Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Яндекс контест

Выполнил:

Студент группы P3233

Перевозчиков И. С.

Преподаватели:

Косяков М. С.

Тараканов Д. С.

Санкт-Петербург

2022

Задача E. Коровы в стойла

```
#include <iostream>
#include <string>
#include <vector>
#include <set>
#include <map>
#include <stack>

using namespace std;

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

	int l = 0;
	int r = stalls[n - 1] - stalls[0] + 1;

	while (l < r) {
		int mid = (l + r) / 2;
		int previous_stall = stalls[0];
		int num_of_cows = 1;
		for (int i = 0; i < n; i++) {
			if (stalls[i] - previous_stall >= mid) {
				num_of_cows++;
				previous_stall = stalls[i];
			}
		}
		if (num_of_cows >= k)
			l = mid + 1;
		else
			r = mid;
	}
	cout << l - 1;
}
```

Бинарный поиск по ответу. За наименьшее возможное расстояние берем 0, за наибольшее берем путь между первым и последним стойлом. Берем их среднее арифметическое и в зависимости от результата увеличиваем или уменьшаем максимальное расстояние.

Алгоритмическая сложность: $O(n \cdot logn)$.

Задача F. Число

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

string max_string(string s1, string s2) {
	if (s1 == s2)
		return s1;

	if (s1.size() < s2.size())
		swap(s1, s2);

	int size1 = s1.size();
	int size2 = s2.size();

	for (int i = 0; i < size2; i++)
		if (s1[i] > s2[i])
			return s1;
		else if (s1[i] < s2[i])
			return s2;

	string temp = s1.substr(size2, size1 - size2);

	if (temp == max_string(temp, s2))
		return s1;
	else
		return s2;
}

void bubble_sort(vector <string>& str) {
	for (int i = 0; i < str.size(); i++)
		for (int j = i; j < str.size(); j++) {
			if (str[j] == max_string(str[i], str[j])) {
				swap(str[i], str[j]);
			}
		}
}

int main()
{
	string s;
	vector <string> str;
	while (cin >> s)
		str.push_back(s);
	bubble_sort(str);
	for (auto c : str)
		cout << c;
}
```

Если строки одинаковой длины, то сравниваем их как обычные числа, иначе, при каждом сравнении строку один делаем большей по длине и сравниваем строку 2 с подстрокой строки один такой же длины, если равны, то вызываем функцию снова. В итоге получим

Алгоритмическая сложность: $O(l \cdot n^2)$, где n – количество поданных строк, l – длины строк.

Задача G. Кошмар в замке

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
	string s;
	cin >> s;
	vector <int> weight(26);
	for (int i = 0; i < 26; i++)
		cin >> weight[i];

	vector <int> freq(26, 0);
	for (int i = 0; i < s.size(); i++)
		freq[s[i] - 'a']++;


	multimap <int, char> letters;

	for (int i = 0; i < 26; i++)
		if (freq[i] > 1)
			letters.emplace(weight[i], char('a' + i));

	s = "";

	for (auto letter : letters) {
		s += letter.second;
		int i = letter.second - 'a';
		freq[i] -= 2;
	}

	string end = s;
	reverse(s.begin(), s.end());
	for (int i = 0; i < 26; i++)
		for (int j = 0; j < freq[i]; j++)
			s += char('a' + i);

	s += end;
	cout << s;
}
```

Всегда выгоднее поставить две буквы с наибольшим весом в начало и конец, а остальные такие же буквы в центр строки, в следующую очередь ставим буквы со вторым по счету наибольшим весом. Если буква встречается в строке только один раз, то ее нужно поставить в центр, так как она не приносит веса.

Алгоритмическая сложность: $O(l)$, где l – длина строки.

Задача H. Магазин

```
#include <iostream>
#include <map>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

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

	sort(prices.rbegin(), prices.rend());

	for (int i = 0; i < n / k; i++) {
		total_price -= prices[k - 1 + k * i];
	}
	cout << total_price;
}
```

Бесплатно достанутся в любом случае не больше n/k товаров. Сортируем товары в убывающем порядке и убираем каждый k-й. Таким образом мы уберем самый дорогой из самых дешевых товаров каждый раз.

Алгоритмическая сложность: $O(n \cdot logn)$, так как применяем сортировку.