Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по дисциплине “Алгоритмы и структуры данных”

Базовые задачи

Выполнила:
Шевченко Д. П.,
группа P3230

Преподаватели:
Косяков М.С.
Тараканов Д. С.

Санкт-Петербург
2024

Задача E. Коровы в стойла

```cpp
#include <iostream>


int main() {
    int count, count_cows = 0;
    std::cin >> count >> count_cows;
    int numbers[count];
    int num;
    for (int i = 0; i < count; i++) {
        std::cin >> num;
        numbers[i] = num;
    }
    int left = 0;
    int right = numbers[count - 1] + 1;
    int distance;
    while (right - left > 1) {
        int counter = 1;
        distance = (right + left) / 2;
        int prev_left = left;
        left = numbers[0];
        for (int i = 0; i < count; i++) {
            if (numbers[i] - left >= distance) {
                counter++;
                left = numbers[i];
            }
        }
        if (counter >= count_cows) {
            left = distance;
        } else {
            left = prev_left;
            right = distance;
        }
    }
    std::cout << left;
    return 0;


}
```

Описание:
Выбираем две границы, левая ноль, правая – крайняя корова. Выбираем первое значение для нашего ответа, разделив пополам отрезок. В цикле проверяем можно ли расположить коров на расстоянии больше либо равному выбранному. Если количество коров, которых можно расположить, превосходит либо равно количеству коров, которые есть, значит левую границу сдвигаем, иначе правую. И повторяем заново. Здесь применяется алгоритм бинарного поиска по ответу.
Сложность:
Ввод элементов O(n)
Цикл while O(log(n)) – бинарный поиск, итерация уменьшает размер промежутка в два раза
Цикл for – O(n)
Сложность всего алгоритма равна O(n*log(n))

Задача F. Число

```cpp
#include <string>
#include <iostream>
#include <vector>

bool max(std::basic_string<char> x, std::basic_string<char> y) {
    if (x[0] == y[0]) {
        return x + y > y + x;
    }
    return x[0] > y[0];
}

int main() {

    std::vector<std::string> numbers;
    std::string number;

    while (std::getline(std::cin, number)) {
        if (number == "\0") {
            break;
        }
        numbers.push_back(number);
    }


    std::sort(numbers.begin(), numbers.end(), max);

    for (const std::basic_string<char> &num: numbers) {
        std::cout << num;
    }

    return 0;
}
```

Нужно отсортировать по убыванию вектор с числами, сравнивая первые цифры у чисел, то есть между 23 и 123, 23 будет “больше”.
Если встретилась ситуация, что первые цифры у чисел одинаковые, то сравниваем следующую цифру, и, когда встречается две разные, выбираем то число, у которого эта цифра больше. При сравнении может возникнуть ситуация, когда у одного числа кончились цифры. Тогда, если у второго числа последующая цифра больше либо равна первой, то выбираем это число.
В коде это можно проверить, “сложив” два числа, как строки, и выбрать наибольшее. То есть два числа 33 и 331, складываем 33331 и 33133, выбираем наибольшее.
Сложность:
std::sort имеет сложность O(n*log(n)), где n – количество строк в векторе

Задача G. Кошмар в замке

```cpp
#include <iostream>
#include <map>
#include <algorithm>
#include <string.h>
#include <vector>

int main() {
    std::string str;
    std::cin >> str;
    std::string middle = str;
    int number[26];
    std::map<char, int> map;
    std::map<char, int> count;
    std::string firstPart;
    std::string back;
    std::string center;
    std::vector<char> vector;
    for (int &i: number) {
        std::cin >> i;
    }
    std::string letter;
    char letters[] = "abcdefghijklmnopqrstuvwxyz";
    for (int i = str.size() - 1; i >= 0; --i) {
        char *size = strchr(letters, str[i]);
        map[str[i]] = number[size - letters];
        count[str[i]]++;

    }
    std::sort(middle.begin(), middle.end(), [map, count](char &x, char &y) ->
bool {
        if (count.at(x) > 1 && count.at(y) > 1) {
            return map.at(x) > map.at(y);
        } else if (count.at(x) == 1 && count.at(y) == 1) {
            return map.at(x) > map.at(y);
        } else {
            return count.at(x) > count.at(y);
        }
    });
    for (char &an: middle) {
        if (!std::count(vector.begin(), vector.end(), an)) {
            vector.push_back(an);
        }
    }
    for (char &i: vector) {
        if (count[i] > 1) {
            firstPart += i;
            count[i] -= 2;
        }
    }
    back = firstPart;
    reverse(back.begin(), back.end());

    for (char &i: vector) {
        for (int j = 0; j < count[i]; j++) {
            center += i;
        }
    }

    std::cout << firstPart << center << back << std::endl;
    return 0;

}
```

Сначала каждой букве алфавита нужно присвоить введенный вес. После чего вместе с каждой буквой строки нужно хранить ее вес и то, сколько раз она встречается в строке.
Сортируем нашу строку следующим образом: если буква встречается больше одного либо один раз, то сортируем по убыванию веса, в случаях если символы имеют разное количество вхождений в строку, то они сравниваются по этому количеству по убыванию.
В вектор записывается строка, где каждый символ повторяется только 1 раз.
Далее нужно правильно расположить символы, если символ встречается больше 1 раза, то записываем его в две строки, после чего вторую строку переворачиваем, чтобы получилось симметрично. Оставшиеся символы вставляем в строку для середины.
Выводим получившиеся три строки.
Сложность:

1. Ввод строки str O(n)
2. Инициализация массива number[26] и заполнение его значений - O(1), так как размер массива фиксирован и не зависит от размера входных данных
3. Цикл, в котором заполняются значения для map и count, выполняется за O(n)
4. Сортировка строки middle с использованием std::sort занимает O(n log(n)) времени, где n - длина строки middle.
5. Создание вектора vector уникальных символов из строки middle также занимает O(n) времени, так как мы проходимся по всей строке middle.
6. Вывод строки str в консоль - O(N), где N - длина строки.

Суммарная алгоритмическая сложность составляет O(nlog(n))

Задача H. Магазин

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

int main() {
    int count, element = 0;
    std::cin >> count >> element;
    std::vector<int> numbers;
    int num;
    for (int i = 0; i < count; i++) {
        std::cin >> num;
        numbers.push_back(num);
    }
    std::cout << numbers.size();
    std::sort(numbers.begin(), numbers.end(), std::greater<>());
    std::cout << numbers.size();

    int i = 1;
    int s = 1;
    int result = 0;
    while (i <= numbers.size()) {
        if (i == s * element) {
            std::cout << numbers[i - 1];
            numbers[i - 1] = 0;
            s++;
        }
        result += numbers[i - 1];
        i++;
    }
    std::cout << result;
    return 0;
}
```

Нужно расположить вектор по убыванию, после этого удалять каждый k-й товар, и сложить оставшиеся товары.
Сложность: ввод и сохранение элементов в векторе O(n), сортировка с помощью std::sort O(n*log(n)), цикл while проходится по вектору n раз O(n). Получается сложность O(n*log(n))