Задача 1. Волшебный сундук

Автор задачи: Инесса Шуйкова, разработчик: Егор Ермолов

Так как мы достаем монеты из сундука вслепую, необходимо рассмотреть наихудший сценарий. Сделаем это для каждого из трех вариантов.

В первом варианте у нас есть всего три монеты, поэтому, очевидно, придется достать их все. Таким образом, ответ равен 3.

Во втором варианте у нас есть 4 золотых монеты, 6 серебряных монет и 3 медные монеты. В худшем случае сначала мы достанем 6 серебряных монет, которые превратятся в медные. Затем мы достанем 4 золотые монеты, которые превратятся в серебряные. Наконец, после этого достаточно будет взять из сундука одну монету, которая гарантированно будет медной, и после взятия превратится в золотую. Таким образом, ответ равен $6 + 4 + 1 = 11$.

В третьем варианте у нас есть 13 золотых монет, 47 серебряных монет и 53 медные монеты. В худшем случае сначала мы достанем 53 медные монеты, которые превратятся в золотые. Затем мы достанем 47 серебряных монет, которые превратятся в медные. Наконец, после этого достаточно будет взять из сундука одну монету, которая гарантированно будет золотой, и после взятия превратится в серебряную. Таким образом, ответ равен $53 + 47 + 1 = 101$.

Итак, ответ на задачу равен `3 11 101`.

Задача 2. Сэндвич

Автор задачи: Инесса Шуйкова, разработчик: Егор Ермолов

Вычислим количество сэндвичей с сыром. Нижний уровень состоит из m сэндвичей с курицей. Второй снизу уровень состоит из $m - 1$ сэндвичей с сыром. Третий снизу уровень состоит из $m - 1$ сэндвичей с курицей. Четвертый снизу уровень состоит из $m - 2$ сэндвичей с сыром. И так далее.

Здесь легко увидеть закономерность — количество сэндвичей с сыром будет равно $(m - 1) + (m - 2) + \ldots + 1$. Эта сумма является арифметической прогрессией, и она равна $\frac{m(m-1)}{2}$.

Таким образом, ответ на задачу равен `m * (m - 1) / 2`.

Задача 3. Пасека — это бизнес, и бизнес идет хорошо

Автор задачи: Инесса Шуйкова, разработчик: Егор Ермолов

В качестве наивного решения данной задачи можно перебрать все клетки поля и вычислить количество белых и синих ульев. Для удобства пронумеруем строки числами от 0 до $n-1$, а столбцы — числами от 0 до $m - 1$. Тогда на пересечении i-й строки и j-го столбца будет находиться клетка с координатами (i, j).

Заметим, что улей в клетке (i, j) будет белым в случае, если $(i + j) \equiv 0 \pmod 2$. В противном случае данный улей будет синим. Таким образом, по сути в задаче описана шахматная раскраска доски размера $n \times m$.

Переберем все клетки исходного поля и посчитаем количество белых и синих ульев. Затем сделаем то же самое для увеличенного поля и вычислим количество ульев каждого цвета, которые необходимо будет докупить.

Данное решение будет работать за $\mathcal{O}(nm)$ и наберет не менее 40 баллов.

Для того, чтобы решить задачу на 100 баллов, заметим некоторые свойства увеличенного поля.

Во-первых, заметим, что в результате увеличения будут добавлены $2n + 2m + 4$ новых клеток. Также заметим, что добавленные клетки все еще должны удовлетворять шахматной раскраске, поэтому цвета клеток будут чередоваться. А значит, количество клеток каждого типа будет равно $\frac{2n+2m+4}{2} = n + m + 2$.

Также необходимо не забыть воспользоваться 64-битным типом данных в случае, если вы используете такие языки как C++, Java, Pascal и C#.

Данное решение будет работать за $\mathcal{O}(1)$ и наберет 100 баллов.

Пример решения задачи на языке Python:

```
n = int(input())
m = int(input())
print(n + m + 2, n + m + 2)
```

Задача 4. Музыкальные аккорды

Автор задачи: Инесса Шуйкова, разработчик: Михаил Первеев

В качестве наивного решения данной задачи можно перебрать все подмножества нот, и для каждого подмножества проверить, действительно ли разница между самой высокой и самой низкой нотами в данном подмножестве не больше d. Если это так, то данное подмножество представляет собой вариант гармоничного аккорда. Остается только выбрать наибольшее из подходящих подмножеств. Реализовать перебор всех подмножеств можно с помощью рекурсивного перебора, а после того как было построено очередное подмножество, проверить, подходит ли оно, можно, найдя ноты с максимальной и минимальной высотами и проверив, что разница между ними не более d. Итого получаем $\mathcal{O}(2^n)$ на перебор всех подмножеств и $\mathcal{O}(n)$ на проверку каждого подмножества. Итоговое время работы: $\mathcal{O}(2^n \cdot n)$. Данное решение наберет не менее 20 баллов.

Для решения на 40 баллов заметим следующее свойство. Если нота с наименьшей высотой в выбранном гармоничном аккорде будет иметь высоту a, то все остальные ноты будут иметь высоты в отрезке $[a, a + d]$. Причем любую ноту, высота которой лежит в указанном отрезке, можно беспрепятственно добавлять в аккорд так, чтобы он все еще остался гармоничным. Тогда можно утверждать, что гармоничный аккорд наибольшего размера, в котором наименьшая высота ноты равна a, просто должен содержать все ноты, высоты которых содержатся в отрезке $[a, a + d]$. Переберем ноту, которая будет иметь наименьшую высоту, пусть ее высота оказалась равна a. После этого посчитаем количество нот, высоты которых лежат в отрезке $[a, a + d]$. Из всех посчитанных количеств нужно выбрать максимальное — оно и будет ответом. Итоговое время работы: $\mathcal{O}(n^2)$. Данное решение наберет не менее 40 баллов.

Для полного решения задачи оптимизируем предыдущее решение. Нам нужно избавиться от перебора всех нот после того, как уже зафиксирована наименьшая по высоте нота с высотой a. Для этого отсортируем ноты по неуменьшению высоты. Заметим, что теперь, если нам нужно найти все ноты, высоты которых лежат в отрезке $[a, a+d]$, то данные ноты образуют непрерывный подотрезок в отсортированном массиве всех нот. Тогда переберем наименьшую по высоте ноту в порядке увеличения высоты. Заметим, что тогда наибольшая высота ноты в рассматриваемых нами отрезках тоже будет не уменьшаться — действительно, если наименьшая высота ноты увеличилась, то уменьшать наибольшую высоту не нужно, так как разница между ними не увеличивалась. Тогда можем поддерживать отрезок из нот, соответствующий наибольшему гармоничному аккорду с фиксированной нотой наименьшей высоты при помощи двух указателей. Левый указатель будет отвечать за ноту с наименьшей высотой, а правый — за ноту с наибольшей высотой. Так как оба указателя двигаются только вправо, то сложность перебора отрезков будет $\mathcal{O}(n)$, однако стоит помнить, что нам еще нужно отсортировать ноты по неуменьшению высоты, это можно сделать за $\mathcal{O}(n \log n)$. Итоговое время работы: $\mathcal{O}(n \log n)$.

Пример решения задачи на языке Python:

```
n = int(input())
d = int(input())
a = []

for i in range(n):
    a.append(int(input()))
a.sort()

ans = 0
r = 0
for l in range(n):
```

```
    while r < n and a[r] - a[l] <= d:
        r += 1
    ans = max(ans, r - l)

print(ans)
```

Задача 5. Вареники

Автор задачи: Инесса Шуйкова, разработчик: Егор Ермолов

Для решения данной задачи необходимо найти последний вареник, съеденный Петей. Будем перебирать этот вареник. Пусть сейчас мы хотим проверить, является ли k-й вареник последним вареником, который съест Петя. Для этого посчитаем время t_P, через которое Петя съест все вареники до k-го. Также посчитаем время t_T, через которое Таня съест все вареники после $k+1$-го вареника. Тогда необходимо проверить, что Таня не съест k-й вареник, а Петя не съест $k+1$-й вареник — это и будет означать, что k-й вареник Петя съел последним.

Чтобы проверить, что Петя не съел $k+1$-й вареник, необходимо проверить, что после съедания Петей k-го варенника Таня уже приступила, или собирается приступить к $k+1$-му варенику, а именно $t_P + t_k >= t_T$. Аналогично необходимо проверить, что Таня не съест k-й вареник, а именно, что $t_T + t_{k+1} > t_P$. Тогда, проверив данные свойства для всех вареников, мы найдем последний вареник, который съел Петя (возможно, это будет «нулевой» вареник в ситуации, когда Петя не успеет съесть ни одного вареника).

Пусть номер найденного вареника равен v. Несложно заметить, что тогда Петя съел v вареников, а Таня — $n - v$ вареников.

Считать t_P и t_T будем каждый раз явно, суммируя время поедания соответствующих вареников. Тогда мы потратим $\mathcal{O}(n)$ времени для перебора вареника и еще $\mathcal{O}(n)$ для каждого вареника, чтобы посчитать t_P и t_T. Итоговое время работы составит $\mathcal{O}(n^2)$. Данное решение будет набирать не менее 40 баллов.

Для того, чтобы решить задачу на 100 баллов, нужно заметить, что при подсчете t_P и t_T мы много раз пересчитываем одну и ту же сумму. Ведь t_P — это сумма времен поедания на каком-то префиксе вареников, а t_T — это сумма времен поедания на каком-то суффиксе вареников. Вместо того, чтобы на каждую проверку заново их пересчитывать, заранее посчитаем все префиксные и суффиксные суммы времен поедания вареников. Тогда мы избавимся от подсчета t_P и t_T за $\mathcal{O}(n)$ на проверку и будем это делать за $\mathcal{O}(n)$ предподсчета и $\mathcal{O}(1)$ на проверку. Итоговое время работы составит $\mathcal{O}(n)$.

Пример решения задачи на языке Python:

```
import sys

n = int(input())
arr = [0] * (n + 2)
for i in range(1, n + 1):
    arr[i] = int(input())

pref = [0] * (n + 3)
suf = [0] * (n + 3)

for i in range(1, n + 3):
    pref[i] = pref[i - 1] + arr[i - 1]
    suf[n + 2 - i] = suf[n + 2 - i + 1] + arr[n + 2 - i]

for i in range(0, n + 1):
    t_p = pref[i]
    t_t = suf[i + 2]
```

```
    if t_p + arr[i] >= t_t and t_t + arr[i + 1] > t_p:
        print (i, n - i)
        sys.exit(0)
```