Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Решения задач блока 2

Алгоритмам и структурам данных

Работу выполнил:

Гаврилин О.С.

Группа:

P3230

Санкт-Петербург,

2025

1.E

```
#include <ios>
#include <iostream>
#include <vector>

bool    is_configure_cows(const    std::vector<int>&
stables, const int k, const int mid) {
    int count = 1;
    int last_position = stables[0];
    for (size_t i = 1; i < stables.size(); i++) {
        if (stables[i] - last_position >= mid) {
            count++;
            last_position = stables[i];
        }
        if (count == k) return true;
    }
    return false;

}

void solve() {
    int n, k;
    std::cin >> n >> k;
    std::vector<int> stables(n);
    for (int i = 0; i < n; i++) {
        std::cin >> stables[i];
    }

    int lower_bound = 0, upper_bound = stables[n -
1] - stables[0];
    int max = 0;
    while (lower_bound <= upper_bound) {
        int  mid  =  lower_bound  +  (upper_bound  -
lower_bound) / 2;
        if (is_configure_cows(stables, k, mid)) {
            max = mid;
            lower_bound = mid + 1;
```

```
            } else upper_bound = mid - 1;
        }
        std::cout << max;
    }

    int main() {
        std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);
        int t = 1;
        while (t--) {
            solve();
        }
    }
```

Описание: Используем идею бинарного поиска и проверяем можно ли расставить коров с расстоянием mid и более, сдвигаем границы в зависимости от результата.
Сложность по времени: O(n*log(n))
Сложность по памяти: O(n)

2.F

```
    #include <ios>
    #include <iostream>
    #include <vector>

    bool compare(const std::string& a, const
    std::string& b){
        return a + b > b + a;
    }

    void solve() {
        std::vector<std::string> parts;
        std::string line;
        while (std::cin >> line) {
            parts.push_back(line);
            size_t size = parts.size();
            if (size > 1) {
```

```
            for (size_t i = size - 1; i > 0; i--) {
                if (compare(parts[i], parts[i-1]))
std::swap(parts[i], parts[i-1]);
            }
        }
    }
    for (const auto& part : parts) std::cout << part;
}
int main() {
    std::ios_base::sync_with_stdio(false);
std::cin.tie(nullptr);
    int t = 1;
    while (t--) {
        solve();
    }
}
```

Описание: При вводе кусков числа они сразу сортируются по принципу пузырьковой сортировки, в худшем случае при добавлении нового элемента необходимо пройти весь массив каждый раз.
Сложность по времени: $O(n^2)$
Сложность по памяти: $O(n)$

3.G

```
#include <algorithm>
#include <ios>
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

bool compare(const std::pair<char, int>& a, const
std::pair<char, int>& b) {
    return a.second > b.second;
}
```

```
void solve() {
    string s;
    cin >> s;
    unordered_map<char, int> char_count;
    vector<pair<char, int>> char_weight;

    for (char c = 'a'; c <= 'z'; c++) {
        int w;
        cin >> w;
        char_weight.emplace_back(c, w);
    }
    for (char c : s) {
        char_count[c]++;
    }
    ranges::sort(char_weight.begin(),
char_weight.end(), compare);
    string edge, middle;
    for (auto item : char_weight) {
        char character = item.first;
        if (char_count[character] > 1) {
            edge.push_back(character);
            char_count[character] -= 2;
        }
        string  chars_to_add(char_count[character],
character);
        middle.append(chars_to_add);
    }
    string result;
    result.reserve(s.size());
    result.append(edge);
    result.append(middle);
    result.append(edge.rbegin(), edge.rend());

    cout << result;
}

int main() {
```

```
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);

    int t = 1;
    while (t--) {
        solve();
    }
}
```

Описание: Сортируем входные веса букв по убыванию и проходимся по ним, те из них, что в количестве больше двух раскладываем по краям, все остальное будет формироваться в середине, неважно в каком порядке.
Сложность по времени: O(n)
Сложность по памяти: O(n)

4.H

```
#include <algorithm>
#include <ios>
#include <iostream>
#include <numeric>
#include <vector>

using namespace std;
void solve() {
    int n, k;
    cin >> n >> k;
    vector<int> prices(n);
    for (int i = 0; i < n; i++) {
        cin >> prices[i];
    }
    ranges::sort(prices, greater{});
    int sum = accumulate(prices.begin(),
prices.end(), 0);
    int k_counter = 0;
    for (const int price : prices) {
        k_counter++;
        if (k_counter == k) {
```

```
            sum -= price;
            k_counter = 0;
        }
    }
    cout << sum;
}
int main() {
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(nullptr);
    int t = 1;
    while (t--) {
        solve();
    }
}
```

Описание: Больше всего извлечем выгоды, если поделим чек на более мелкие, состоящие из K продуктов, таким образом сможем взять самые дорогие бесплатно, если чеки будут состоять больше чем из K элементов, то согласно условиям, бесплатно заберем более дешевые продукты.

Сложность по времени: O(n*log(n))

Сложность по памяти: O(n)