Разбор задач

Задача 1. Самолёт

Если $n \leqslant 6$, то ответом будет число 1. Каждая следующая полная или неполная четвёрка кресел добавляет один проход, то есть для $n = 7..10$ ответом будет 2, а для $n = 11...14$ ответом будет 3.

Поэтому для нахождения ответа нужно взять значение $n - 6$ и поделить на 4 с округлением вверх, что можно сделать по формуле $(n - 6 + 4 - 1)//4$, то есть $(n - 3)//4$.

К результату нужно прибавить 1 (один проход для первых 6 рядов), и возьмём максимум из этого числа и 1 (т.к. при маленьких n должен быть хотя бы один проход).

Пример решения на языке Python.

```
n = int(input())
print(max(1, 1 + (n - 3) // 4))
```

Задача 2. Майки и носки

Эта задача оказалась довольно сложной для многих участников, т.к. нужно было разобрать все принципиально возможные случаи. Многие участники набрали по этой задаче от 52 до 58 баллов, рассмотрев только основной случай. Несколько участников при этом решили все остальные задачи на 100 баллов, но так и не нашли полное решение этой задачи.

Первоначально предметно-методическая комиссия считала эту задачу несложной и планировала дать её на школьный этап, однако, в ходе подготовки задачи оказалось, что она куда более интересна, чем предполагалось заранее, поэтому эта задача была отложена на муниципальный этап.

У Саши есть несколько стратегий действий.

1. Выбрать $B+1$ майку и $D+1$ пару носков. Тогда поскольку количество красных маек равно B, то он обязательно вытащит синюю майку. Также Саша обязательно вытащит синюю пару носков, взяв $D + 1$ пару носков. Итого, вытащив $B + 1$ майку и $D + 1$ пару носков, он обязательно получит синий комплект вещей.

2. Аналогично, можно получить комплект из красной майки и красных носков, взяв $A+1$ майку и $C + 1$ пару носков.

Решения, которые рассматривали только два этих случая, набирали 56 баллов.

Самый простой пример, на котором такое решение работает неправильно, это случай $A = B = C = D = 1$. Такое решение выдаст ответ $(2, 2)$, хотя можно вытащить всего три предмета: например, $(2, 1)$ или $(1, 2)$. Это следующие стратегии.

3. Взять $\max(A, B) + 1$ майку. Тогда Саша гарантированно вытащит и синюю, и красную майки и тогда ему достаточно вытащить 1 пару носков, которая может быть любого цвета.

4. Взять 1 майку и $\max(C, D) + 1$ пар носков.

Решение, которое аккуратно разбирает все эти случаи (нужно не забыть, что ещё некоторые из данных чисел могут быть равны 0, поэтому не все указанные случаи возможны), набирает 100 баллов. Пример такого решения.

```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
ans = []
if a > 0 and c > 0:
    ans.append([b + 1, d + 1])
if b > 0 and d > 0:
    ans.append([a + 1, c + 1])
if a > 0 and b > 0:
    ans.append([max(a, b) + 1, 1])
if c > 0 and d > 0:
    ans.append([1, max(c, d) + 1])
m = min(ans, key=sum)
```

```
print(*m)
```

Пояснения по указанному решению. Здесь заводится список `ans`, в который будут складываться возможные варианты стратегии Саши, например, $(b+1, d+1)$ и т.д. Каждая стратегия — это пара чисел (тип данных кортеж). Кортежи мы будем добавлять в список, только проверив, что нет проблем с нулями, например, первая стратегия применима только в случае $a > 0$ и $c > 0$.

Затем мы в списке `ans` находим наименьший элемент используя функцию `min` с параметром `key=sum`. Это означает, что кортежи сравниваются по результату вызова функции `sum` от каждого кортежа, т.е. мы находим кортеж с минимальной суммой элементов.

Также частичные баллы можно набрать при помощи «переборного» решения, рассматривающего все возможные ответы, выбирающее среди них подходящий и наилучший среди подходящих. Например, решение сложности $O((a+b)(c+d))$ можно получить, перебирая количество выбранных маек x от 1 до $a+b$ и количество выбранных пар носков y от 1 до $c+d$. Далее проверим, может ли пара (x, y) являться ответом. Для этого проще проверить, что пара (x, y) **не может быть ответом**. Это происходит, если все выбранные майки и все выбранные носки могут оказаться разноцветными, то есть если $x \leqslant a$, $y \leqslant d$ или $x \leqslant b$, $y \leqslant c$.

В приведённом ниже решении функция `bad(x, y)` проверяет это условие, то есть проверяет, что пара (x, y) **не походит**. Поэтому в основной программе перебираются числа x и y и проверяется условие `not bad(x, y)`.

Пример переборного решения, набирающего 52 балла.

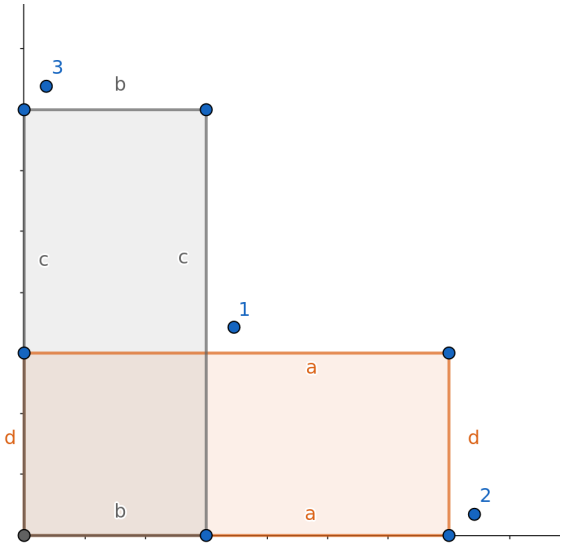
```
a = int(input())
b = int(input())
c = int(input())
d = int(input())
ans1 = a + b
ans2 = c + d

def bad(x, y):
    return x <= a and y <= d or x <= b or y <= c

for x in range(1, a + b + 1):
    for y in range(1, c + d + 1):
        if not bad(x, y) and x + y < ans1 + ans2:
            ans1 = x
            ans2 = y
print(ans1, ans2)
```

Переборное решение на 68 баллов можно получить, если перебирать только одно из чисел в ответе, а второе находить быстро.

Наконец, рассмотрение областей, которые не являются ответом, можно превратить и в решение на полный балл сложности $O(1)$. Пусть (x, y) — рассматриваемый ответ. Точка (x, y) **не подходит**, если она попадает в один из двух прямоугольников: $0 \leqslant x \leqslant a$, $0 \leqslant y \leqslant d$ или $0 \leqslant x \leqslant b$, $0 \leqslant y \leqslant c$. Значит, нужно выбрать точку, которая не принадлежит этим прямоугольникам и имеет минимальное значение $x + y$. В качестве таких точек имеет смысл рассматривать только три точки, изображённые на рисунке. Несложно видеть, что эти точки соответствуют разным возможным стратегиями Саши, описанным в начале разбора.



Задача 3. Престижный номер

Нам нужно получить минимальное число, произведение цифр которого равно N. В частности, ответ должен содержать как можно меньше цифр. Также отметим, что в ответе не может быть цифры 1 (кроме случая $n = 1$).

В разложение числа N на простые множители могут входить только простые меньшие 10: 2, 3, 5, 7. С делителями 5 и 7 просто, они войдут в ответ, как отдельные цифры. А вот двойки и тройки можно сгруппировать так, чтобы цифр стало меньше. Три двойки можно заменить на 8: $2^3 = 8$. Две тройки можно заменить на 9: $3^2 = 9$. Осталось не более двух двоек и не более одной тройки. Если осталась одна двойка и одна тройка, их можно заменить на 6. Если после этого осталось две двойки, их можно заменить на 4. Если после этого остались двойка или тройка, то они войдут в ответ, как самостоятельные цифры. Полученные цифры нужно вывести в порядке неубывания (от меньшей к большей).

Но есть и более простое решение. Оказывается, «выгодно» выделять как можно большие цифры в числе. Например, если $n = 18$ то из двух вариантов ответа 29 или 36 меньшим будет ответ 29. При этом в ответе 29 есть цифра 9, и если попробовать выделить сначала большую цифру 9, то после деления 18/9 получится меньший результат, который мы сможем использовать для старших разрядов записи ответа.

Получаем «жадное» решение — давайте делить наше число на все возможные цифры от 9 до 2 в порядке уменьшения, то есть попробуем в произведении выделить как можно больше девяток, затем как можно больше восьмёрок и т.д. Если в результате деления от числа n осталась 1, то нужно вывести найденные цифры в обратном порядке. Если же осталось число, большее 1, то значит у исходного n был делитель, больший 10, поэтому нужно вывести «-1». Также нужно отдельно обработать случай, когда $n = 1$, тогда ответ также равен 1. В приведённом ниже решении это происходит, если строка `ans` пустая, то есть если ни одного однозначного делителя у числа n не было найдено.

Пример решения на языке Python.

```
n = int(input())
ans = ""
for d in range(9, 1, -1):
    while n % d == 0:
        n //= d
        ans = str(d) + ans
if n == 1:
    print(ans or "1")
else:
```

```
    print(-1)
```

Решение на 30 баллов можно получить, если перебирать все числа, начиная с 1, пока не найдётся число, произведение цифр которого равно n. Пример такого решения.

```
n = int(input())
ans = 1

def prod(n):
    res = 1
    while n > 0:
        res *= n % 10
        n //= 10
    return res

ans = 1
while prod(ans) != n:
    ans += 1
print(ans)
```

Задача 4. Морской бой

Каждый корабль можно подбить отдельным выстрелом. Проще всего это сделать, выстрелив в момент времени $t = 0$ в начальную позицию корабля. Всегда будем подбивать отдельные корабли такими выстрелами, если мы не сможем подбить этот корабль в паре с другим кораблём.

Если есть корабль, который движется направо из точки r и корабль, который движется налево из точки l, при этом $r < l$, то их можно подбить одновременно в момент $\frac{l-r}{2}$ в точке $\frac{l+r}{2}$. Поскольку все начальные координаты кораблей — чётные, то все числа в ответе будут целыми.

Необходимо составить как можно больше пар кораблей (один движется направо, другой движется налево), которые можно подбить одновременно. Это можно сделать разными способами. Например, можно перебирать все корабли, двигающиеся налево, в порядке возрастания их координаты. Очередной корабль может пересечься со всеми кораблями, которые движутся направо, и имеют меньшую начальную координату. Выберем среди них какой-то подходящий.

В зависимости от деталей реализации можно получить решение сложности $O(n + m)$ или $O((n + m)^2)$. Например, решение сложности $O(n + m)$ можно получить используя метод «двух указателей». Для каждого корабля, двигающегося налево, мы попробуем подобрать корабль, двигающийся направо, с наименьшей начальной координатой, если только этот корабль не был уже сопоставлен другому кораблю, двигающемуся налево.

Заведём два указателя: переменная i — индекс корабля в списке кораблей, двигающихся налево, индекс j — индекс корабля в списке кораблей, двигающихся направо. Будем перебирать значения i. Для фиксированного i посмотрим, можно ли подбить одновременно i-й корабль, двигающийся налево и j-й корабль, двигающийся направо. Это можно сделать, если `right[j] <= left[i]`. Тогда мы подобьём эти два корабля одним выстрелом, и перейдём к следующим кораблям, увеличив i и j. Если условие не выполняется, то для i-го корабля, двигающегося налево, парного не нашлось, поэтому подобьём его отдельным выстрелом. Также в конце отдельными выстрелами подобъём оставшиеся корабли, двигающиеся направо, им тоже не нашлось пары.

Пример решения на языке Python.

```
n = int(input())
m = int(input())
left = [int(input()) for i in range(n)]
right = [int(input()) for i in range(m)]
i = 0
j = 0
while i < len(left):
```

```
    if j < len(right) and right[j] <= left[i]:
        print((left[i] - right[j]) // 2, (left[i] + right[j]) // 2)
        i += 1
        j += 1
    else:
        print(0, left[i])
        i += 1
while j < len(right):
    print(0, right[j])
    j += 1
```

Задача 5. Олимпиада по выживанию

Сначала рассмотрим решение сложности $O(n^2)$. Будем считать, что рейтинговая таблица участников хранится в массиве `a` из $4n$ элементов с индексами, начиная с 0.

Для каждой команды t найдём минимального участника, который не стал призёром, то есть это минимальное $i \geqslant 2n$, такое, что `a[i] == t`. Удобней просто перебирать всех участников, начиная с $i = 2n$, и пытаться i-го участника сделать призёром, при этом пропуская команды, для которых уже был найден ответ.

Пусть мы рассмотрели i-го участника и хотим сделать его призёром. Это означает, что все предыдущие участники также должны стать призёрами, то есть общее число призёров дожно быть не менее, чем $i+1$ (используется нумерация с нуля). Объявим $i+1$ участников призёрами. Тогда призёров будет больше половины от числа участников, и нам необходимо уменьшить и число участников, и число призёров, дисквалифицировав несколько команд. Так как мы хотим дисквалифицировать минимальное число команд, то сначала будем дисквалифицировать команды, имеющие по 4 призёра. Такая дисквалификация уменьшает и число участников на 4, и число объявленных призёров на 4. Если этого окажется недостаточно, будем дисквалифицировать команды, имеющие по 3 призёра. Дисквалификация такой команды уменьшает число участников на 4, а число объявленных призёров — на 3.

Если в итоге удалось добиться того, что количество объявленных призёров стало не больше, чем половина от количества участников, то мы нашли ответ для команды t. Дисквалифицировать команды, имеющие не более 2 призёров, не имеет смысла.

Пример такого решения на языке Python. Сложность решения равна $O(n^2)$ и это решение набирает 60 баллов. Значение `winners[i]` равно количеству призёров из команды i. Значение `teams[k]` равно количеству команд, имеющих ровно k призёров, нам будут нужны только значения `teams[4]` и `teams[3]`. При подсчёте этих значений также нужно пропустить рассматриваемую команду (её номер равен t), так так её нельзя дисквалифицировать.

```
n = int(input())
a = [int(input()) for i in range(4 * n)]
ans = [-1] * (n + 1)
for i in range(2 * n, 4 * n):
    t = a[i]
    if ans[t] != -1:
        continue;
    winners = [0] * (n + 1);
    for j in range(i + 1):
        winners[a[j]] += 1
    teams = [0] * 5
    for j in range(1, n + 1):
        if j != t:
            teams[winners[j]] += 1
    final_people = 4 * n
    final_win = i + 1
```

```
    while final_win > final_people // 2 and teams[4] > 0:
        teams[4] -= 1
        final_win -= 4
        final_people -= 4
    while final_win > final_people // 2 and teams[3] > 0:
        teams[3] -= 1
        final_win -= 3
        final_people -= 4
    if final_win <= final_people // 2:
        ans[t] = n - final_people // 4;
for i in range(1, n + 1):
    print(ans[i])
```

Улучшим это решение до сложности $O(n)$. Для этого заметим, что не стоит заново пересчитывать количества объявленных призёров из каждой команды и количества команд, которые имеют по k призёров, потому что при увеличении i на 1 к числу призёров добавляется ровно участник. Посчитаем сначала сколько участников из каких команд стало призёрами, рассмотрев первые $2n$ участников. Также посчитаем значения списка `teams` — сколько команд имеет 0, 1, 2, 3 или 4 призёров.

Далее будем перебирать участников по одному, начиная с $i = 2n$. Для i-го участника посчитаем, сколько «лишних» призёров образуется, если он должен стать призёром. Это число равно `extra_win = i + 1 - 2 * n`. Дисквалификация одной команды, имеющих 4 призёра, уменьшает значение `extra_win` на 3, а дисквалификация одной команды, имеющей 3 призёра, уменьшает значение `extra_win` на 1. Поэтому минимальное количество команд, которое нужно дисквалифицировать, можно посчитать за $O(1)$ — сначала попробуем дисквалифицировать `min((extra_win + 1) // 2, teams[4])` команд, которые имеют по 4 призёра, а потом команды, имеющие по 4 призёра.

Затем «обработаем» i-го участника, увеличив `winners[t]` на 1 и обновив значения списка `teams`.

```
n = int(input())
a = [int(input()) for i in range(4 * n)]
winners = [0] * (n + 1)
for i in range(2 * n):
    winners[a[i]] += 1
teams = [0] * 5
for i in range(1, n + 1):
    teams[winners[i]] += 1
ans = [-1] * (n + 1)
for i in range(2 * n, 4 * n):
    t = a[i]
    if ans[t] == -1:
        extra_win = i + 1 - 2 * n
        disqual_teams_4 = min((extra_win + 1) // 2, teams[4])
        extra_win -= disqual_teams_4 * 2
        if extra_win > 0:
            disqual_teams_3 = min(teams[3], extra_win)
            if disqual_teams_3 == teams[3] and winners[t] == 3:
                disqual_teams_3 -= 1
        else:
            disqual_teams_3 = 0
        if disqual_teams_3 >= extra_win:
            ans[t] = disqual_teams_4 + disqual_teams_3
    teams[winners[t]] -= 1
    winners[t] += 1
```

```
    teams[winners[t]] += 1
for i in range(1, n + 1):
    print(ans[i])
```