Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа №2

по «Алгоритмам и структурам данных»

Timus

Выполнил:

Студент группы P3233

Перевозчиков И. С.

Преподаватели:

Косяков М. С.

Тараканов Д. С.

Санкт-Петербург

2022

Задача 1322. Шпион

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

using namespace std;

int main()
{
	int n;
	cin >> n;
	string last;
	cin >> last;
	string first = last;
	sort(first.begin(), first.end());
	first = ' ' + first;
	vector <int> permutation(last.size());
	int j = 0;
	for (int i = 1; i <= last.size(); i++) {
		if (first[i] != first[i - 1] || j == last.size()) {
			j = 0;
		}
		while (first[i] != last[j]) {
			j++;
		}
		permutation[i - 1] = j++;
	}

	n--;

	for (int i = 0; i < last.size(); i++) {
		cout << last[permutation[n]];
		n = permutation[n];
	}

}
```

Сортируем заданную строку. Получим первый столбец таблицы. Заведем массив, в который будем записывать номер строки в отсортированной таблице, под которым идет следующая по счету циклическая перестановка. Мы можем его найти, если будем сравнивать первую и последнюю буквы, а они нам известны.

Алгоритмическая сложность: $O(n^2)$.

Задача 1604. В стране дураков

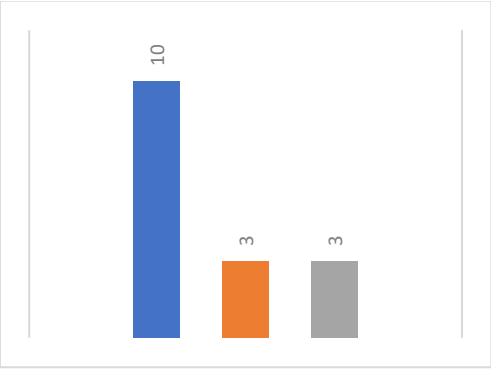
```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;


void align(int k, vector <pair<int, int>>& road_signs, vector <int>& road) {
	if (k == 0) {
		return;
	}

	int i = 1;

	while (road_signs[0].first > road_signs[1].first && road_signs[1].first > 0) {
		if (i == k || road_signs[i].first == 0) {
			i = 1;
		}
		road.push_back(road_signs[0].second + 1);
		road.push_back(road_signs[i].second + 1);
		road_signs[0].first--;
		road_signs[i].first--;
		i++;
		sort(road_signs.rbegin(), road_signs.rend());
	}
}

void place_signs(int k, vector <pair<int, int>>& road_signs, vector <int>& road) {
	int index = k;

	for (int i = 1; i < k; i++) {
		if (road_signs[i].first != road_signs[0].first) {
			index = i;
			break;
		}
	}

	while (road_signs[0].first > 0) {

		if (index == k) {
			break;
		}

		int size = road_signs[0].first - road_signs[index].first;
		for (int i = 0; i < size; i++) {
			for (int j = 0; j < index; j++) {
				road.push_back(road_signs[j].second + 1);
				road_signs[j].first--;
			}
		}

		index++;
	}

	if (road_signs[0].first > 0) {
		int size = road_signs[0].first;
		for (int i = 0; i < size; i++) {
			for (int j = 0; j < k; j++) {
				road.push_back(road_signs[j].second + 1);
				road_signs[j].first--;
			}
		}
```

```
	}
}

void left(int k, vector <pair<int, int>>& road_signs, vector <int>& road) {
	for (int i = 0; i < road_signs[0].first; i++) {
		road.push_back(road_signs[0].second + 1);
	}
	road_signs[0].first = 0;
}

int main()
{
	int k;
	cin >> k;
	vector <pair<int, int>> road_signs;
	vector <int> road;
	int num_of_signs = 0;
	for (int i = 0; i < k; i++) {
		int x;
		cin >> x;
		road_signs.push_back(make_pair(x, i));
		num_of_signs += x;
	}

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

	align(k, road_signs, road);

	if (num_of_signs - road_signs[0].first >= road_signs[0].first) {
		place_signs(k, road_signs, road);
	}
	else {
		left(k, road_signs, road);
	}

	for (int i = 0; i < road.size(); i++) {
		cout << road[i] << ' ';
	}

	return 0;
}
```

Сортируем исходные данные по убыванию. После этого их можно представить в виде столбцов.
Пример: исходные данные:

3

3 10 3



Далее пытаемся выровнять первый и второй столбцы. Для этого убираем первый знаки в следующем порядке: 1, 2, 1, 3, 1, 4 … Если в какой-то момент первый и второй столбцы равна по количеству оставшихся знаков, ставим их в следующем порядке: 1, 2, 1, 2, 1, 2 до тех пор, пока они не станут равна по количеству третьему столбцу. Продолжаем процедуру с такой же логикой до конца. В итоге мы запишем все знаки. Если не удалось приравнять по количеству знаков первый и второй столбец, то все знаки, кроме знаков из первого столбца, уже установлены. Нужно лишь установить оставшиеся знаки.

Алгоритмическая сложность: $O(n \cdot k \cdot logk)$, где $n = \sum n_k$.