1 Леброн и мультивселенные

Постановка задачи: Леброн забил в игре r трехочковых и t двухочковых бросков, но в этой вселенной из результативности игрока вычитается k очков, пока его результативность не станет меньше k.

Подзадача 2

Найдем общее количество очков, которые наберет Леброн, это $sm = 3 \cdot r + 2 \cdot t$. Будем вычитать k, пока $sm \geq k$.

```
1
2 r = int(input())
3 t = int(input())
4 k = int(input())
5 sm = 3 * r + 2 * t
6 while sm >= k:
7     sm -= k
8 print(sm)
```

Подзадача 3

Для полного решения нужно понять, что после исполнения цикла мы получаем остаток от деления на k.

```
1 r = int(input())
2 t = int(input())
3 k = int(input())
4 print((3 * r + 2 * t) % k)
```

Ответ может не влезать в 32-разрядный целочисленный тип, поэтому вам следует использовать 64-разрядные целые числа, такие как `long long` в C++, чтобы избежать переполнения целочисленного типа.

2 Леброн и задачка

Постановка задачи: Дан прямоугольник со сторонами n и m, в левом нижним уголу точка $(0;0)$. Надо найти количество прямоугольных треугольников, у которых вершины в целых точках и одна из сторон параллельна оси Ox, а вторая оси Oy.

Подзадача 1

Ограничения позволяют перебрать все точки и посчитать количество подходящих треугольников.

Подзадача 2

Зафиксируем какую-нибудь точку $A(x_1; y_1)$, тогда расположение двух других точек может быть только на прямых $x = x_1$ и $y = y_1$. Переберем расположение двух оставшихся точек на этих прямых. Оценим время работы данного алгоритма: надо перебрать координаты одной точки — $O(n \cdot m)$, для двух других — только по одной координате $O(n \cdot m)$, в итоге $O(n^2 \cdot m^2)$.

```
n = int(input())
m = int(input())

ans = 0
for x1 in range(n+1):
    for y1 in range(m+1):
        for y2 in range(m+1):
            if y1 == y2:
                continue

            for x3 in range(n+1):
                if x1 == x3:
                    continue
                ans += 1
print(ans)
```

Подзадача 3

Для решения задачи на полный балл оценим точное количество итераций, которые делают данные циклы:

```
for y2 in range(m+1):
    if y1 == y2:
        continue

    for x3 in range(n+1):
        if x1 == x3:
            continue
        ans += 1
```

Цикл с переменной $x3$ делает n итераций, а цикл с $y2$ делает m итераций, получается, во вложенных циклах к переменной ans прибавляется $m \cdot n$ единиц. Заменим эти два цикла на $ans+ = n \cdot m$. Асимптотика решения теперь стала $O(n \cdot m)$.

Дополнительная подзадача

Данная задача может быть решена за асимптотику $O(1)$, для такого решения можно еще раз сделать точную оценку количества итераций и модифицировать формулу.

```python
n = int(input())
m = int(input())
print(n*m*(n+1)*(m+1))
```

Ответ может не влезать в 32-разрядный целочисленный тип, поэтому вам следует использовать 64-разрядные целые числа, такие как `long long` в C++, чтобы избежать переполнения целочисленного типа.

3 Подарок Леброну

Постановка задачи: Дан массив из n целых чисел. Нужно для каждого префикса сказать, есть ли ровно два различных числа, которые встречаются одинаковое количество раз на этом префиксе.

Подзадача 1

В первой подзадаче $a_i \leq 100$, давайте перебирать на каждом префиксе числа от 1 до 100 и считать, сколько каждого числа. Теперь осталось проверить: есть ли два числа, которые встречаются одинаковое количество раз.

```
n = int(input())
a = list(map(int, input().split()))
for i in range(n):
  b = a[:i+1]

  cnt = []
  for j in range(1, 101):
    tmpCnt = b.count(j)
    if tmpCnt > 0:
      cnt.append(tmpCnt)

  ans = False
  for j in range(n+1):
    if cnt.count(j) == 2:
      ans = True
      break

  if ans:
    print("YES")
  else:
    print("NO")
```

Подзадача 2

Во второй подзадаче $1 \leq a_i \leq 2$ воспользуемся этим для решения. Заведем две переменные *one* и *two*, в которых будем считать количество единиц и двоек на префиксе соответсвенно. Во время обработки нового элемента будем добавлять единицу в одну из переменных, после чего надо проверить: одинаковое ли их количество.

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

one = 0
two = 0

for i in range(n):
  if a[i] == 1:
    one += 1
  else:
    two += 1

  if one == two:
    print("YES")
  else:
    print("NO")
```

Другое решение подзадачи 1

Прежде чем изложить решение 3 подзадачи, рассмотрим иное решение первой подзадачи. Ограничения на элемент массива $1 \leq a_i \leq 100$, воспользуемся этим для решения, заведем массив cnt, в котором будем поддерживать, сколько раз встречается k, где $k = 1, 2, \ldots, 100$. Когда будем обрабатывать новый элемент a_i, к ячейке массива a_i будем добавлять единицу, то есть $cnt[a_i] += 1$. Теперь осталось найти ровно два элемента, которые будут иметь одинаковое значение, для этого воспользуемся решениями, рассказанными выше.

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


mx = max(a)
cnt = [0] * (mx + 1)


for i in range(n):
  cnt[a[i]] += 1

  ans = False
  for j in range(1, n+1):
    if cnt.count(j) == 2:
      ans = True
      break

  if ans:
    print("YES")
  else:
    print("NO")
```

Подзадача 3

Заметим, что можно повторно использовать технику с массивом, но для подсчета количества каждого элемента массива cnt. Теперь для проверки того, что на префиксе ровно два элемента, которые встречаются одинаковое количество раз, надо пройтись по массиву $cnt2$.

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

mx = max(a)
cnt = [0] * (mx + 1)

cnt2 = [0] * (n+1)
for i in range(n):
  if cnt2[cnt[a[i]]] != 0:
    cnt2[cnt[a[i]]] -= 1

  cnt[a[i]] += 1
  cnt2[cnt[a[i]]] += 1

  ans = False
  for j in range(1, n+1):
    if cnt2[j] == 2:
      ans = True
      break
  if ans:
    print("YES")
  else:
    print("NO")
```

Данная реализация идеи набирала 50 баллов, но немного доработав это решение, можно было получить 65 баллов.

Подзадача 4

Для решения задачи на полный балл надо заметить, что при обработке нового элемента a_i количество пар элементов, которые подходят, меняется не более чем на 2. Заведем переменную ans, отвечающую за количество ровно двух различных чисел, которые встречаются одинаковое количество раз на этом префиксе. При обработке нового числа возможно несколько вариантов изменения переменной ans (считаем что уже прибавили единицу к $cnt[a[i]]$ и произошли изменения в $cnt2$):

1. если $cnt[a[i]] > 1$ и $cnt2[cnt[a[i]] - 1] == 1$, то $ans- = 1$
2. если $cnt2[cnt[a[i]] - 1] == 2$, то $ans+ = 1$
3. если $cnt2[cnt[a[i]]] == 3$, то $ans- = 1$
4. если $cnt2[cnt[a[i]]] == 2$, то $ans+ = 1$

После изменения ans осталось только проверить: больше ли 0 эта переменная, если да, то ответ **YES**, если нет, то **NO**.

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

mx = max(a)
ans = 0
cnt = [0] * (mx + 1)
cnt2 = [0] * (n+1)

for i in range(n):
  if cnt2[cnt[a[i]]] != 0:
    cnt2[cnt[a[i]]] -= 1

  if cnt2[cnt[a[i]]] == 1 and cnt[a[i]] > 0:
    ans -= 1

  if cnt2[cnt[a[i]]] == 2:
    ans += 1

  cnt[a[i]] += 1
  cnt2[cnt[a[i]]] += 1

  if cnt2[cnt[a[i]]] == 3:
    ans -= 1

  if cnt2[cnt[a[i]]] == 2:
    ans += 1

  if ans > 0:
    print("YES")
  else:
    print("NO")
```

4 Леброн и домино

Постановка задачи: Дан массив пар чисел (доминошек). В паре числа можно менять местами, надо найти максимальную по длине последовательность подряд идущих пар, у которых любые две соприкасались сторонами с одинаковым числом.

Подзадача 1

Для решения первой подзадачи можно было написать перебор.

Подзадача 3

Заметим тот факт, что если зафиксируем определенное положение пары, то только один вариант расположение следующей пары. Теперь переберем начало последовательности и рассмотрим два варианта положения этой доминошки. После чего будем моделировать построение.

```
n = int(input())
a = []
for i in range(n):
  a.append(list(map(int, input().split())))

ans = 0
for i in range(n):
  tmp = [1, 1]
  for t in range(2):
    last = a[i][t]
    for j in range(i+1, n):
      if last == a[j][0]:
        last = a[j][1]
      elif last == a[j][1]:
        last = a[j][0]
      else:
        break
      tmp[t] += 1
  ans = max(ans, max(tmp))
print(ans)
```

Подзадача 4

Для полного решения задачи надо воспользоваться методом динамического программирования. Пусть $dp[i][0]$ — длина подряд идущих доминошек, заканчивающихся i-ой без ее переворота, а $dp[i][1]$ — с поворотом. База у динамики $dp[i][0] = dp[i][1] = 1$, так как минимальная длина равна 1 (одна доминошка это уже последовательность). Варианты переходов:

1. $a[i-1][1] == a[i][0]$, то $dp[i][0] = max(dp[i][0], dp[i-1][0] + 1)$
2. $a[i-1][0] == a[i][0]$, то $dp[i][0] = max(dp[i][0], dp[i-1][1] + 1)$
3. $a[i-1][1] == a[i][1]$, то $dp[i][1] = max(dp[i][1], dp[i-1][0] + 1)$
4. $a[i-1][0] == a[i][1]$, то $dp[i][1] = max(dp[i][1], dp[i-1][1] + 1)$

Ответ это максимум среди всех $dp[i][0]$ и $dp[i][1]$.

```
n = int(input())
a = []
for i in range(n):
  a.append(list(map(int, input().split())))

```

```
ans = 1
dp = [[1, 1] for i in range(n)]

for i in range(1, n):
  if a[i - 1][1] == a[i][0]:
    dp[i][0] = max(dp[i][0], dp[i - 1][0] + 1)

  if a[i - 1][0] == a[i][0]:
    dp[i][0] = max(dp[i][0], dp[i - 1][1] + 1)

  if a[i - 1][1] == a[i][1]:
    dp[i][1] = max(dp[i][1], dp[i - 1][0] + 1)

  if a[i - 1][0] == a[i][1]:
    dp[i][1] = max(dp[i][1], dp[i - 1][1] + 1)
  ans = max(ans, dp[i][0], dp[i][1])
print(ans)
```

5 Окружности и баскетбол

Постановка задачи: Дано n неравенств вида $(x-x_i)^2+(y-y_i)^2 \le r_i^2$. Надо посчитать количество целых точек (x, y), которые удовлетворяют хотя бы одному из неравенств.

Подзадача 1

Ограничения позволяют перебрать целые точки, которые могут удовлетворять хотя бы одному из неравенств. Для того чтобы не учитывать точки несколько раз, можно использовать двумерный булевый массив или set. Асимптотика $O(200^2 \cdot n)$

```
n = int(input())
inequalities = []
for i in range(n):
  inequalities.append(list(map(int, input().split())))

used = set()
for x_i, y_i, r_i in inequalities:
  for x in range(x_i - r_i, x_i + r_i+1):
    for y in range(y_i - r_i, y_i + r_i+1):
      if (x_i - x) * (x_i - x) + (y_i - y) * (y_i - y) <= r_i*r_i:
        used.add((x, y))
print(len(used))
```

Подзадача 2

Проанализировав ограничения, понятно, что координаты точек могут быть от $-2 \cdot 10^4$ до $2 \cdot 10^4$. Давайте перебирать прямую $y = j$ и подставлять y в неравенство (то есть пересекаем прямую $y = j$ с окружностью $(x - x_i)^2 + (y - y_i)^2 = r_i^2$).

$$
\begin{gathered}
(x - x_i)^2 + (y - y_i)^2 = r_i^2 \\
(x - x_i)^2 = r_i^2 - (y - y_i)^2 \\
x - x_i = \pm\sqrt{r_i^2 - (y - y_i)^2} \\
x = x_i \pm \sqrt{r_i^2 - (y - y_i)^2}
\end{gathered}
$$

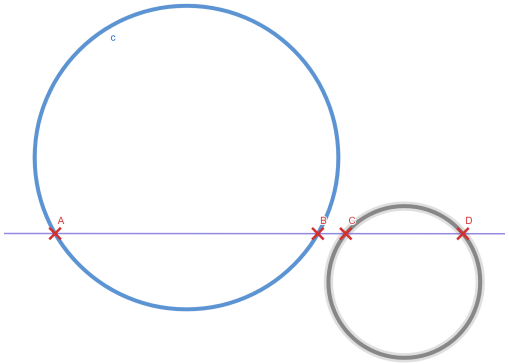
Запись $\lfloor x \rfloor$ обозначает округление вниз значения x (например, $\lfloor 1.4 \rfloor = 1$, $\lfloor 2 \rfloor = 2$). Так как подходят только целые точки, то нужные точки будут иметь координаты $(\lfloor x_i - \sqrt{r_i^2 - (y - y_i)^2} \rfloor, y)$, $(\lfloor x_i - \sqrt{r_i^2 - (y - y_i)^2} \rfloor + 1, y)$, …, $(\lfloor x_i + \sqrt{r_i^2 - (y - y_i)^2} \rfloor, y)$. Удовлетворяющих точек с координатой $y = j$ всего $\lfloor x_i + \sqrt{r_i^2 - (y - y_i)^2} \rfloor - \lfloor x_i - \sqrt{r_i^2 - (y - y_i)^2} \rfloor + 1 = 2 \cdot \lfloor \sqrt{r_i^2 - (y - y_i)^2} \rfloor + 1$.

```
from math import sqrt

n = int(input())
inequalities = []
for i in range(n):
  inequalities.append(list(map(int, input().split())))

cnt = 0
for y in range(-20000, 20001):
  for x_i, y_i, r_i in inequalities:
    if abs(y - y_i) > r_i:
      continue
    d = int(sqrt(r_i * r_i - (y - y_i) * (y - y_i)))
    cnt += 2*d + 1
print(cnt)
```

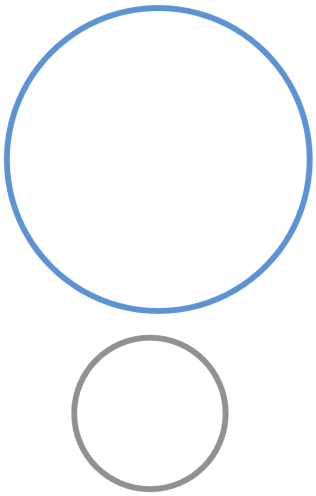
Подзадача 3

В подзадаче $n = 2$, воспользуемся уже изложенным методом в подзадаче 2. Будем пересекать прямую с окружностями, но в этот раз рассмотрим расположение точек пересечения с окружностью. Всего 4 варианта расположения:

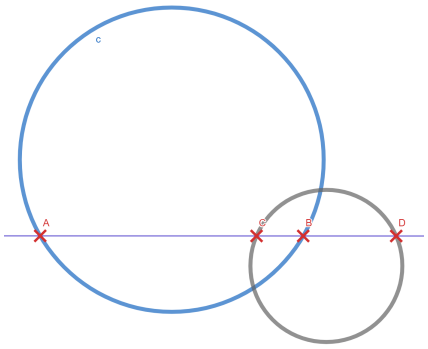
1. В данном случае надо проверить, что точка B левее C, то есть $x_B \leq x_c$, количество подходящих точек $(x_B - x_A + 1) + (x_D - x_C + 1)$
2. В этом случае прямая никогда не будет пересекать одновременно две окружности
3. Здесь нужно выполнить проверку, что точки располагаются в данном порядке: $A \to C \to B \to D$, то есть $x_A \leq x_C \leq x_B \leq x_D$, количество подходящих точек $x_D - x_A + 1$
4. В четвертом случае одна окружность внутри другой, то есть $x_A \leq x_C \leq x_D \leq x_B$, количество подходящих точек $x_B - x_A + 1$



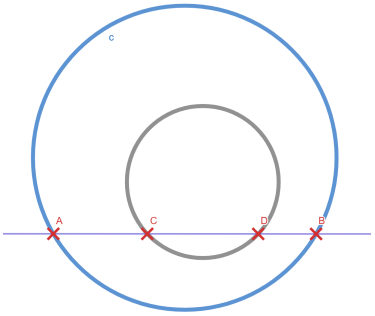
1.



2.



3.



4.

```python
from math import sqrt

n = int(input())
inequalities = []
for i in range(n):
  inequalities.append(list(map(int, input().split())))

cnt = 0
for y in range(-20000, 20001):
```

```
  points = []
  for x_i, y_i, r_i in inequalities:
    if abs(y - y_i) > r_i:
      continue
    d = int(sqrt(r_i * r_i - (y - y_i) * (y - y_i)))
    points.append((x_i - d, x_i + d))

  if len(points) == 0:
    continue

  points.sort()
  tmp = 0
  if len(points) == 1:
    A, B = points[0]
    tmp += B - A + 1
  elif len(points) == 2:
    A, B = points[0]
    C, D = points[1]

    if B < C:
      tmp += B - A + 1 + D - C + 1
    elif A <= C <= B <= D:
      tmp += D - A + 1
    else:
      tmp += B - A + 1
  cnt += tmp

print(cnt)
```

Подзадача 4

Для решения четвертой подзадачи надо обрабатывать пересечение нескольких отрезков для этого воспользуемся методом сканирующей прямой (ScanLine). Получается, надо посчитать на прямой количество точек, покрытых отрезками.

```
from math import sqrt

n = int(input())

x = [0]*n
y = [0]*n
r = [0]*n
for i in range(n):
  x[i], y[i], r[i] = map(int, input().split())


plY = [[] for i in range(-20000, 20001)]

for i in range(n):
  for lineY in range(y[i] - r[i], y[i]+r[i]+1):
    if(abs(lineY - y[i]) > r[i]):
      continue
    d = int(sqrt(r[i]*r[i] - (lineY - y[i])**2))
    x1 = x[i] - d
    x2 = x[i] + d

    plY[lineY].append((x1, -1))
    plY[lineY].append((x2, 1))

ans = 0
for i in range(-20000, 20001):
  if len(plY[i]) == 0:
    continue
```

```
29
30    plY[i].sort()
31    cnt = 0
32    lastX = 0
33    for curX, curType in plY[i]:
34      cnt -= curType
35
36      if cnt == 0 and curType == 1:
37        ans += curX - lastX + 1
38      if cnt == 1 and curType == -1:
39        lastX = curX
40
41  print(ans)
```