

Разбор задач муниципального этапа олимпиады по информатике

1. Пирожки (все классы)

Тема: вывод формулы

Сложность: простая

Во всех подзадачах необходимо найти минимальное такое C , что $C \cdot N \leq X$ (X рублей достаточно на покупку N пирожков), но $C \cdot (N+1) > X$ (X рублей недостаточно на покупку $N+1$ пирожка).

Подзадача 1

Можно заметить, что $C \leq X$. В противном случае $C \cdot N$ при любом натуральном N было бы больше X , т.е. не удалось бы купить ни одного пирожка.

Значит, искомая величина лежит в пределах отрезка $[1; X]$. Переберем все натуральные числа до X в порядке возрастания. Первое же подходящее является ответом. Если ни одно из чисел не подходит, выведем -1 . Временная сложность такого решения - $O(X)$.

Если завершать перебор сразу же после нахождения первого подходящего решения, то получим вариант с временной сложностью $O(C)$. Поскольку C - натуральное, и $C \cdot N \leq X$, то C не превосходит целой части X/N , и $O(C) = O(X/N)$.

Подзадача 2

С одной стороны, C не превосходит целой части X/N . С другой стороны, поскольку $C \cdot (N+1) > X$, то $C > X/(N+1)$. То есть, C принадлежит полуинтервалу $(X/(N+1); X/N]$.

Его длина $X/N - X/(N+1) = X/(N \cdot (N+1))$. Значит, если мы переберем целые значения в этом полуинтервале и выберем минимальное подходящее, временная сложность такого варианта будет $O(X/(N \cdot (N+1)))$. При $X \leq 10^{12}$ и $N \geq 10^3$ эта величина не превосходит 10^6 .

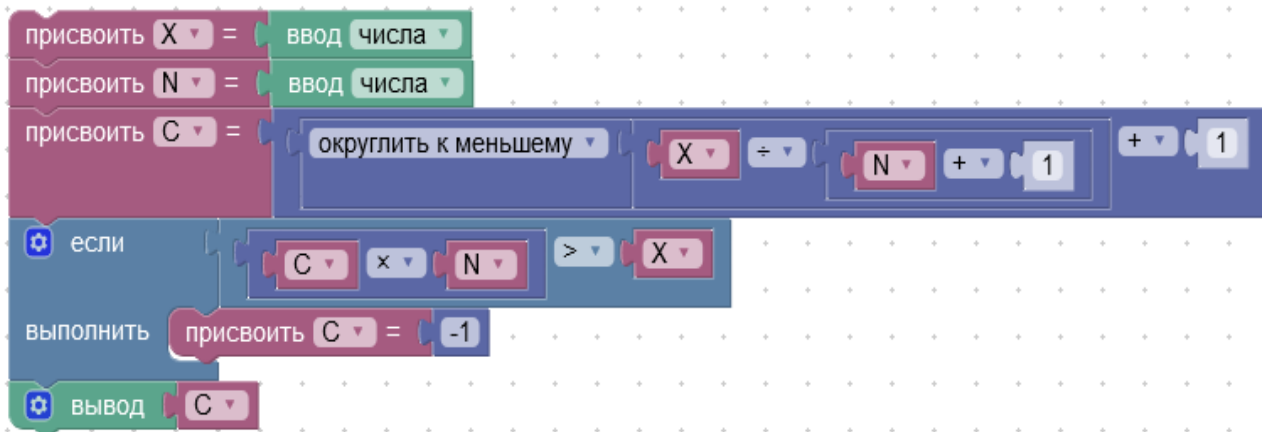
Подзадача 3

От нас требуется найти минимальное целое C , для которого $C \cdot (N+1) > X$, то есть $C > X/(N+1)$. Заметим, что таким числом является $C = \lfloor X/(N+1) \rfloor + 1$, где $\lfloor \cdot \rfloor$ обозначает взятие целой части числа. Для доказательства этого утверждения достаточно рассмотреть случаи X , кратного и не кратного $N+1$. Если найденное таким образом C удовлетворяет и второму условию задачи ($C \cdot N \leq X$), то оно является ответом. В противном случае, ни одного подходящего числа не существует.

Временная сложность решения — $O(1)$

Пример реализации:

```
x = int(input())
n = int(input())
c = x // (n+1) + 1
if c*n > x:
    c = -1
print(c)
```



2. Алгоритм (7-8 класс)

Тема: реализация программы по схеме алгоритма

Сложность: простая

Пример реализации:

```
n=int(input())
k=0
while not((n==1) or (n==4)): # not так как на схеме выход по "да"
    k+=1
    m=0
    while n>0:
        m+=(n%10)**2
        n=n//10
    n=m
print(k,n)
```

3. Рисование снежинки (7-8 классы)

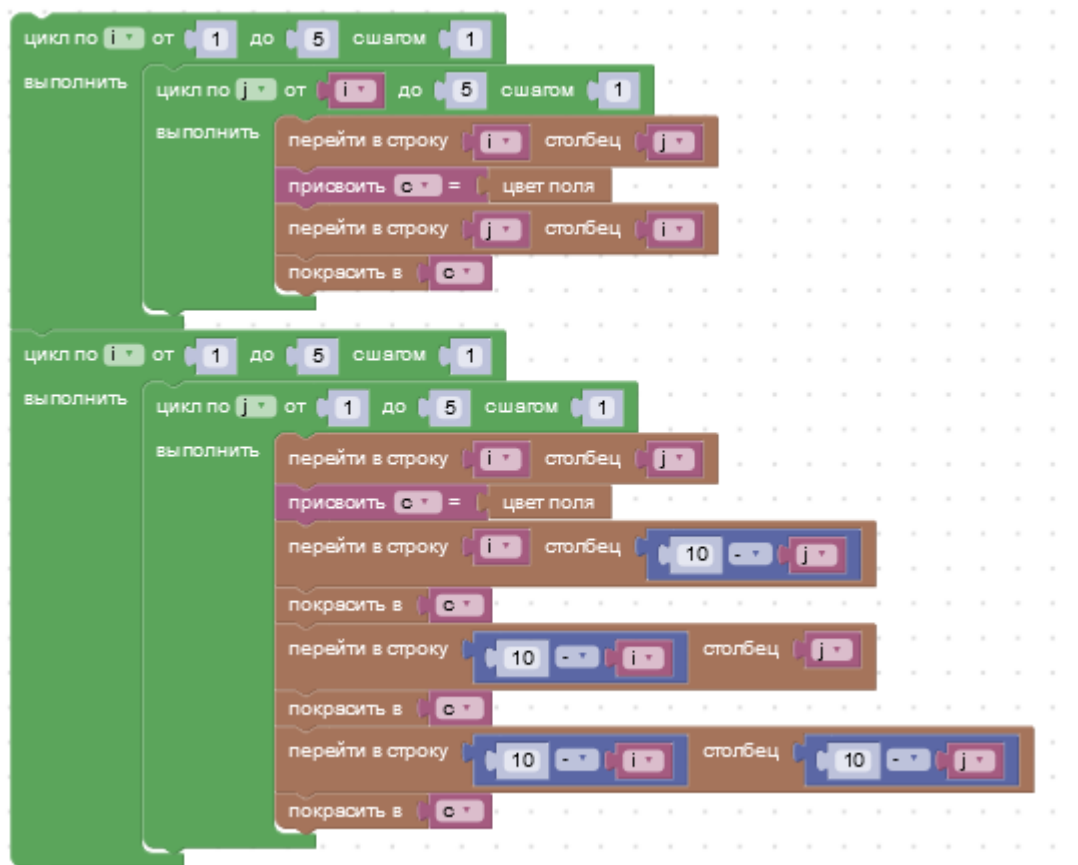
Тема: исполнители, циклы

Сложность: простая

В подзадаче 2 достаточно считать цвет начальной клетки и двойным циклом закрасить все поле.

Для упрощения решения лучше выделить каждое отражение в отдельный цикл, сначала диагональное, затем вертикальное и затем горизонтальное. Можно сэкономить два цикла, выполняя горизонтальное и вертикальное отражение вместе.

Пример реализации:



2 или 4. Склад (все классы)

Тема: сортировка, жадность

Сложность: ниже средней

Каждая цепь, отделенная по способу Марины, дает звено, а каждая цепь, собранная из одиночных звеньев, уменьшает количество отделяемых цепей, поэтому выгоднее всего использовать одиночные звенья для формирования маленьких цепей. В подзадаче 1 можно искать самую длинную цепь, затем искать самую короткую цепь и уменьшать ее длину на 1.

Для решения подзадачи 2 необходимо применить сортировку. Тогда все короткие цепи будут в начале списка, а длинные — в конце и вместо поиска можно просто передвигать индекс. Если по завершению цикла остается одна цепь, то ее придется отделять старым способом (при этом добавляя к ней оставшиеся одиночные звенья)

Пример реализации:

```
n=int(input())
a=list(map(int,input().split()))
a.sort()
i=n-1
j=0
ans=0
while j<i:
    a[j]-=1
    if a[j]==0:
        j+=1
    ans+=1
    i-=1
if i==j:
    ans+=1
print(ans)
```

5. Справедливые подмассивы-1 (7-8 класс)

Тема: частичные суммы, ассоциативные массивы

Сложность: средняя

Подзадача 1

Переберем все подмассивы входного массива. Для каждого подмассива пройдем по его элементам и вычислим отдельно суммы четных и нечетных его элементов. Если суммы оказались равны, значит подмассив - справедливый, и ответ нужно увеличить на 1. Временная сложность решения - $O(N^3)$.

Подзадача 2

Массив является справедливым тогда и только тогда, когда знакопередающая сумма его элементов равна нулю, т.е. $a[L] - a[L+1] + a[L+2] - a[L+3] + \dots = 0$. Чтобы находить значение подобной знакопередающей суммы для любого подмассива за $O(1)$, подсчитаем префиксные знакопередающие суммы для входного массива:

```
prefix[0] = 0
prefix[1] = prefix[0] + a[0] = a[0]
prefix[2] = prefix[1] - a[1] = a[0] - a[1]
prefix[3] = prefix[2] + a[2] = a[0] - a[1] + a[2]
```

...

Тогда значение суммы для подмассива от L до R равно разности $prefix[R+1] - prefix[L]$. Значит, ответ равен количеству пар индексов (L; R) таких, что $prefix[L] = prefix[R+1]$. Переберем все пары индексов и подсчитаем среди них количество подходящих. Временная сложность решения — $O(N^2)$.

```
n = int(input())
a = list(map(int, input().split()))
pref = [0]
for i in range(n):
    pref.append(pref[-1] + (a[i] if i % 2 else -a[i]))
cnt = 0
for left in range(n):
    for right in range(left, n):
        if pref[right+1] == pref[left]:
            cnt += 1
print(cnt)
```

Подзадача 3

Аналогично предыдущей подзадаче, подсчитаем массив префиксных знакопередающих сумм для

исходного массива и попробуем найти количество пар индексов с совпадающими значениями этих сумм быстрее, чем за $O(N^2)$. Суммы принимают значения из отрезка $[-N \cdot \max(a_i); N \cdot \max(a_i)]$. Подсчитаем, сколько раз префиксные суммы принимают каждое из значений X в этом отрезке и сохраним результат в элементе массива $\text{cnt}[X + N \cdot \max(a_i)]$ (добавление $N \cdot \max(a_i)$ требуется для того, чтобы сделать индекс неотрицательным). Тогда для каждого X возможно $\text{cnt}[X] \cdot (\text{cnt}[X] - 1) / 2$ вариантов отрезка с концами, в которых префиксная сумма имеет соответствующее значение. Переберем все возможные значения суммы и для каждой из них вычислим ответ по формуле. Временная сложность решения - $O(N \cdot \max(a_i))$.

Подзадача 4

Поступим аналогично предыдущей подзадаче, но вместо массива счетчиков размером $2 \cdot N \cdot \max(a_i) + 1$ заведем ассоциативный массив (словарь). Ключом в словаре будут являться значения знакопередающей префиксной суммы, значением - сколько раз такая сумма встречается в массиве. В словаре будет $O(N)$ элементов, которые можно перебрать и вычислить для каждого количество соответствующих подмассивов по формуле. Временная сложность решения - $O(N)$ либо $O(N \log N)$ в зависимости от используемой реализации словаря.

Пример реализации:

```
n = int(input())
a = list(map(int, input().split()))
cnt = dict()
pref = 0
cnt[0] = 1
for i in range(n):
    pref += a[i] if i % 2 else -a[i]
    if pref in cnt:
        cnt[pref] += 1
    else:
        cnt[pref] = 1
result = 0
for c in cnt.values():
    result += c * (c - 1) // 2
print(result)
```