Задача A. Газонокосилка

Заметим, что оптимальный способ вынимать доски это убриать подряд идущую $w-1$ доску, а затем оставлять одну. Таким образом останется каждая w-я доска. Тогда ответ $n - \left\lfloor \frac{n}{w} \right\rfloor$.

Задача B. Серебряный статус в авиакомпании

При данных ограничениях задачу проще всего решать перебором вариантов. А именно, переберём, сколько пакетов по 30 000 будем менять на 5 000 квалификационных, а из оставшихся средств — сколько пакетов по 7 500 миль будем менять на 1 000 каждый. Из этих вариантов выберем тот, при котором количество квалификационных миль будет не менее 20 000, а общее число миль будет как можно больше.

Код программы на языке Python:

```
n = int(input())
k = int(input())
mx = 0
for i in range(n // 30000 + 1):
    for j in range((n - i * 30000) // 7500 + 1):
        nn = n - i * 30000 - j * 7500
        kk = k
        if nn < k:
            kk = nn #  квалификационные мили были потрачены на обмен
        kk += i * 5000 + j * 1000 #  новое число квалификационных миль
        nn += i * 5000 + j * 1000 #  новое число миль
        if kk >= 20000: #  квалификационных миль хватает для статуса
            if nn > mx:
                mx = nn
if mx > 0:
    print(mx)
else:
    print(-1)
```

Но задачу можно решить и разбором случаев. Определим, сколько пакетов по 1 000 квалификационных миль нам не хватает до достижения 20 000 миль. Разделим это количество на пять и обменяем как минимум такое количество пакетов по 5 000 миль, остаток обменяем по 1 000 миль. Но если осталось четыре пакета по 1 000, то вместо них выгоднее обменять дополнительный пакет в 5 000 миль, потому что при той же стоимости мы получим в итоге на 1 000 миль больше. При этих необходимых операциях на обмены могут расходоваться и имевшиеся квалификационные мили, что может привести к невозможности получения серебряного статуса, даже когда общего количества миль на обмены хватает.

```
n = int(input())
k = int(input())
#  используем округление вверх до 1000 при подсчете недостающих пакетов
l = (max(0, 20000 - k) + 999) // 1000
if l % 5 == 4:
    #  выгоднее обменять лишний пакет в 5000 миль вместо четырех по 1000 миль
    l5 = (l + 1) // 5
    #  квалификационные мили с учетом возможного их уменьшения при обмене
    k = min(k, n - 30000 * l5) + 5000 * l5
    n -= 25000 * l5 #  оставшиеся мили
else:
    l1 = l % 5
    l5 = l // 5
    k = min(k, n - 30000 * l5 - 7500 * l1) + 5000 * l5 + 1000 * l1
    n -= (25000 * l5 + 6500 * l1) #  оставшиеся мили
if k >= 20000: #  квалификационных миль достаточно
    print(n)
else:
```

```
    print(-1)
```

Задача C. Офшоры

Пусть мы хотим сделать i-й элемент максимумом. Посчитаем $S_i = \sum\limits_{j=1\ldots n}^{j\neq i} \lfloor \frac{a_j}{x} \rfloor$, заметим что максимально мы можем сделать именно столько переводов на i-й счет со всех остальных суммарно (так как выполняется условие $y \leqslant x$, то если мы сделаем перевод с какого-то счета u на какой-то счет v, то $\lfloor \frac{a_u}{x} \rfloor$ уменьшится ровно на 1, а $\lfloor \frac{a_v}{x} \rfloor$ увеличится не более чем на 1, поэтому промежуточные переводы между счетами не помогут увеличить S_i). Таким образом задача свелась к тому чтобы научиться быстро считать S_i для каждого i. Для этого можно посчитать $S_{all} = \sum\limits_{j=1}^{n} \lfloor \frac{a_j}{x} \rfloor$, тогда $S_i = S_{all} - \lfloor \frac{a_i}{x} \rfloor$.

Задача D. Шифровка

Подгруппа 1 ($n \leqslant 10$).

В этой подзадаче можно было перебрать все возможные варианты расшифровки (для каждой позиции выбрать, какую из двух букв мы возьмём), для каждого варианта определить его информативность и выбрать оптимальную расшифровку. Информативность строки можно найти, честно перебрав d и проверив, верно ли, что s представляет собой префикс длины d, повторённый $\frac{n}{d}$ раз. Такую проверку можно осуществить за $O(n)$, при этом проверять имеет смысл только те d, которые являются делителями n. Поскольку делителей n не более чем $2\sqrt{n}$, то нахождение информативности будет работать за $O(n\sqrt{n})$. Всего существует 2^n различных вариантов расшифровок, поэтому общая асимптотика алгоритма — $O(2^n n\sqrt{n})$.

Подгруппа 2.

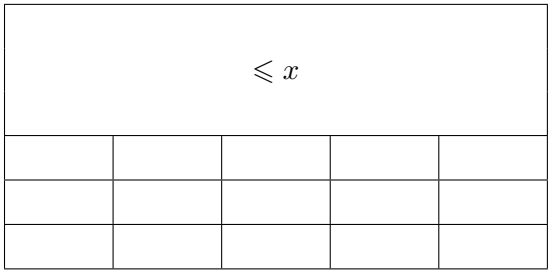
В этой подзадаче можно было заметить, что оптимальной будет расшифровка, содержащая только буквы `a` и `c`. Действительно, ведь при такой расшифровке буквы совпадают на всех парах позиций, на которых они вообще могли бы совпасть, поэтому информативность в таком случае будет минимально возможной. Построить требуемую расшифровку можно за $O(n)$.

Полное решение.

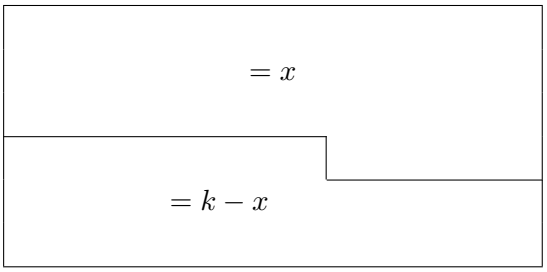
Давайте, как в первой подзадаче, будем перебирать d (делитель n) и проверять, можем ли мы получить строку с информативностью d. Для этого для каждого $1 \leqslant i \leqslant d$ надо проверить, что у позиций $i, i+d, i+2d, \ldots$ существует общая буква. Для этого переберём букву на позиции i и проверим, что на позициях $i+d, i+2d, \ldots$ эта буква также присутствует. Для позиции i проверку можно осуществить за $O\left(\frac{n}{d}\right)$, для всех i это будет работать за $O(n)$. Суммарно по всем d алгоритм будет работать за $O(n\sqrt{n})$.

Задача E. Разрез таблицы

Пусть в таблице k единиц. Тогда максимально возможное произведение $\left\lfloor \frac{k}{2} \right\rfloor \cdot \left(k - \left\lfloor \frac{k}{2} \right\rfloor \right)$. Покажем, что ответ всегда достигается. Хотим выделить разрезом часть с $x = \left\lfloor \frac{k}{2} \right\rfloor$. Найдем последнюю строку, что сумма единиц на префиксе строк $\leqslant x$.



В следующей строке возьмем на суффиксе нужное количество элементов чтобы сумма сверху стала равна x.



После этого разрез восстанавливается из рисунка выше. Асимптотика $\mathcal{O}(n \cdot m)$.