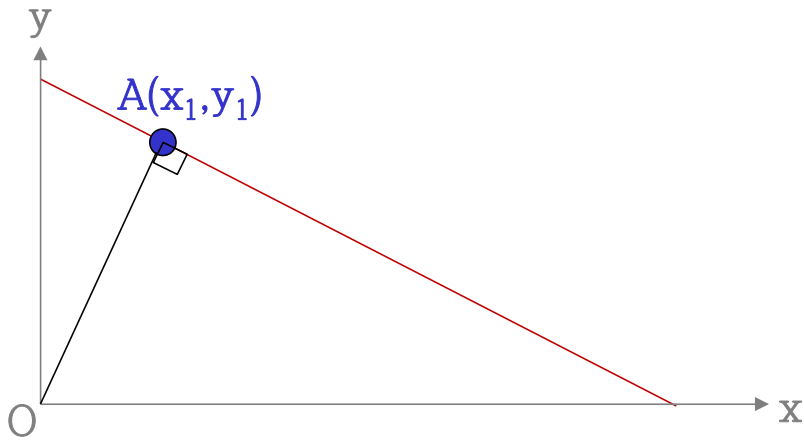
③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

③을 어떻게 구현할지 생각해 보자. 즉 점 $A(x_1, y_1)$ 를 $1 \leq x_1, y_1 \leq N$ 범위에서 선택한 후, $\angle OAB$ 가 직각이 되는 $B(x_2, y_2)$ 를 어떻게 선택해야 할까?



- ① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count
- ② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq O$
- ③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

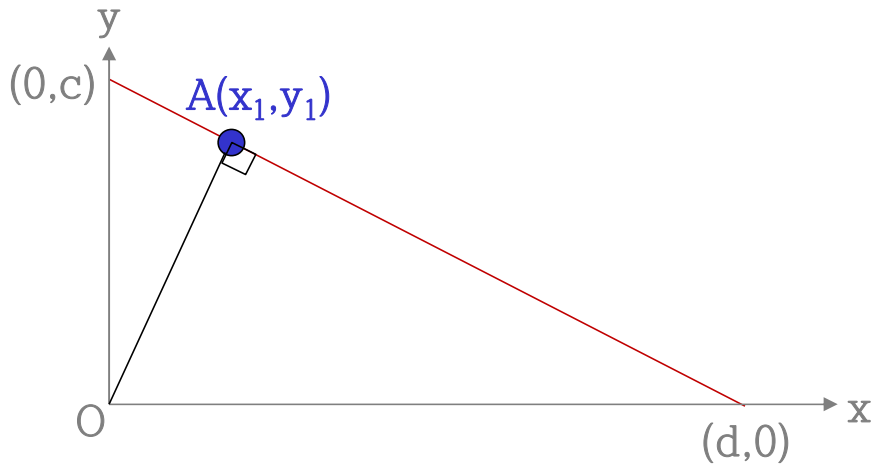
(11) 붉은 선의 방정식이 $y = ax + b$ 라면, a와 b는 각각 무엇인가?



$$y = -(x_1 / y_1)x + (x_1^2 + y_1^2) / y_1$$

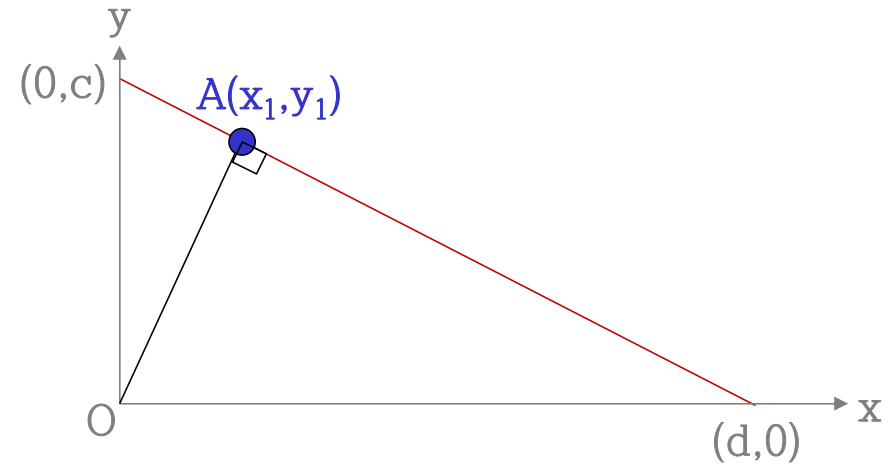
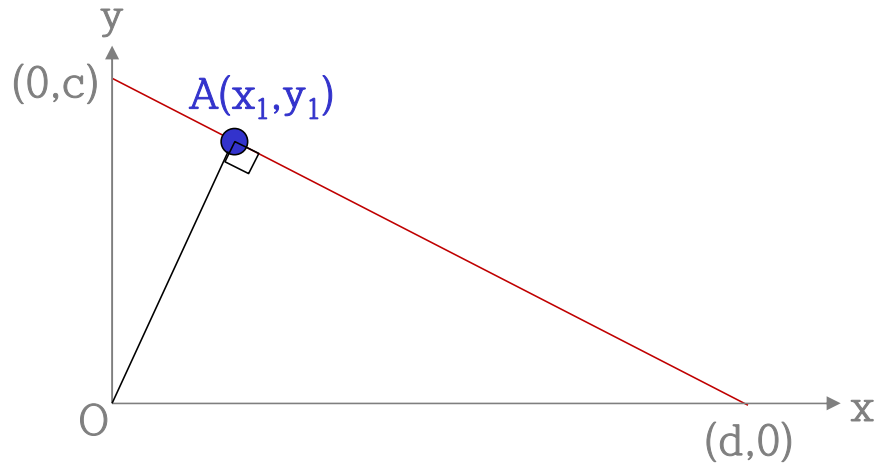
- ① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count
- ② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq O$
- ③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

(12) 붉은 선이 y축과 만나는 교점을 $(0, c)$ 라 하고, x축과 만나는 교점을 $(d, 0)$ 이라 할 때, c 와 d 는 무엇인가?

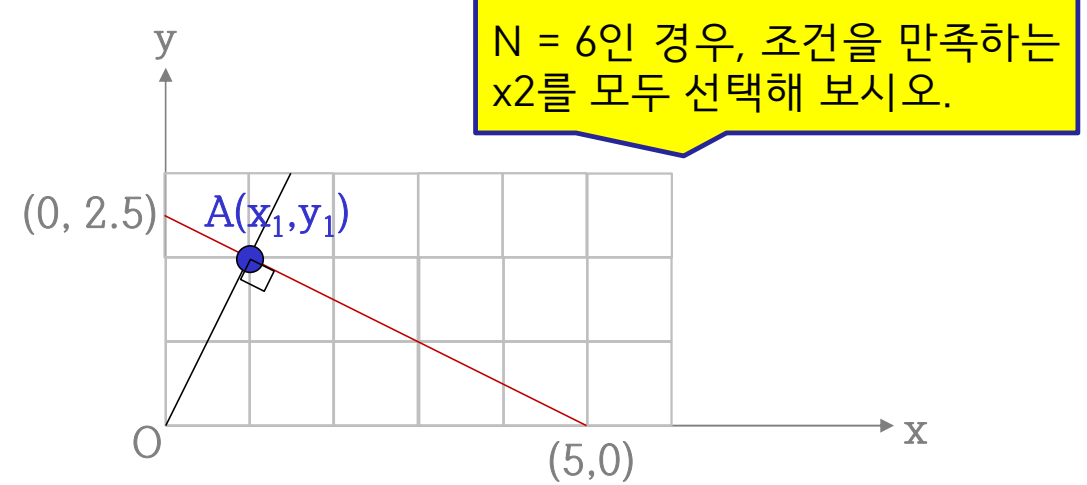
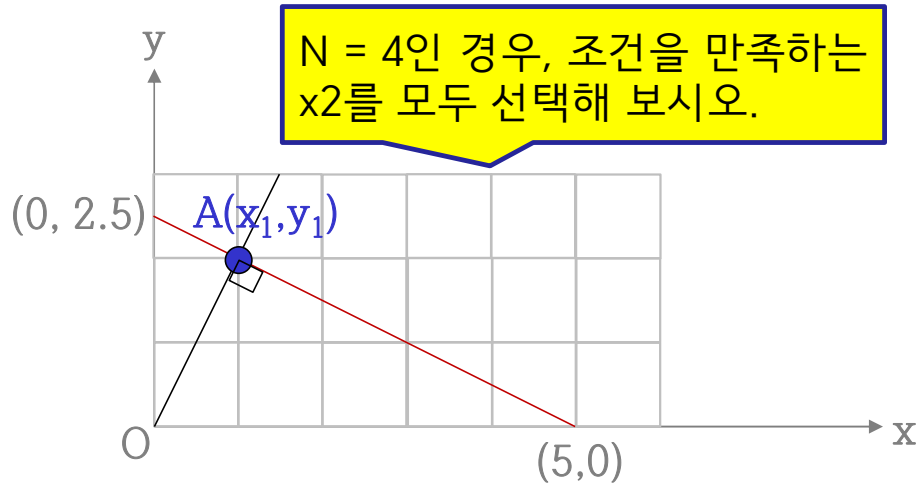


(13) c, d 값이 구해졌다고 가정하고, 붉은 선 위에서 A 오른쪽에 있는 점 $B(x_2, y_2)$ 의 수를 구해보자. x_2 는 $0 \leq x_2 \leq N$ 범위에 들어가야 하며, 또한 $x_1 < x_2 \leq d$ 이기도 해야 한다. 이러한 조건을 만족하는 정수 x_2 는 몇 개인가?

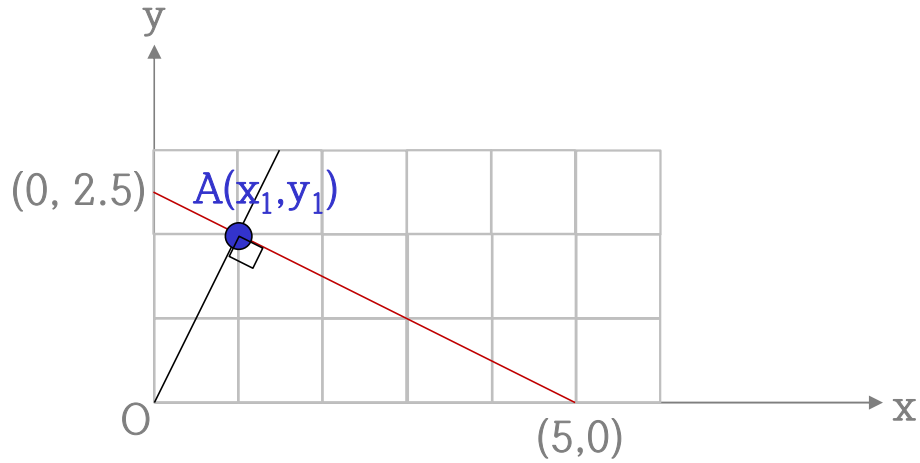
23



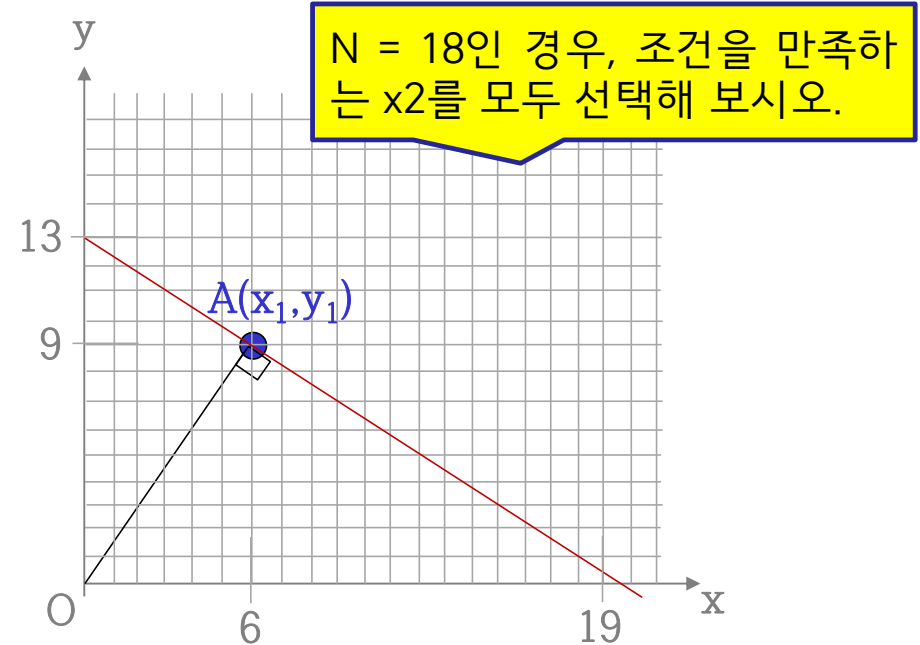
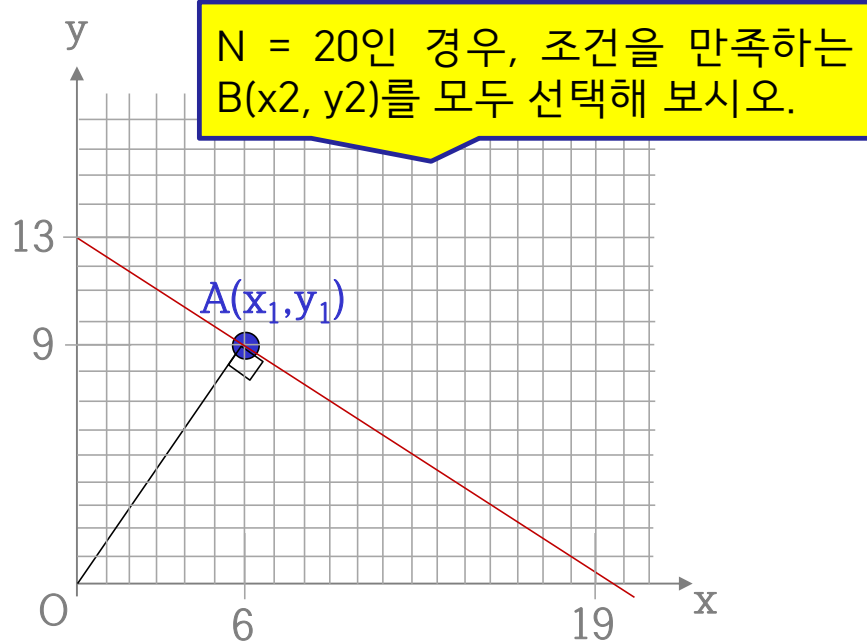
(13) c, d 값이 구해졌다고 가정하고, 붉은 선 위에서 A 오른쪽에 있는 점 $B(x_2, y_2)$ 의 수를 구해보자. x_2 는 $0 \leq x_2 \leq N$ 범위에 들어가야 하며, 또한 $x_1 < x_2 \leq d$ 이기도 해야 한다. 이러한 조건을 만족하는 정수 x_2 는 몇 개인가?



(14) $N = 5$ 라고 가정하자. 붉은 선 위에서 A 오른쪽에 있는 점을 $B(x_2, y_2)$ 라 할 때, $0 \leq x_2 \leq N$ 와 $x_1 < x_2 \leq d$ 범위를 만족하는 정수 x_2 는 $\{2, 3, 4, 5\}$ 이다. 이 중 대응되는 y_2 값도 정수인 경우는 어느 정도의 비율로 나오는가?

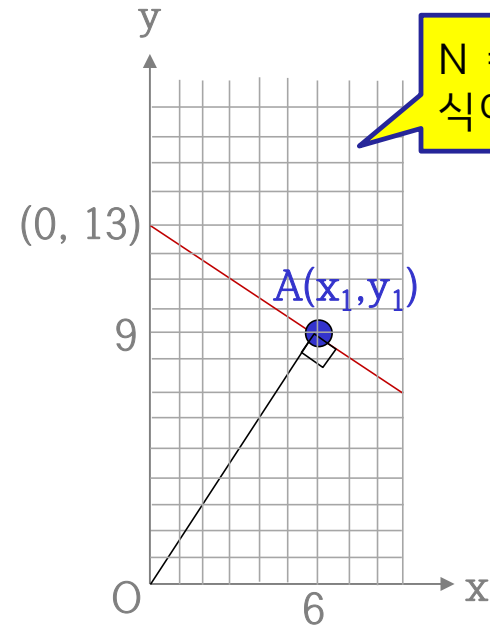
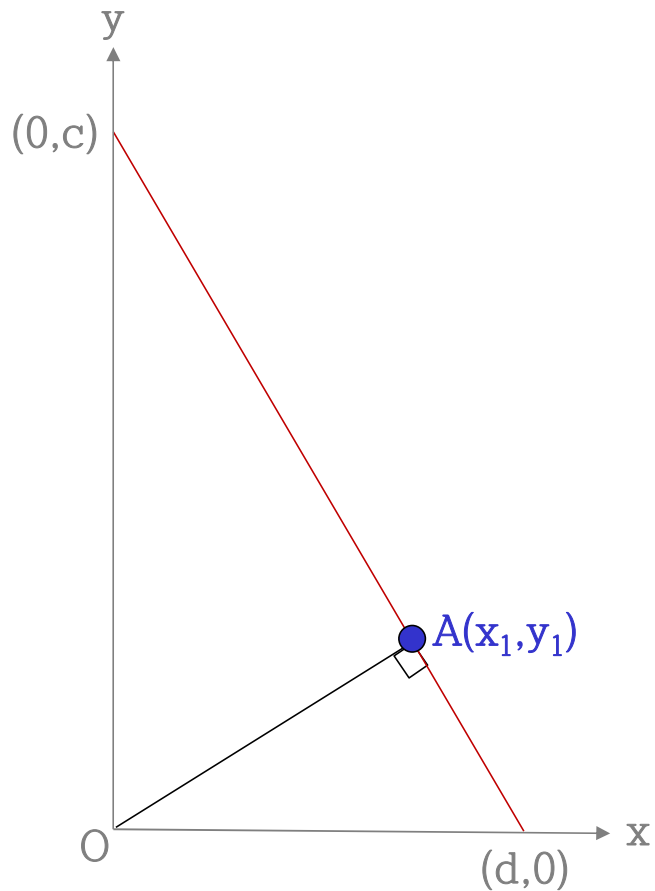


(15) 아래 예와 같이 조금 더 복잡한 경우에 대해 조건을 만족하는 점 $B(x_2, y_2)$ 의 개수를 세어보자. 조건을 만족하는 $\min(\lfloor d \rfloor, N) - x_1$ 개의 x_2 중 y_2 역시 정수인 경우는 어느 정도의 비율로 나오는가? 이로부터 임의의 점 $A(x_1, y_1)$ 이 선택되었을 때, A 의 오른쪽에 있으면서 $\angle OAB$ 를 직각으로 만드는 정수 좌표인 점 $B(x_2, y_2)$ 의 개수는 무엇이라고 할 수 있을까?



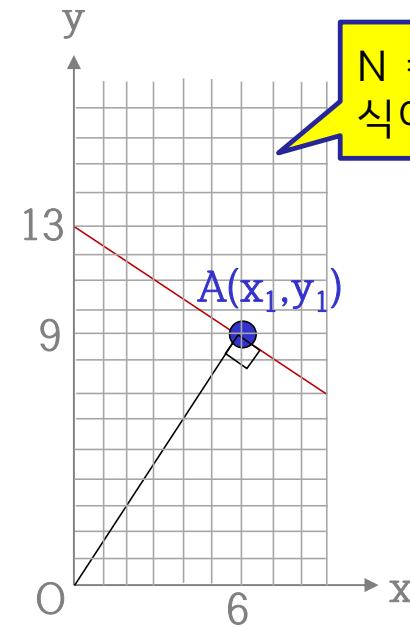
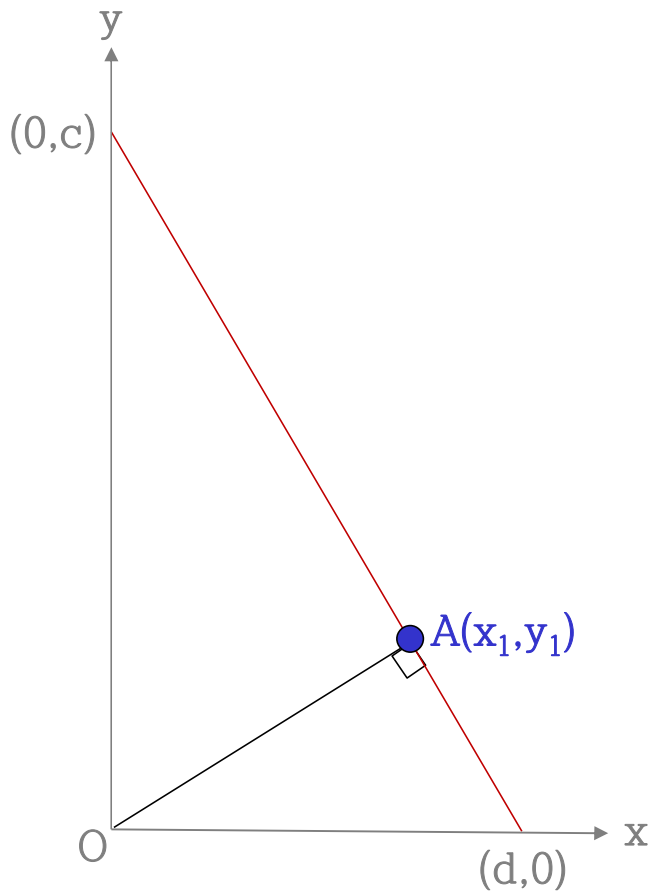
(16) 붉은 선 위에서 **A 왼쪽**에 있는 점 $B(x_2, y_2)$ 의 수를 구해보자. y_2 는 $0 \leq y_2 \leq N$ 범위에 들어가야 하며, 또한 $y_1 < y_2 \leq c$ 이기도 해야 한다. 이러한 조건을 만족하는 정수 y_2 는 몇 개인가?

27



$N = 20$ 이라 가정하고, 이끌어낸 식이 맞음을 검증해 보시오.

(17) 앞 문제에서 조건을 만족하는 정수 y_2 중에서 이에 대응되는 x_2 좌표 역시 정수인 경우의 비율은 어느 정도인가?
 이로부터 임의의 점 $A(x_1, y_1)$ 이 선택되었을 때, A 의 왼쪽에 있으면서 $\angle OAB$ 를 직각으로 만드는 정수 좌표인 점 $B(x_2, y_2)$ 의 개수는 무엇이라고 할 수 있을까?



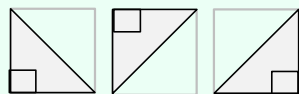
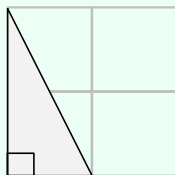
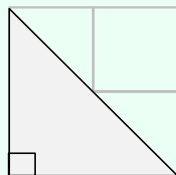
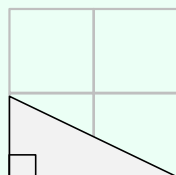
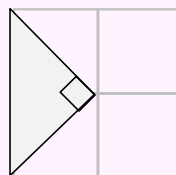
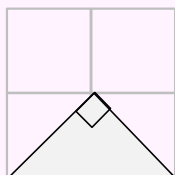
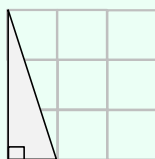
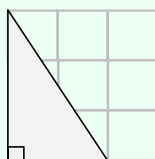
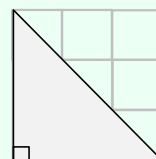
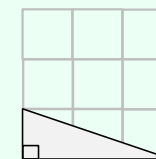
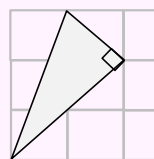
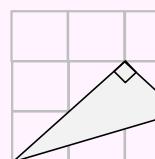
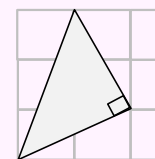
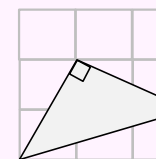
$N = 20$ 이라 가정하고, 이끌어낸 식이 맞음을 검증해 보시오.



두 번째 방법 version 2

29

- ① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count
- ② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq 0$
- ③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

N	유형 ①: 직각이 x축 or y축에 인접한 경우 (누적해 더함)					유형 ②: 직각이 x/y축에 인접하지 않은 경우			
1									
	[1×1] 격자를 활용하는 3가지 경우								
2									
	3가지 경우 [1×2]	3가지 경우 [2×2]	3가지 경우 [2×1]			y=x에 대해 대칭			
3									
	3 cases [1×3]	3 cases [2×3]	3 cases [3×3]	3 cases [3×2]	3 cases [3×1]	y=x에 대해 대칭		y=x에 대해 대칭	



OA와 수직인 직선과, 이 직선이 y축/x축과 만나는 교점 (0, c) (d, 0) 구하기

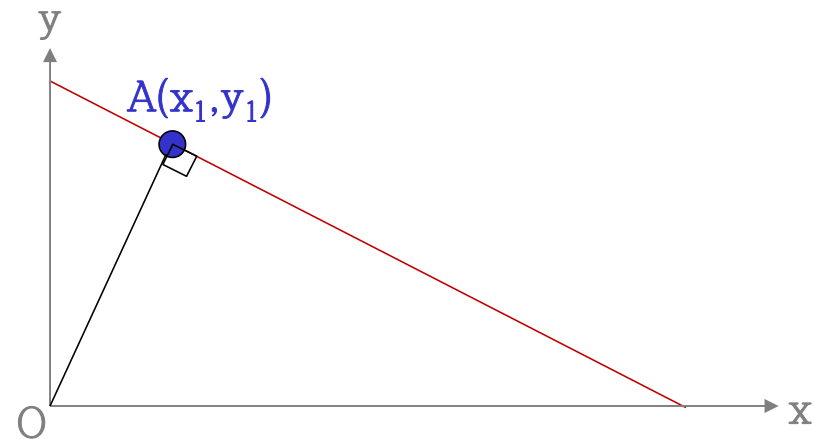
③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

A의 왼쪽으로

개의 $B(x_2, y_2)$ count

A의 오른쪽으로

개의 $B(x_2, y_2)$ count



두 번째 방법 version 2

31

① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count

② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq 0$

OA와 수직인 직선과, 이 직선이 y축/x축과 만나는 교점 $(0, c)$ $(d, 0)$ 구하기

③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count

A의 왼쪽으로

개의 $B(x_2, y_2)$ count

A의 오른쪽으로

개의 $B(x_2, y_2)$ count

$$\gcd(x_1, y_1) \sim \log(\min(x_1, y_1)) \quad 1 \leq x_1, y_1 \leq N$$

즉 평균 $N/2$

따라서 평균 $\sim \log(N/2)$

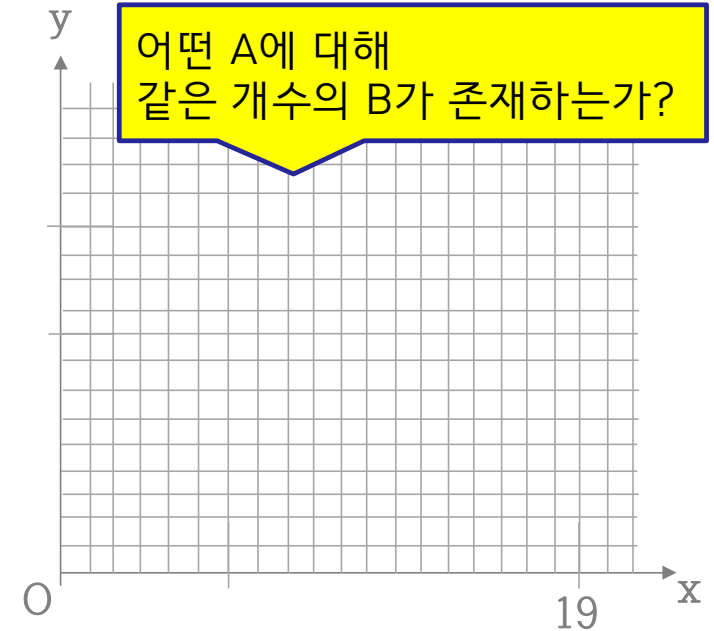
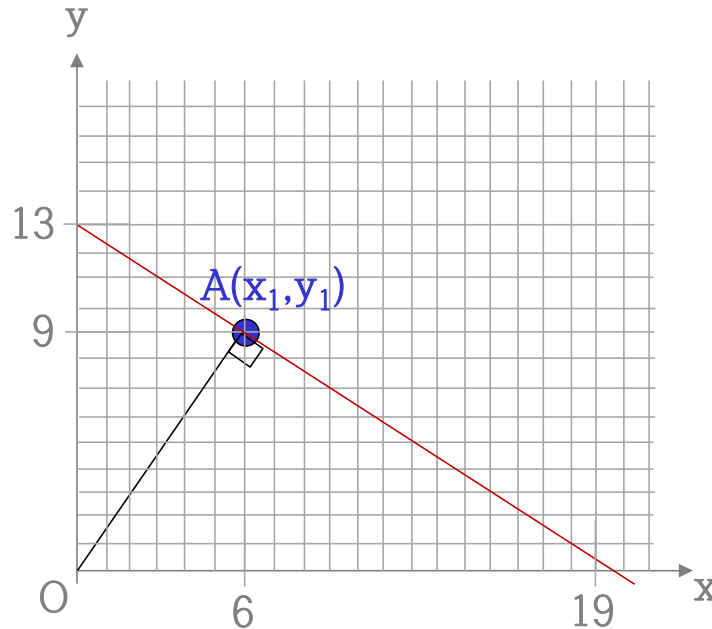
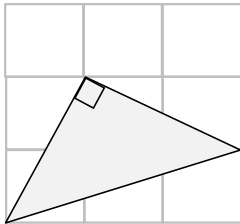
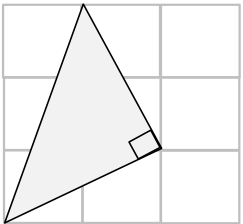
$$= \log N$$

(18) 위 방법의 수행 시간은 무엇에 비례하는가?

두 번째 방법의 추가적인 최적화 방법

32

- ① 직각이 x축 혹은 y축 상에 있는 (원점도 포함) 직각 삼각형의 개수만 count
- ② for each $A(x_1, y_1)$ in $1 \leq x_1, y_1, x_2, y_2 \leq N$, such that $A \neq 0$
OA와 수직인 직선과, 이 직선이 y축/x축과 만나는 교점 $(0, c)$ $(d, 0)$ 구하기
- ③ chose only $B(x_2, y_2)$'s, such that $\triangle OAB$ is a right triangle and then count
A의 왼쪽과 오른쪽에 있는 $B(x_2, y_2)$ 의 개수 count





두 번째 방법의 추가적인 최적화 방법

33



<정리>

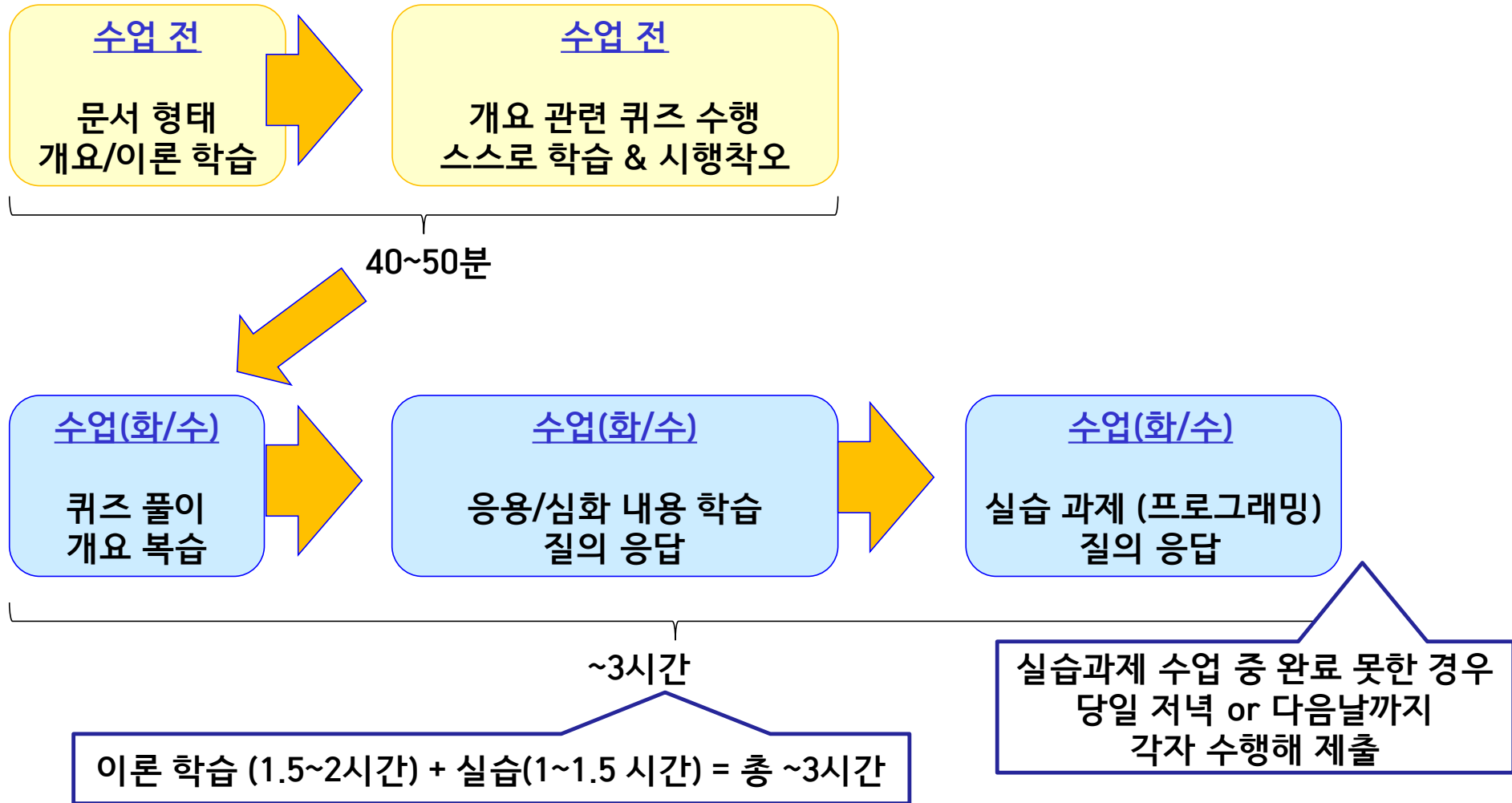
- 이번 시간 문제도 solution space를 탐색하며 해를 찾는 문제이다. 이러한 문제를 풀 때는 가능하면 불필요한 탐색을 제거하여 탐색하는 범위를 줄여나가는 것이 좋다. 이때 탐색 범위를 최소로 만드는 것은 **직접 해로 가는 방법**이다(즉 탐색하는 것마다 100% 문제에 대한 해인 경우임). 이번 주 문제에서도 점 A를 $\sim N^2$ 공간에서 선택한 후 점 B도 역시 $\sim N^2$ 공간에서 선택하며 탐색하는 방법에서 시작하였다. 하지만 최종적으로는 점 A를 선택한 후에는 이와 함께 해를 만들어 낼 수 있는 점 B를 직접 짚어내는 방법으로 최적화하였다. 이를 위해 수학, 특히 기하학을 활용하였다. 기존에도 많이 보았지만 의외로 많은 경우에 이처럼 탐색 범위를 최소화하는 것이 가능하므로 탐색 문제에서는 항상 직접 해로 가는 방법이 있는지 생각해 보자. 또한 이를 위해서 수학이 도움이 되는 경우가 많음도 기억하자.

효율적으로 알고리즘을 개선하는데
'끈질김' + '관찰'은 항상 중요



스마트 출결

35





05. 실습문제풀이 & 질의응답

- 이번 시간에 배운 내용에 대한 실습 문제 풀이 & 질의 응답
- 채점 방식은 지난 시간 문제 풀이때와 유사함
- 왜 중요한가?
- 이번 주 배운 내용을 총괄하는 문제 풀이 통해 배운 내용 활용 & 복습
- 문제 풀이 점수는 이번 주 과제 점수에 포함됨

rightTriangle2(N) 함수 구현 조건:
정수 좌표의 직각 삼각형 개수 세는 코드 작성

- 입력으로 양의 정수 N이 주어졌을 때 ($1 \leq N \leq 1000$)
- 원점을 포함하고 모든 좌표가 0 ~ N 범위의 정수인 서로 다른 직각 삼각형의 개수를 구해 반환하는 함수 구현
def rightTriangle2 (N):
- 입력 N: $1 \leq N \leq 1000$ 범위의 정수
 - 위 범위를 벗어나는 값은 입력으로 들어오지 않는다고 가정 (즉 오류 처리 하지 않아도 됨)
- 반환 값: 문제의 조건을 만족하는 삼각형의 개수
- 이번 시간에 제공한 코드 RightTriangle.py의 rightTriangle2 () 함수 내에 코드 작성해 제출

입출력 예: 문제의 조건을 만족하는 삼각형의 개수 반환

```
print(rightTriangle2(1))
```

3

```
print(rightTriangle2(2))
```

14

```
print(rightTriangle2(3))
```

33

```
print(rightTriangle2(4))
```

62

```
print(rightTriangle2(5))
```

101

그 외 예제는 __main__ 아래 테스트 코드를 참조하세요.

그 외 프로그램 구현 조건

- 최종 결과물로 RightTriangle.py 파일 하나만 제출하며, 이 파일만으로 코드가 동작해야 함
- import는 사용할 수 없음
- __main__ 아래의 코드는 작성한 함수가 올바른지 확인하는 코드로
- 코드 작성 후 스스로 채점해 보는데 활용하세요.
- 각 테스트 케이스에 대해 P(or Pass) 혹은 F(or Fail)이 출력됩니다.
- 코드를 제출할 때는 __main__ 아래 코드는 제거하거나 수정하지 말고 제출합니다. (채점에 사용되기 때문)
- 속도 테스트는 거의 항상 pass 해야만 통과입니다.



이번 시간 제공 코드 함께 보기

■ RightTriangle.py

중간고사 점수 확인

41

- 오늘 실습 마칠 때까지 공개하므로 (오후 5시까지)
- 점수 관련 문의는 이 시간 내에 해주세요.



실습 문제 풀이 & 질의 응답

- 종료 시간(17:00) 이전에 **일찍 모든 문제를 통과**한 경우 각자 **퇴실 가능**
- 실습 문제에 대한 코드는 lms 과제함에 **다음 날 23:59까지 제출** 가능합니다.
- 마감 시간까지 작성을 다 못한 경우는 (제출하지 않으면 0점이므로) 그때까지 작성한 코드를 꼭 제출해 부분점수를 받으세요.

- 실습 문제는 **개별 평가**입니다. 제출한 코드에 대한 유사도 검사를 실습 문제마다 진행하며, 코드를 건네 준 사례가 발견되면 0점 처리됩니다.
- 로직에 대해 서로 의견을 나누는 것은 괜찮지만, 코드를 직접 건네 주지는 마세요.
- 본인 실력 향상을 위해서도 **코드는 꼭 각자 직접 작성**해 주세요.