Университет ИТМО

Факультет программной инженерии и компьютерной техники

**Лабораторная работа №2**

по «Алгоритмам и структурам данных»

Базовые задачи

Выполнил:

Студент группы P32092

Голиков А.С.

Преподаватели:

Косяков М.С.

Тараканов Д.С.

Санкт-Петербург

2023

Задача E «Коровы в стойла»

Пояснение к примененному алгоритму:

Бин поиск по ответу. Проверка - ставим на минимальном расстоянии >= заданное d, если все поставилосб - победа

Бин поиск работает, так как наша последователность проверок монотонная невозрастающая функция (до максимального d имеем true, потом всегда false просто из определения расстояния)

Асимптотика - O(nlogn)

Код решения:

#include <iostream>  
#include <vector>  
#include <algorithm>  
  
using namespace std;  
  
bool is\_ok(int dist, const vector<int> &coords, int count, size\_t n) {  
 count -= 1;  
 int last = coords[0];  
 for (size\_t i = 1; i < n; ++i) {  
 if (coords[i] - last >= dist) {  
 last = coords[i];  
 count -= 1;  
 if (count <= 0) {  
 break;  
 }  
 }  
 }  
 return count <= 0;  
}  
  
int bin\_search(int start, int end, const vector<int> &coords, int k, size\_t n) {  
 if (is\_ok(end, coords, k, n)) {  
 return end;  
 }  
 if (end - start > 1) {  
 int mid = (start + end) / 2;  
 if (is\_ok(mid, coords, k, n)) {  
 return bin\_search(mid, end, coords, k, n);  
 } else {  
 return bin\_search(start, mid, coords, k, n);  
 }  
 } else {  
 return start;  
 }

}  
  
int main() {  
 int n, k;  
 cin >> n >> k;  
 vector<int> vec(n);  
 for (size\_t i = 0; i < n; ++i) {  
 cin >> vec[i];  
 }  
 int size = (vec[n - 1] - vec[0]) / (k - 1);  
 cout << bin\_search(1, size, vec, k, n);  
 return 0;  
}

Задача F «Число»

Пояснение к примененному алгоритму:

Сортировка по очевидному компаратору - так как длины a+b и b+a равны, сравнение даст корректный результат.

Асимптотика - O(n)

Код решения:

#include <iostream>  
#include <vector>  
#include <algorithm>  
  
using namespace std;  
  
int main() {  
 string s;  
 vector<string> input;  
 int i = 0;  
 while (cin >> s) {  
 input.push\_back(s);  
 }  
  
 sort(input.begin(), input.end(), [](const auto &lhs, const auto &rhs) {  
 return lhs + rhs > rhs + lhs;  
 });  
  
 for (const auto &p: input) {  
 cout << p;  
 }  
 return 0;  
}

Задача G «Кошмар в замке»

Пояснение к примененному алгоритму:

Необходимо взять все парные буквы, сортануть их в порядке весов и вложить пары по убыванию этого порядка, то есть

aabbccdef => c b a def a b c (если веса пропорциональны алфавитному порядку).

Предположим, что это не так, и есть какая-то буква, стоящая не на своем месте, например есть ab\*\*\*ba и мы хотим переставить вторую(можно и первую, но мне справа удобнее) b

Влево двигать b не имеет смысла, так как тогда макс расстояние для нее уменьшается, а для а не меняется. Остается вправо. Так как веса букв различны, a-b >=1, итого при сдвиге буквы b вправо расстояния увеличатся и уменьшатся на 1 соответствоенно, веса не изменятся, то есть мы получим разницу сумм как a-b, что нам не нравится, так как мы хотим все максимизировать

Асимптотика - O(len(input\_string))

Код решения:

#include <iostream>  
#include <cmath>  
#include <vector>  
#include <algorithm>  
  
using namespace std;  
  
int main() {  
 string input;  
 cin >> input;  
  
 vector<int> weights(26);  
 for (size\_t i = 0; i < 26; ++i) {  
 cin >> weights[i];  
 }  
  
  
 string doubles;  
 string trash;  
 vector<int> counts(26);  
 for (char c: input) {  
 ++counts[c - 'a'];  
 if (counts[c - 'a'] == 2) {  
 doubles.push\_back(c);  
 }  
 if (counts[c - 'a'] > 2) {  
 trash.push\_back(c);  
 }  
 }  
  
 for (int i = 0; i < 26; ++i) {  
 if (counts[i] == 1) {  
 trash.push\_back((char) ('a' + i));  
 }  
 }  
  
 std::sort(doubles.begin(), doubles.end(), [&weights](char a, char b) {  
 return weights[a - 'a'] > weights[b - 'a'];  
 });  
  
 cout << doubles + trash;  
 std::reverse(doubles.begin(), doubles.end());  
 cout << doubles;  
 return 0;  
}

Задача H «Магазин»

Пояснение к примененному алгоритму:

Сортиуем цены по убыванию. Хотим сэкономить как можно больше - будем получать скидку на как можно более дорогие товары. Так как мы получаем скику на каждый k-тый, как раз каждый k-тый и получим бесплатно.

Предположим мы можем поменять какой-то пакет так, что сэкономим еще больше. Давайте поменяем, но тогда у любой позиции товара цена уменьшится, а значит и сэкономить больше мы не можем - противоречие.

Асимптотика - O(max(logn, n/k))

Код решения:

#include <iostream>  
#include <vector>  
#include <algorithm>  
  
using namespace std;  
  
int main() {  
 int n, k;  
 cin >> n >> k;  
 vector<int> costs(n);  
 int sum = 0;  
 for (int i = 0; i < n; ++i) {  
 cin >> costs[i];  
 sum += costs[i];  
 }  
 sort(costs.begin(), costs.end(), [](int a, int b) { return a > b; });  
 for (int i = 1; i <= n / k; ++i) {  
 sum -= costs[k \* i - 1];  
 }  
 cout << sum;  
  
 return 0;  
}