Задача A. Максимальный Делитель

Для начала опишем тривиальное решение данной задачи. Необходимо перебрать все делители числа N, после чего выбрать наибольший делитель. При этом следует перебирать делители от единицы до $\lfloor\sqrt{N}\rfloor$.

Асимптотика: $O(\sqrt{N})$.

Для решения задачи на полный балл необходимо обратить внимание на необычный вид числа N, которое выглядит так: $999\ldots9$ (цифра 9, повторенная k раз). Заметим, что такое число можно представить как $3 \cdot 333\ldots3$. Таким образом, число $333\ldots3$ является максимальным подходящим делителем. Не следует забывать, что данное число не поместится даже в 64-битный тип данных, поэтому для вывода необходимо просто вывести символ «3» k раз.

Асимптотика: $O(k)$.

Пример решения на языке C++:

```
#include <cstdio>

int main() {
    int k;
    scanf("%d", &k);

    for (int i = 0; i < k; ++i) {
        printf("3");
    }
    printf("\n");

    return 0;
}
```

Задача B. Перекраска Одежды

Рассмотрим решение задачи для первой группы тестов. Для этого посчитаем, сколько есть футболок первого и второго цвета. Обозначим эти количества как c_1 и c_2. Если какое-то из этих чисел не меньше, чем k, то перекрашивать футболки не придется, и ответ на задачу равен нулю. В противном случае придется перекрасить некоторые футболки. Нетрудно понять, что придется перекрасить $\min(k - c_1, k - c_2)$ футболок.

Асимптотика: $O(N)$.

Для решений второй подзадачи можно использовать следующий алгоритм. Переберем цвет, в который мы будем перекрашивать футболки. Пройдем циклом по всем футболкам и посчитаем, сколько есть футболок выбранного цвета, пусть их количество равно c. Тогда придется перекрасить $\max(0, k - c)$ футболок. Найдем минимальное значение данного количества по всем цветам, это и будет ответ на задачу.

Асимптотика: $O(N \cdot C)$.

Теперь рассмотрим полное решение данной задачи. Из решения для второй подзадачи нетрудно понять, что чем больше количество футболок некоторого выбранного цвета, тем меньше придется купить краски. Заведем массив cnt, в i-й ячейке которого сохраним, сколько есть футболок i-го цвета. Это можно сделать при считывании данных. Теперь пройдем циклом по данному массиву и найдем цвет, который встречается чаще всего (найдем максимальное значение cnt_i). Пусть это значение равно $maxC$. Тогда ответ равен: $\max(0, k - maxC)$.

Асимптотика: $O(N)$.

Пример решения на языке C++:

```
#include <cstdio>
#include <algorithm>
```

```
using namespace std;

const int N = 100100;

int cnt[N];

int main() {
    int n, k;
    scanf("%d%d", &n, &k);

    for (int i = 0; i < n; ++i) {
        int cur;
        scanf("%d", &cur);
        ++cnt[cur];
    }

    int mx = 0;
    for (int i = 0; i < N; ++i) {
        mx = max(mx, cnt[i]);
    }

    printf("%d\n", max(0, k - mx));

    return 0;
}
```

Задача C. Сложная Задача

Рассмотрим решение первой подзадачи. Так как значение N довольно небольшое, можно перебрать, какое из двух действий мы будем делать на каждом шаге. Подобный перебор работает за 2^N действий, так как на каждом из N шагов можно выбрать одно из двух действий.

Асимптотика: $O(2^N)$.

Теперь рассмотрим полное решение задачи. Заметим, что на каждой итерации мы заменяем одно из чисел на $A + B$, после чего продолжаем работу. Нетрудно доказать, что выгоднее заменять меньшее из чисел A и B на их сумму, так как в будущем это даст больший вклад в сумму. Таким образом, будем выполнять действия последовательно, заменяя меньшее из чисел на $A + B$. Для получения максимального балла, нужно не забыть воспользоваться 64-битным типом данных, так как ответ может получиться достаточно большим.

Асимптотика: $O(N)$.

Пример решения на языке C++:

```
#include <cstdio>
#include <algorithm>

using namespace std;

int main() {
    long long a, b;
    int n;

    scanf("%lld%lld%d", &a, &b, &n);
```

```
    for (int i = 0; i < n; ++i) {
        if (a < b) {
            a += b;
        } else {
            b += a;
        }
    }

    printf("%lld\n", max(a, b));

    return 0;
}
```

Задача D. Игра в Города

Решение на первую подзадачу достаточно несложное. Нужно сохранить в массиве все слова, которые запоминает Миша, а в момент, когда нужно для некоторого слова подобрать продолжение, нужно пройтись циклом по массиву и проверить каждое слово.

Асимптотика: $O(N^2)$.

Для полного решения нужно быстро отвечать на запросы Миши. Для этого заведем массив, состоящий из 26 элементов — столько существует латинских букв. Для каждой буквы сохраним в массиве, сколько существует слов, которые Миша запомнил, которые начинаются на текущую букву.

Теперь для ответа на запрос необходимо вывести содержимое ячейки массива, соответствующей последней букве запрашиваемого слова.

Асимптотика $O(N)$.

Пример решения на языке C++:

```
#include <iostream>
#include <string>

using namespace std;

int cnt[26];

int main() {
    int n;
    cin >> n;

    for (int i = 0; i < n; ++i) {
        int type;
        string s;

        cin >> type >> s;

        if (type == 1) {
            ++cnt[s[0] - 'a'];
        } else {
            cout << cnt[s.back() - 'a'] << '\n';
        }
    }
```

```
    return 0;
}
```

Задача E. Упаковка Груза

Для решения первой подзадачи необходимо перебрать все подмножества коробок, проверить каждое из них на корректность и найти корректное подмножество максимального размера. Реализовать это можно следующим образом. Для начала отсортируем коробки в порядке неубывания размера. Далее реализуем рекурсивный (или нерекурсивыный) перебор всех подмножеств множества коробок. Пусть в подмножество вошли коробки с номерами $i_1, i_2, \ldots, i_k$. Тогда для того, чтобы проверить набор на корректность, необходимо проверить что для любого $j \geqslant 2$ верно, что $\frac{s_{i_j}}{s_{i_{j-1}}} \geqslant K$ (данной проверки достаточно, так как коробки были отсортированы).

Асимптотика: $O(2^N \cdot N)$.

Рассмотрим решение второй подзадачи. В ней $K = 10^9$ — максимально возможное значение. Нетрудно понять, что при таком значении K ответ не бывает больше двух (размеры 1 и 10^9). Поэтому в случае, если есть хотя бы одна коробка размера 1 и хотя бы одна коробка размера 10^9, ответ будет равен двум. Во всех остальных случая ответ будет равен одному.

Асимптотика: $O(N)$.

Рассмотрим полное решение данной задачи. Отсортируем коробки по неубыванию размера. Далее будем набирать коробки жадно, в порядке сортировки. Нетрудно показать, что данный алгоритм оптимален. То есть, для начала возьмем в набор коробку минимального размера. После этого будем пропускать коробки больших размеров до тех пор, пока мы не сможем взять следующую коробку. Берем первую следующую подходящую коробку, после чего продолжаем алгоритм.

Асимптотика: $O(N \log N)$.

Пример решения на языке C++:

```
#include <cstdio>
#include <algorithm>

using namespace std;

const int N = 100100;

int a[N];

int main() {
    int n, k;
    scanf("%d%d", &n, &k);

    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }

    sort(a, a + n);
    reverse(a, a + n);

    int ans = 1;
    int last = a[0];

    for (int i = 1; i < n; ++i) {
        if (1ll * a[i] * k <= last) {
            last = a[i];
```

```
            ++ans;
        }
    }

    printf("%d\n", ans);

    return 0;
}
```

Задача F. Решение Задач

Для решения первой подзадачи достаточно перебрать всевозможные пары чисел (X, Y), где $1 \leqslant X, Y \leqslant N$ и посчитать количество подходящих пар (тех, для которых выполнено $\frac{X}{Y} = \frac{A}{B}$). Проверять данный критерий можно либо воспользовавшись дробными числами, либо преобразовав выражение и сравнив числа $X \cdot B$ и $A \cdot Y$.

Асимптотика: $O(N^2)$.

Для решения второй подзадачи заметим, что можно перебрать число X, после чего число Y определяется однозначно (в случае, если существует подходящая пара с таким значением X). Пусть мы зафиксировали некоторое число X, такое что $1 \leqslant X \leqslant N$. Должно быть выполнено равенство $X \cdot B = A \cdot Y$. Отсюда $Y = \frac{X \cdot B}{A}$. Если $X \cdot B$ не делится на A, данное значение X не подходит. В противном случае вычислим Y и проверим, что он лежит в диапазоне $[1, N]$. Следует обратить внимание, что нужно использовать 64-битный тип данных, чтобы избежать переполнения при умножении.

Асимптотика: $O(N)$.

Для решения третьей и четвертой подзадач нужно заметить несколько дополнительных фактов. В задаче требуется найти количество дробей, равных исходной дроби, числитель и знаменатель которых не превосходят N. Заметим, что дроби $\frac{A}{B}$ и $\frac{A \cdot K}{B \cdot K}$ равны для любого $K \geqslant 1$. Также, для всех K таких, что A и B делятся на K, можно получить равные дроби, разделив числитель и знаменатель на K. Для того, чтобы не рассматривать отдельно эти два случая, поделим изначально A и B на их наибольший общий делитель: пусть $D = gcd(A, B)$ — наибольший общий делитель A и B. Тогда скажем, что $A := \frac{A}{D}$, и $B := \frac{B}{D}$. Теперь задача сведена к тому, что нужно посчитать количество целых чисел $K \geqslant 1$, таких что $A \cdot K$ и $B \cdot K$ лежат в диапазоне $[1, N]$. Нетрудно заметить, что это количество равно $\min\left(\left\lfloor \frac{N}{A} \right\rfloor, \left\lfloor \frac{N}{B} \right\rfloor\right)$.

Таким образом, самая сложная часть задачи — вычисление наибольшего общего делителя. Для решения третьей подзадачи достаточно вычислить наибольший общий делитель при помощи разложения числа на простые множители.

Асимптотика: $O(\sqrt{N})$.

Для полного решения задачи необходимо вычислить наибольший общий делитель при помощи алгоритма Евклида. Также можно воспользоваться встроенными в некоторые языки функциями, вычисляющими НОД чисел. Например, `math.gcd` в языке Python, `std::gcd` в языке C++.

Асимптотика: $O(\log N)$.

Пример решения на языке C++:

```
#include <cstdio>
#include <algorithm>

using namespace std;

long long gcd(long long a, long long b) {
```

```
    if (b == 0) {
        return a;
    }
    return gcd(b, a % b);
}

int main() {
    long long n, a, b;
    scanf("%lld%lld%lld", &n, &a, &b);

    long long d = gcd(a, b);
    a /= d;
    b /= d;

    printf("%lld\n", min(n / a, n / b));

    return 0;
}
```