Робот-пылесос

В этой задаче требовалось перебрать все возможные случаи расположения базы относительно робота-пылесоса. Например, программа должна вывести 2 при условиях $x > x2$ $y > y2$, вывести 4 при условиях y $<$ y1 и x $>$ x2 и т. д. Необходимо аккуратно разобрать все случаи.

Пример реализации на языке Python:

```
x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
x = int(input())
y = int(input())
if x > x2 and y > y2:
    print(2)
elif x > x2 and y < y1:
    print(4)
elif x < x1 and y < y1:
    print(6)
elif x < x1 and y > y2:
    print(8)
elif y > y2:
    print(1)
elif y < y1:
    print(5)
elif x > x2:
    print(3)
else:
    print(7)
```

Кондитерская фабрика

На неполный балл можно написать переборное решение. Переберём все числа, пока не будет найдено число, которое даёт остаток от деления N при делении на K и остаток от деления $(L - M)$ при делении на L.

Пример реализации на языке программирования Python

```
K = int(input())
L = int(input())
N = int(input())
M = int(input())
t = K + N
while t % K != N or t % L != (L - M):
    t += 1
print(t)
```

Для оптимизации решения мы можем перебирать не все числа, а только те, которые дают остаток N при делении на K. Поскольку мы начали с числа $t = K + N$, которое при делении на K даёт остаток N, то дальше нам нужно прибавлять к этому числу K ячеек, что бы среди каждого Kтого числа найти то, которое будет давать остаток $(L - M)$ при делении на L.

Пример реализации на языке программирования Python

```
K = int(input())
L = int(input())
N = int(input())
M = int(input())
t = K + N
while t % L != (L - M):
    t += K
print(t)
```

Для того, чтобы набрать максимальный балл, необходимо выбрать из чисел L и K максимальное, а так же поставить соответствующий шаг в цикле.

```
K = int(input())
L = int(input())
N = int(input())
M = int(input())
if K >= L:
    t = K + N
    while t % L != (L - M):
        t += K
else:
    t = L - M
    if t < K:
        t += L
    while t % K != N:
        t += L
print(t)
```

От нуля до девяти

Для того, что бы набрать неполный балл достаточно написать переборное решение, когда элементы массива обрабатываются по очереди. Требуется запоминать для каждого числа, сколько раз оно было встречено в массиве. Для этого можно завести массив ans, в котором ans_i равно количеству вхождений числа i. Вычислительная сложность такого алгоритма будет $O(n \cdot C)$, где C - максимальное число в массиве.

Что бы оптимизировать алгоритм можно перебрать каждую пару строк (i, j), после чего проверить, что $s_i + s_j$ является невозрастающей последовательностью. Если строка не подходит под это условие, с определённого символа, в таком случае следует прервать её обработку. Вычислительная сложность такого алгоритма будет $O(n^2 + m)$.

Для дальнейшей оптимизации решения заметим, что если строка s_i не удовлетворяет условию не возрастания, тогда её нет смысла складывать с любой другой строкой, результат не будет удовлетворять условию. По этой причине следует обрабатывать только те строки, которые удовлетворяют условию. Заметим так же, что если первый символ $s_i < s_j$, то результат сложения двух строк так же будет удовлетворять условию задачи. Вычислительная сложность такого алгоритма будет $O(n^2+m)$.

Что бы еще ускорить решение, будем быстро искать количество подходящих строк для строки s_j. Данное количество равно количеству подходящих строк встреченных ранее, чем s_j. Для реализации этой идеи в ассоциативном массиве для каждой последней цифры строки от 0 до 9 будем хранить количество встреченных ранее подходящих строк оканчивающихся на одну из указанных цифр. Вычислительная сложность такого алгоритма будет $O(n + m)$.

Пример реализации на языке программирования python

```
def eq(s):
    for i in range(len(s) - 1):
        if s[i] > s[i + 1]:
            return False
    return True
n = int(input())
a = []
for i in range(n):
    a.append(input())
ans = 0
cnt = [0] * 10
for i in range(n - 1, -1, -1):
    if not eq(a[i]):
        continue
    last = int(a[i][-1])
    for j in range(last, 10):
        ans += cnt[j]
    cnt[int(a[i][0])] += 1
print(ans)
```

Пример реализации на языке программирования C++

```
#include <bits/stdc++.h>

using ll = long long;

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int n;
    std::cin >> n;
    std::vector<std::string> a(n);
    for (auto &item : a) {
```

```
        std::cin >> item;
    }

    ll ans = 0;
    std::vector<int> cnts(10);
    for (int i = n - 1; i >= 0; i--) {
        if (!std::is_sorted(a[i].begin(), a[i].end())) {
            continue;
        }
        int last = a[i].back() - '0';
        for (int j = last; j < 10; j++) {
            ans += cnts[j];
        }
        cnts[a[i][0] - '0']++;
    }

    std::cout << ans << '\n';
}
```

Ремонт дороги

Данную задачу можно решить двумя способами: 1 способ - будем хранить словарь (ассоциативный массив), в котором ключами будет являться длина участков дороги, а значение – количество участков, имеющих заданную длину. Извлекаем самый длинный участок: если количество участков не больше, чем осталось разделить - делим все. Если количество участков больше – делим на части необходимое количество.

Пример реализации на языке Python:

```
n = int(input())
k = int(input())
q = int(input())
queries = list(map(int, input().split()))
lengths = {n: 1}
while k > 0:
    max_len = max(lengths.keys())
    count = lengths[max_len]
    if k > count:
        k -= count
        lengths[max_len // 2] = lengths.get(max_len // 2, 0) + count
        lengths[(max_len + 1) // 2] = lengths.get((max_len + 1) // 2, 0) + count
        del lengths[max_len]
    else:
        lengths[max_len // 2] = lengths.get(max_len // 2, 0) + k
        lengths[(max_len + 1) // 2] = lengths.get((max_len + 1) // 2, 0) + k
        lengths[max_len] -= k
        k = 0
result = []
for query in queries:
    total_count = 0
    for length, count in sorted(lengths.items(), reverse=True):
        total_count += count
        if query <= total_count:
            result.append(length)
            break
print(*result)
```

2 способ - заметим, что различных текущих длин всегда не больше 4

$l, l+1, 2l, 2l+1$ или

$l, l+1, 2l-1, 2l$

Поддерживаем, какой из двух случаев имеет место и количество участков каждой из длин.

Подготовка к олимпиаде

Задача решается методом динамического программирования. Функция динамического программирования $dp[i]$ – это максимальное суммарное количество решенных задач из наборов $0 \dots i$. Получим формулу для пересчета значений dp: $dp[i] = max(dp[j] + a[i])$ для всех $j = 0 \dots i - 1$, таких что $a[j] < a[i]$.

Пример реализации на языке программирования Python:

```
n = int(input())
a = list(map(int, input().split()))
dp = [0] * n
for i in range(n):
    dp[i] = a[i]
    for j in range(i):
        if a[j] < a[i] and dp[j] + a[i] > dp[i]:
            dp[i] = dp[j] + a[i]
print(max(dp))
```