Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №1

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P3233

Фамилия И.О.

Шикунов Максим Евгеньевич

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Компилятор везде: Clang 17.0.1 C++ 20

Задача №E «Коровы в стойла»

Код:

```cpp
#include <iostream>
using namespace std;

int main() {
    long n, k, left, right, distance, cow;
    long long allDistance, previousCow, nowCow;
    cin >> n >> k;
    long long cows[n];
    for (long i {0}; i < n; i++) {
        cin >> cows[i];
    }
    left  = 0;
    right = cows[n - 1];
    while (left < right - 1) {
        distance = (right + left) / 2;
        cow = 1;
        allDistance = 0;
        previousCow = cows[0];
        for (long i {1}; i < n; i++) {
            nowCow = cows[i];
            allDistance += nowCow - previousCow;
            if (allDistance >= distance) {
                cow++;
                allDistance = 0;
            }
            previousCow = nowCow;
        }
        if (cow < k) {
            right = distance;
        } else {
            left = distance;
        }
    }
    cout << right - 1;
    return 0;
}
```

Пояснение к примененному алгоритму:

Применен метод бин поиск по ответу. Принцип таков, что мы сначала берем почти что невозможный результат и проверяем, наши входные данные подходят под него или нет, если нет то сужаем его, а если подходит то берем его и заканчиваем программу.

Сложность: $O(n)$

Задача №F «Число»

Код:

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

using namespace std;

int main() {
    string input;
    vector<string> myVector;
    while(cin >> input) {
        myVector.push_back(input);
    }
    size_t size = myVector.size();
    for (size_t i {0}; i + 1 < size; i++) {
        for (size_t j {0}; j < size - i - 1; j++) {
            if (myVector[j] + myVector[j + 1] > myVector[j + 1] +
myVector[j]) {
                swap(myVector[j], myVector[j + 1]);
            }
        }
    }
    string output;
    for (size_t i {size - 1}; i < size; i--) {
        output += myVector[i];
    }
    cout << output;
    return 0;
}
```

Пояснение к примененному алгоритму:

Принимаем числа, а потом применяем метод сортировки пузырьком, который нам расставит числа по возрастанию. Дальше выводим числа, от наибольшего до наименьшего вместе.

Сложность: $O(n^2)$

Задача №G «Кошмар в замке»

Код:

```cpp
#include <iostream>
#include <map>
using namespace std;

int main() {
    map<char, long> alphabetCount;
    multimap <long, char, greater<>> alphabetPrice;
    long price, count {0}, begin, end;
    string input, output;
    cin >> input;
    for (char letter = 'a'; letter <= 'z'; letter++) {
        alphabetCount[letter] = 0;
        cin >> price;
        alphabetPrice.insert({price, letter});
    }
    for (char letter : input) {
        count++;
        alphabetCount[letter]++;
    }
    begin = 0;
    end = count;
    output.assign(count, ' ');
    for (pair<long, char> pair : alphabetPrice) {
        if (alphabetCount[pair.second] >= 2) {
            output[begin] = pair.second;
            output[end - 1] = pair.second;
            begin++;
            end--;
            alphabetCount[pair.second] -= 2;
        }
    }
    for (pair<char, long> pair : alphabetCount) {
        if (pair.second > 0) {
            for (long i = 0; i < pair.second; i++) {
                output[end - 1] = pair.first;
                end--;
            }
        }
    }
    cout << output;
    return 0;
}
```

Пояснение к примененному алгоритму:

Метод таков, что мы сначала считаем, сколько раз встречается каждое буква, которое есть во входной строке. Дальше все буквы, которые встречаются как минимум два раза, выписываем по парно в начало и в конец (выписываем так, что у следующей пары буквы стоимость была меньше), а те которые после этого остались вставляем в середину строки.

Сложность: $O(n)$

Задача №H «Магазин»

Код:

```cpp
#include <iostream>
#include <set>
using namespace std;

int main() {
    long n;
    int k;
    int number;
    long long price {0};
    cin >> n >> k;
    multiset<int> prices;
    for (long i {0}; i < n; i++) {
        cin >> number;
        price += number;
        prices.insert(number);
    }
    long i {0};
    for (auto pr : prices) {
        if ((n - i) % k == 0) {
            price -= pr;
        }
        i++;
    }
    cout << price;
    return 0;
}
```

Пояснение к примененному алгоритму:

Используем структуру multiset, чтобы числа были в порядке возрастания, а также, чтобы числа могли повторяться. В конце берем каждое цену каждого k-ого продукта и вычитаем его из общего чека. Тем самым найдем самую минимальную сумму, которую мы потратим на данный входные данные.

Сложность: $O(n * \log(n))$

Задача №1322 «Шпион»

Код:

```
#include <iostream>
#include <set>
using namespace std;

int main() {
    int n;
    string s;
    cin >> n >> s;
    multiset<pair<char, int>> chars;
    int index = 0, i = 0, count = 0;
    for (char ch : s) {
        chars.insert(pair(ch, index));
        index++;
        count++;
    }
    pair<char, long> arr[count];
    for (pair<char, long> p : chars) {
        arr[i] = p;
        i++;
    }
    index = n - 1;
    for (int j = 0; j < s.size(); j++) {
        cout << arr[index].first;
        index = arr[index].second;
    }
    cout << endl;
    return 0;
}
```

Пояснения к примененному алгоритму:

Принимаем числа и записываем их в пары с индексом, в порядке котором они были поданы на вход. Дальше сортируем их по порядку, в котором они находятся в первой позиции (то есть у наших слов первые буквы будут все в порядке возрастания). Первым число выводим букву по индексу, который нам дали на вход, а дальше у этого числа берем изначальный индекс (то есть номер слова в котором эта буква находится в конце слова) и выводим первую букву этого слова. Повторяем это пока не выведем строку нашей длины.

Сложность: $O(n \log n)$

Задача №1726 «Кто ходит в гости…»

Код:

```
#include <iostream>
#include <set>
using namespace std;


int main() {
    size_t n, sumi = 0, count;
    cin >> n;
    count = n * (n - 1);
    multiset<long> coordX, coordY;
    long arrX[n], arrY[n];
    long x, y;
    for (size_t i = 0; i < n; i++) {
        cin >> x >> y;
        coordX.insert(x);
        coordY.insert(y);
    }
    long countNum = 0;
    for (long i : coordX) {
        arrX[countNum] = i;
        countNum++;
    }
    countNum = 0;
    for (long i : coordY) {
        arrY[countNum] = i;
        countNum++;
    }
    for (long i = n - 1; i > 0; i--) {
        sumi += (arrX[i] - arrX[i - 1] + arrY[i] - arrY[i - 1]) * (n - i) * i
* 2;
    }
    cout << sumi / count;
    return 0;
}
```

Пояснение к алгоритму:

В данной задаче мы сортируем наши координаты X и Y для того, чтобы переделать координаты входных точке так, чтобы они лежали в одном направлении. Это нам поможет, потому что теперь чтобы дойти от первой точки до третей, мы точно будет проходить через вторую и так далее (если будем идти от 1 до n, то пройдем 2, 3, …, n-1 точки). Тем самым, зная сколько у нас точек, мы можем посчитать, сколько каждый промежуток студенты буду проходить и тем самым посчитать окончательную сумму.

Сложность: $O(n \log n)$