Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P32302

Терновский И.Е.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задача №8 «I. Машинки»

```
#include <iostream>
#include <vector>
#include <map>
#include <queue>
#include <set>

using namespace std;

int main() {
    int n, k, p;
    int ans = 0;
    cin >> n >> k >> p;
    int cars[p];

    bool on_shelf[n];

    vector<queue<int>> used_cars(n);

    set<int> floor_queue;

    for (int i = 0; i < p; i++) {
        on_shelf[i] = true;
    }

    for (int i = 0; i < p; i++) {
        cin >> cars[i];

        used_cars[cars[i] - 1].emplace(i);
    }
    int cars_on_floor = -1;
    for (int i = 0; i < p; i++) {
       int car = cars[i] - 1;
        if (on_shelf[car]) {
            ans++;
            cars_on_floor++;
            if (cars_on_floor >= k) {
```

```
            int remove_car = *floor_queue.rbegin();
            // и тут костыль =(
            if (remove_car >= 999999){
                on_shelf[remove_car - 999999] = true;
            }
            else{
                on_shelf[cars[remove_car] - 1] = true;
            }
            floor_queue.erase(*floor_queue.rbegin());
        }
        on_shelf[car] = false;
    }
    int a = used_cars[car].front();
    used_cars[car].pop();
    if (used_cars[car].empty()) {
        // костыль что бы сохранять уникальность машинки =(
        floor_queue.insert(999999 + car);
    } else {
        floor_queue.insert(used_cars[car].front());
    }
  }

  cout << ans << "\n";
  return 0;
}
```

Пояснение к примененному алгоритму:

Если очень кратко, то смысл заключается в том, что мы должны убирать с пола машинку, которая пригодится самой последней. Для этого я использую set, который внутри себя сортирует значения по возрастанию, то есть я туда складываю индекс следующей машинки с таким номером, поэтому в самом вверху будет машинка, которая понадобится максимально не скоро.

Сложность O(p log(p))

Задача №9 «J. Гоблины и очереди»

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

```
using namespace std;



int main() {
    int n;
    vector<int> queue;
    cin >> n;

    char request;
    for (int i = 0; i < n; ++i){
        cin >> request;
        if (request == '-'){
            cout << queue[0] << "\n";
            queue.erase(queue.begin());
        }
        else {
            int num;
            cin >> num;
            if (request == '+'){
                queue.push_back(num);
        } else {
                int add = queue.size()%2;
                queue.insert(queue.begin() + ((queue.size())/2)+add, num);
            }
        }
    }
}
```

Пояснение к примененному алгоритму:

Не знаю, что тут объяснять, алгоритм просто выполняет то что написано по условию и все. Скорее всего можно добиться лучшей производительности если использовать очередь, но в нее нельзя делать insert, поэтому я использовал вектор как очередь.

Сложность в худшем случае O(n^2), но таких случаев крайне мало.

Задача №10 «K. Менеджер памяти-1»

Не осилил =(

Постараюсь решить, но до дедлайна не смог(

Задача №11 «L. Минимум на отрезке»

```
#include <iostream>
#include <deque>
#include <vector>

using namespace std;

int main() {
    int n, k;
    cin >> n >> k;
    vector<int> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    deque<int> deque;
    for (int i = 0; i < k; i++) {
        deque.push_back(a[i]);
        while (deque.size() > 1 && deque.back() < deque[deque.size() - 2]) {
            deque.erase(deque.end() - 2);
        }
    }

    cout << deque.front() << " ";
    for (int i = k; i < n; i++) {
        if (deque.front() == a[i - k] || deque.size() > k) {
            deque.pop_front();
        }
        deque.push_back(a[i]);
        while (deque.size() > 1 && deque.back() < deque[deque.size() - 2]) {
            deque.erase(deque.end() - 2);
        }
        cout << deque.front() << " ";
    }
    return 0;
```

}

Пояснение к примененному алгоритму:

Тут основная мысль заключается в том, что я прохожусь “окном” по массиву, при этом я выкидываю элемент если он не принадлежит этому окну. Так же я выкидываю все значения, которые больше последнего добавленного значения в окно. Таким образом сверху очереди всегда будет минимальное значение в окне.

Общая сложность вроде O(n).