BaekJoon Code Algorithm Solution Collection

Park Cheon Bok

Q 1000. A+B

Two Input Values A and B. Return A+B

|  |
| --- |
| import sys  A, B = map(int, sys.stdin.readline().split())  print(A+B) |

★ sys.stdin.readline() = If Q needs multi line input testcases, Using “input()” will result in a verdict that will take longer than using “sys.stdin.readline()”.

Q 1001. A-B

Two Input Values A and B. Return A-B

|  |
| --- |
| import sys  A, B = map(int, sys.stdin.readline().split())  print(A-B) |

Q 1002. Turret

Two Turret staff, A and B. Each A and B calculated distance from their Turret Area to the enemy.

A’s position is (x1, x2). B’s position is (x2, y2).

The distance between A to the enemy is r1.

The distance between B to the enemy is r2.

Where can the enemy be located?

1. Testcase count = T

2. x1, y1, r1, x2, y2, r2 ( -10,000 <= x1, y1, x2, y2 <= 10,000), (10,000 >= r1, r2)

3. If the enemy can be anywhere, return -1

Testcase Sample

|  |  |
| --- | --- |
| 3  0 0 13 40 0 37  0 0 3 0 7 4  1 1 1 1 1 5 | 2  1  0 |

The solution link : <https://zifmfmphantom.tistory.com/107>

A and B described the enemy and Turret distance as r1, r2.

That means, the enemy is within the perimeter of circle with radii r1, r2.

This problem is solved by using “**Inscribed and circumscribed circle**”.

“d” is distance between A Point to B Point.

“d” equation =

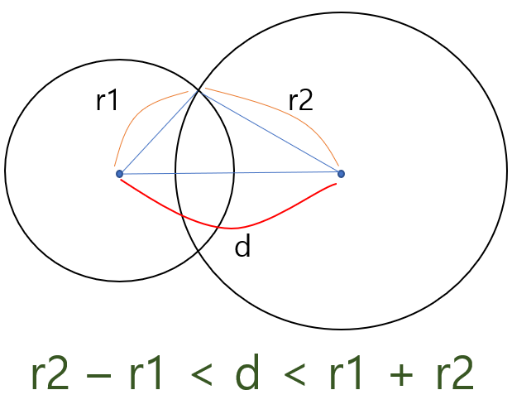
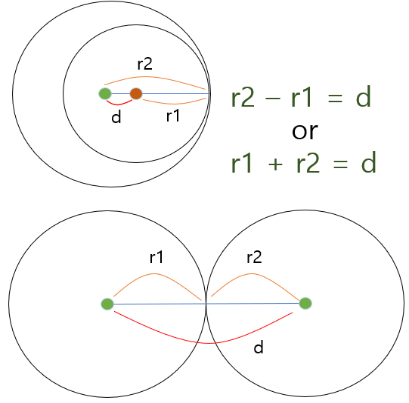
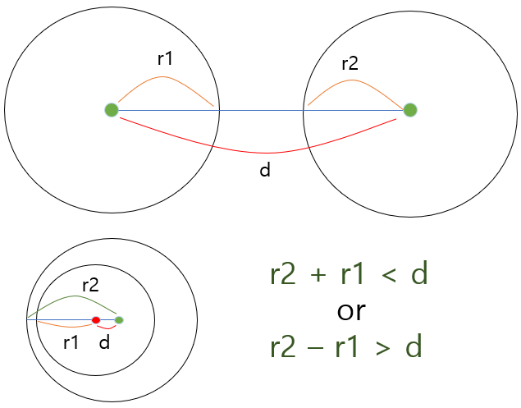
 Figure 1. Outer Product Figure 2. Inner Product Figure 3. Two Points overlap

Figure 4. “d = 0” means Two Turrets are in the same location.

That time, “r1 = r2” means Two circles are same.

|  |
| --- |
| import sys  from math import sqrt *# sqrt : 제곱근 함수*  inputF = sys.stdin.readline *# 입력 함수 텍스트 간소화*  def ***TestFunction***(TList : list):  *# d = 두 점의 거리 (Distance)*  d = sqrt((TList[**0**] - TList[**3**])\*\***2** + (TList[**1**] - TList[**4**])\*\***2**)  '''  경우 1 : 동심원 (두 점의 위치가 같고(d=0) 거리(r1,r2)도 서로 같다. => 무한대(-1)  경우 2 : 만나지 않음(0) = d가 두 반지름보다 길다 = 두 원이 떨어짐.  d가 두 반지름의 차보다 작다 = 하나의 원은 다른 원 안에 존재.  경우 3 : 1개의 점에서 만남(1) = d가 두 반지름의 거리 합과 같다. (외접)  d가 두 반지름의 거리 차와 같다. (내접)  경우 4 : 일반적인 경우(2) = 위의 특이 경우에 해당하지 않으면 두 점과 만남.  '''  if d == **0** and TList[**2**] == TList[**5**]: *# Figure 4*  return -**1**  elif d > TList[**2**]+TList[**5**] or d < abs(TList[**2**]-TList[**5**]): *# Figure 1*  return **0**  elif d == TList[**2**]+TList[**5**] or d == abs(TList[**2**]-TList[**5**]): *# Figure 2*  return **1**  else: *# Figure 3*  return **2**  if **\_\_name\_\_** == '\_\_main\_\_':  T = int(inputF()) *# TestCase Count*  for \_ in range(T):  TestList = list(map(int, inputF().split()))  print(TestFunction(TestList)) *# 구현 함수* |

Q 1003. 피보나치 함수

This is a C++ Function to describe the “Fibonacci Function”.

|  |
| --- |
| int fibonacci(int n) {  if (n == **0**) {  printf("0");  return **0**;  } else if (n == **1**) {  printf("1");  return **1**;  } else {  return fibonacci(n‐**1**) + fibonacci(n‐**2**);  }  } |

If “Fibonacci(3)” executed, Number 1 and 0 are called each twice and once.

If “Fibonacci(N)” executed. How many times will be called Number 1 and 0?

1. Testcase T = count. First Input

2. N ( N= 0 or N <= 40)

Testcase Sample

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Output | Input | Output |
| 3  0  1  3 | 1 0  0 1  1 2 | 2  6  22 | 5 8  10946 17711 |

★ First Solution Used “Recursive Function” likes Sample C++ codes.

But the result is “Time out”. Q does not want to solve using recursive.

-> Using Recursive Function

|  |
| --- |
| Cnt1 = **0**  Cnt0 = **0**  def ***FiboCnt***(N : int):  global Cnt1, Cnt0  if N == **0**:  Cnt0 += **1**  elif N == **1**:  Cnt1 += **1**  else:  FiboCnt(N-**1**)  FiboCnt(N-**2**) |

The Key of the solution is finding rules of recursive.

That is “the next value is equal to the sum of the previous two values”.

ex. N=3 -> 2+1 = (1+0) + 1

N=4 -> 3+2 = (1+0+1) + (1+0)

N=5 -> 4+3 = (1+0+1+1+0) + (1+0+1)