

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 задача

Постановка задачи: Дан прямоугольник со сторонами n и m , в левом нижнем углу 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 $O(n^2 \cdot m^2)$.

```
1 n = int(input())
2 m = int(input())
3
4 ans = 0
5 for x1 in range(n+1):
6     for y1 in range(m+1):
7         for y2 in range(m+1):
8             if y1 == y2:
9                 continue
10
11             for x3 in range(n+1):
12                 if x1 == x3:
13                     continue
14                 ans += 1
15 print(ans)
```

Подзадача 3

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

```
1 for y2 in range(m+1):
2     if y1 == y2:
3         continue
4
5     for x3 in range(n+1):
6         if x1 == x3:
7             continue
8     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. Заменяем 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.

```
1 n = int(input())
2 m = int(input())
3 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 Леброн

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.

```
1 n = int(input())
2 a = list(map(int, input().split()))
3 for i in range(n):
4     b = a[:i+1]
5
6     cnt = []
7     for j in range(1, 101):
8         tmpCnt = b.count(j)
9         if tmpCnt > 0:
10            cnt.append(tmpCnt)
11
12     ans = False
13     for j in range(n+1):
14         if cnt.count(j) == 2:
15             ans = True
16             break
17
18     if ans:
19         print("YES")
20     else:
21         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 соответственно. Во время обработки нового элемента будем добавлять единицу в одну из переменных, после чего надо проверить: одинаковое ли их количество.

```
1 n = int(input())
2 a = list(map(int, input().split()))
3
4 one = 0
5 two = 0
6
7 for i in range(n):
8     if a[i] == 1:
9         one += 1
10    else:
11        two += 1
12
13    if one == two:
14        print("YES")
15    else:
16        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 $k = 1, 2, \dots, 100$. 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.

```
1 n = int(input())
2 a = list(map(int, input().split()))
3
4
5 mx = max(a)
6 cnt = [0] * (mx + 1)
7
8
9 for i in range(n):
10     cnt[a[i]] += 1
11
12 ans = False
13 for j in range(1, n+1):
14     if cnt.count(j) == 2:
15         ans = True
16         break
17
18 if ans:
19     print("YES")
20 else:
21     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 пройти по массиву *cnt2*.

```
1 n = int(input())
2 a = list(map(int, input().split()))
3
4 mx = max(a)
5 cnt = [0] * (mx + 1)
6
7 cnt2 = [0] * (n+1)
8 for i in range(n):
9     if cnt2[cnt[a[i]]] != 0:
10         cnt2[cnt[a[i]]] -= 1
11
12     cnt[a[i]] += 1
13     cnt2[cnt[a[i]]] += 1
14
15 ans = False
16 for j in range(1, n+1):
17     if cnt2[j] == 2:
18         ans = True
19         break
20 if ans:
21     print("YES")
22 else:
23     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**.

```
1 n = int(input())
2 a = list(map(int, input().split()))
3
4 mx = max(a)
5 ans = 0
6 cnt = [0] * (mx + 1)
7 cnt2 = [0] * (n+1)
8
9 for i in range(n):
10     if cnt2[cnt[a[i]]] != 0:
11         cnt2[cnt[a[i]]] -= 1
12
13     if cnt2[cnt[a[i]]] == 1 and cnt[a[i]] > 0:
14         ans -= 1
15
16     if cnt2[cnt[a[i]]] == 2:
17         ans += 1
18
19     cnt[a[i]] += 1
20     cnt2[cnt[a[i]]] += 1
21
22     if cnt2[cnt[a[i]]] == 3:
23         ans -= 1
24
25     if cnt2[cnt[a[i]]] == 2:
26         ans += 1
27
28     if ans > 0:
29         print("YES")
30     else:
31         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 расположения следующей пары. Теперь переберем начало последовательности и рассмотрим два варианта положения этой доминошки. После чего будем моделировать построение.

```
1 n = int(input())
2 a = []
3 for i in range(n):
4     a.append(list(map(int, input().split())))
5
6 ans = 0
7 for i in range(n):
8     tmp = [1, 1]
9     for t in range(2):
10        last = a[i][t]
11        for j in range(i+1, n):
12            if last == a[j][0]:
13                last = a[j][1]
14            elif last == a[j][1]:
15                last = a[j][0]
16            else:
17                break
18        tmp[t] += 1
19    ans = max(ans, max(tmp))
20 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 $dp[i][1]$.

```
1 n = int(input())
2 a = []
3 for i in range(n):
4     a.append(list(map(int, input().split())))
5
```

```
6 ans = 1
7 dp = [[1, 1] for i in range(n)]
8
9 for i in range(1, n):
10     if a[i - 1][1] == a[i][0]:
11         dp[i][0] = max(dp[i][0], dp[i - 1][0] + 1)
12
13     if a[i - 1][0] == a[i][0]:
14         dp[i][0] = max(dp[i][0], dp[i - 1][1] + 1)
15
16     if a[i - 1][1] == a[i][1]:
17         dp[i][1] = max(dp[i][1], dp[i - 1][0] + 1)
18
19     if a[i - 1][0] == a[i][1]:
20         dp[i][1] = max(dp[i][1], dp[i - 1][1] + 1)
21     ans = max(ans, dp[i][0], dp[i][1])
22 print(ans)
```

5 Окружности и баскетбол

Постановка задачи: Дано n неравенств вида $(x-x_i)^2+(y-y_i)^2 \leq r_i^2$. 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 $O(200^2 \cdot n)$

```
1 n = int(input())
2 inequalities = []
3 for i in range(n):
4     inequalities.append(list(map(int, input().split())))
5
6 used = set()
7 for x_i, y_i, r_i in inequalities:
8     for x in range(x_i - r_i, x_i + r_i + 1):
9         for y in range(y_i - r_i, y_i + r_i + 1):
10            if (x_i - x) * (x_i - x) + (y_i - y) * (y_i - y) <= r_i * r_i:
11                used.add((x, y))
12 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 $(x - x_i)^2 + (y - y_i)^2 = r_i^2$).

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

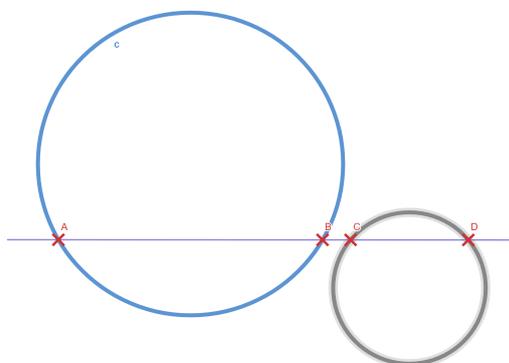
Запись $\lfloor x \rfloor$ обозначает округление вниз значения x (например, $\lfloor 1.4 \rfloor = 1$, $\lfloor 2 \rfloor = 2$). Так как подходят только целые точки, то нужные точки будут иметь координаты $(\lfloor x_i - \sqrt{r_i^2 - (y - y_i)^2} \rfloor, y)$, $(\lfloor x_i - \sqrt{r_i^2 - (y - y_i)^2}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

```
1 from math import sqrt
2
3 n = int(input())
4 inequalities = []
5 for i in range(n):
6     inequalities.append(list(map(int, input().split())))
7
8 cnt = 0
9 for y in range(-20000, 20001):
10    for x_i, y_i, r_i in inequalities:
11        if abs(y - y_i) > r_i:
12            continue
13        d = int(sqrt(r_i * r_i - (y - y_i) * (y - y_i)))
14        cnt += 2*d + 1
15 print(cnt)
```

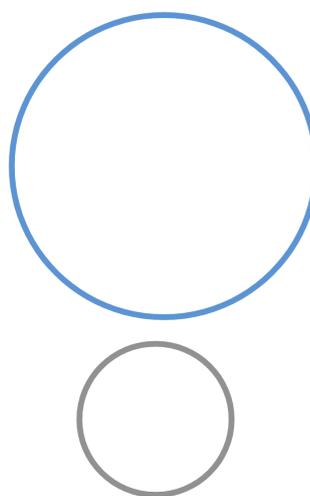
Подзадача 3

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:

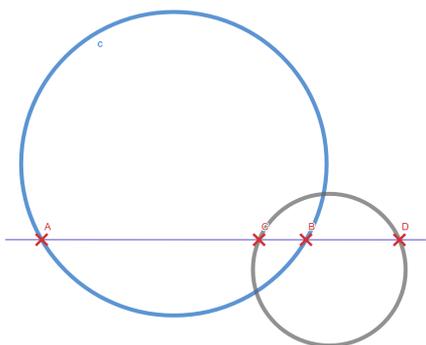
1. В данном случае надо проверить, что точка B левее C , то есть $x_B \leq x_C$, количество подходящих точек $(x_B - x_A + 1) + (x_D - x_C + 1)$
2. В этом случае прямая никогда не будет пересекать одновременно две окружности
3. Здесь нужно выполнить проверку, что точки располагаются в данном порядке: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



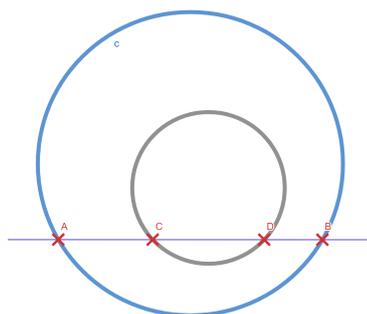
1.



2.



3.



4.

```
1 from math import sqrt
2
3 n = int(input())
4 inequalities = []
5 for i in range(n):
6     inequalities.append(list(map(int, input().split())))
7
8 cnt = 0
9 for y in range(-20000, 20001):
```

```

10 points = []
11 for x_i, y_i, r_i in inequalities:
12     if abs(y - y_i) > r_i:
13         continue
14     d = int(sqrt(r_i * r_i - (y - y_i) * (y - y_i)))
15     points.append((x_i - d, x_i + d))
16
17 if len(points) == 0:
18     continue
19
20 points.sort()
21 tmp = 0
22 if len(points) == 1:
23     A, B = points[0]
24     tmp += B - A + 1
25 elif len(points) == 2:
26     A, B = points[0]
27     C, D = points[1]
28
29     if B < C:
30         tmp += B - A + 1 + D - C + 1
31     elif A <= C <= B <= D:
32         tmp += D - A + 1
33     else:
34         tmp += B - A + 1
35 cnt += tmp
36
37 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.

```

1 from math import sqrt
2
3 n = int(input())
4
5 x = [0]*n
6 y = [0]*n
7 r = [0]*n
8 for i in range(n):
9     x[i], y[i], r[i] = map(int, input().split())
10
11
12 plY = [[] for i in range(-20000, 20001)]
13
14 for i in range(n):
15     for lineY in range(y[i] - r[i], y[i]+r[i]+1):
16         if(abs(lineY - y[i]) > r[i]):
17             continue
18         d = int(sqrt(r[i]*r[i] - (lineY - y[i])**2))
19         x1 = x[i] - d
20         x2 = x[i] + d
21
22         plY[lineY].append((x1, -1))
23         plY[lineY].append((x2, 1))
24
25 ans = 0
26 for i in range(-20000, 20001):
27     if len(plY[i]) == 0:
28         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)
```