Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Базовые задачи / Timus

Выполнил:

Студент группы P32121

Гиниятуллин Арслан Рафаилович

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задачи с **Яндекс Контеста**, handle = **giniiatullinarslan**

Задача E: Коровы в стойла

```cpp
bool check(std::vector<int> &cows, int m, int k) {
    int cow_cnt = 1;
    int cur_cow = cows[0];
    for (auto c: cows) {
        if (c - cur_cow >= m) {
            cow_cnt++;
            cur_cow = c;
        }
    }
    return cow_cnt >= k;
}

void solve() {
    int n, k;
    std::cin >> n >> k;
    std::vector<int> cows(n);
    for (auto &i: cows) std::cin >> i;
    int l = 0, r = cows.back() - cows.front() + 1;
    while (r - l > 1) {
        int m = l + (r - l) / 2;
        if (check(cows, m, k)) {
            l = m;
        } else {
            r = m;
        }
    }
    std::cout << l << '\n';
}
```

Описание решения:

Напишем бинпоиск по ответу. Будем перебирать минимальное расстояние между стойлами (натуральное число). Для каждого из расстояний будем проверять, помещается ли k коров на прямой. Вследствие нестрогой монотонности функции зависимости количества коров от минимального расстояния между стойлами можем делать переход, сужая границы поиска.

Асимптотика: O(nlogn). O(log(n)) операций для поиска нужного стойла, O(n) – проверка возможности расстановки коров за линию

Задача F: Число

```
void solve() {
    std::vector<std::string> a;
    std::string s;
    while (std::cin >> s) {
        a.push_back(s);
    }
    sort(a.begin(), a.end(), [](const std::string &s1, const std::string &s2) {
return s1 + s2 > s2 + s1; });
    for (auto &i: a) std::cout << i;
}
```

Описание решения:

Хотим жадно отсортировать строки: чтобы конкатенация предыдущей со следующей была лексикографически максимальна. Соответственное выражение напишем в компараторе. (s1, s2) -> s1 + s2 > s2 + s1;

Асимптотика: O(nlogn), сортировка с компаратором

Задача G: Кошмар в замке

```
void solve() {
    std::string s;
    std::cin >> s;
    std::size_t len = s.size();
    std::vector<std::pair<char, int>> p;
    int mp[26] = {0};
    for (auto &i: s) {
        mp[i -'a']++;
    }
    int weight;
    for (char c = 'a'; c <= 'z'; ++c) {
        std::cin >> weight;
        p.emplace_back(c, weight);
    }

    sort(p.begin(), p.end(),
         [](const std::pair<char, int> &first,
            const std::pair<char, int> &second) {
             return second.second < first.second;
         }
    );
    std::size_t l = 0, r = len - 1;
    char result[len];
    for (auto &[c, i]: p) {
        if (mp[c - 'a'] > 1) {
            result[l++] = c;
            result[r--] = c;
            mp[c - 'a'] -= 2;
        }
    }
```

```
    for (auto &[c, i]: p) {
        while (mp[c-'a']--) {
            result[l++] = c;
        }
    }
    for (auto &i: result)
        std::cout << i;
}
```

Описание решения:

Запишем в вектор пары <буква, вес> и жадно отсортируем по убыванию веса. Также посчитаем встречаемость каждой буквы алфавита в массиве. Соответственно, будем жадно расставлять буквы: сначала с большим весом и количеством, не меньшим двойки, чтобы увеличить вес строки(расстояние между буквами * вес буквы), после докинем оставшиеся.

Асимптотика: O(nlogn), сортировка с компаратором + проход по вектору за линию.

Задача H: Магазин

```
oid solve() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    long long result = 0;
    for (auto &i: a)  {
        cin >> i;
        result += i;
    }
    sort(a.begin(), a.end());
    for (int i = n - k; i >= 0; i -= k) {
        result -= a[i];
    }
    cout << result << '\n';
}
```

Описание решения:
Отсортируем массив и с конца будем жадно брать каждый k-й товар (в чеке ровно k товаров, кроме мб последнего).
Очевидно, что, если в чеке будет менее k товаров, то скидка не сработает, а более – нам только ухудшит ситуацию: кол-во чеков -> 1

Асимптотика: O(nlogn) – сортировка + линия