Министерство образования и науки Российской Федерации

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

ОТЧЁТ

по лабораторной работе Разделяй и властвуй (Stepic)

Студенка Кузенкова Елизавета группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

Задача 1: двоичный поиск

В первой строке даны целое число $1 \le n \le 10^5$ и массив $A[1 \dots n]$ из n различных натуральных чисел, не превышающих 10^9, в порядке возрастания, во второй — целое число $1 \le k \le 10^5$ и k натуральных чисел $b_1, \dots, b_k$, не превышающих 10^9. Для каждого i от 1 до k необходимо вывести индекс $1 \le j \le n$, для которого $A[j] = b_i$, или -1, если такого j нет.

Sample Input:

```
5 1 5 8 12 13
5 8 1 23 1 11
```

Sample Output:

```
3 1 -1 1 -1
```

Исходный код к задаче 1

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

int my_bsearch(int val, std::vector<int>& mas) {
	int l = 0;
	int r = mas.size() - 1;
	if (val > mas.at(r)) return -1;
	while (l <= r) {
		int m = l + (r - l) / 2;
		unsigned int elem = mas.at(m);
		if (elem == val) return m + 1; // Для нумерации с 1
		if (val > elem) l = m + 1;
		else r = m - 1;
	}
	return -1;
}

int main()
{
	int num_count = 0;
	std::cin >> num_count;
	std::vector<int> mas;
	while (num_count-- > 0) {
		int val = 0;
		std::cin >> val;
		mas.push_back(val);
	}
	std::cin >> num_count;
	std::vector<int> nums_to_seach;
	while (num_count-- > 0) {
		int val = 0;
		std::cin >> val;
		nums_to_seach.push_back(val);
	}
	for (auto val : nums_to_seach) {
		std::cout << my_bsearch(val, mas) << " ";
	}
	std::cout << std::endl;
	std::cout.flush();
	return 0;
}
```

Задача 2: число инверсий

Первая строка содержит число $1 \le n \le 10^5$, вторая — массив $A[1 \dots n]$, содержащий натуральные числа, не превосходящие 10^9. Необходимо посчитать число пар индексов $1 \le i < j \le n$, для которых $A[i] > A[j]$. (Такая пара элементов называется инверсией массива. Количество инверсий в массиве является в некотором смысле его мерой неупорядоченности: например, в упорядоченном по неубыванию массиве инверсий нет вообще, а в массиве, упорядоченном по убыванию, инверсию образуют каждые два элемента.)

Sample Input:

```
5
2 3 9 2 9
```

Sample Output:

```
2
```

Исходный код к задаче 2

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

std::vector<unsigned int> merge(std::vector<unsigned int> a1, std::vector<unsigned int> a2,
long& count) {
	std::vector<unsigned int> result;
	auto p1 = a1.begin();
	auto p2 = a2.begin();
	bool isP2 = false, isP1 = false;
	while (true) {
		if (*p1 <= *p2) {
			result.push_back(*p1);
			p1++;
			if (p1 == a1.end()) {
				isP1 = true;
				break;
			}
		}
		else {
			size_t c = (a1.end() - p1
			count += c;
			result.push_back(*p2);
			p2++;
			if (p2 == a2.end()) {
				isP2 = true;
				break;
			}
		}
	}
	if (isP1) result.insert(result.end(), p2, a2.end());
	else result.insert(result.end(), p1, a1.end());
	return result;
}

std::vector<unsigned int> merge_sort(std::vector<unsigned int>& a, size_t l, size_t r, long&
count) {
	if (r == l) {
		std::vector<unsigned int> vec;
		vec.push_back(a[r]);
		return vec; // Если границы пересеклись
	}
	size_t m = l + (r - l) / 2; // Делим пополам
	return merge(merge_sort(a, l, m, count), merge_sort(a, m + 1, r, count), count);

}
```

```
int main()
{
	unsigned int size = 0;
	std::vector<unsigned int> mas;
	std::cin >> size;
	while (size-- >= 1) {
		unsigned int val = 0;
		std::cin >> val;
		mas.push_back(val);
	}
	long count = 0;
	auto a = merge_sort(mas, 0, mas.size() - 1, count);
	std::cout << count << std::endl;
	return 0;
}
```

Задача 3: точки и отрезки

В первой строке задано два целых числа $1 \le n \le 50000$ и $1 \le m \le 50000$ — количество отрезков и точек на прямой, соответственно. Следующие n строк содержат по два целых числа a_i и b_i ($a_i \le b_i$) — координаты концов отрезков. Последняя строка содержит m целых чисел — координаты точек. Все координаты не превышают 10^8 по модулю. Точка считается принадлежащей отрезку, если она находится внутри него или на границе. Для каждой точки в порядке появления во вводе выведите, скольким отрезкам она принадлежит.

Sample Input:

```
2 3
0 5
7 10
1 6 11
```

Sample Output:

```
1 0 0
```

Исходный код к задаче 3

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


int main()
{
	// Количество отрезков и точек
	unsigned int num_lines = 0, num_points = 0;
	// масиив отрезков
	std::vector<std::pair<int, int>> lines;
	// массив точек
	std::vector<std::pair<int, int>> points;
	std::cin >> num_lines >> num_points;
	while (num_lines-- >= 1) {
		int a = 0, b = 0;
		std::cin >> a >> b;
		lines.push_back(std::make_pair(a, b));
	}
	int counter = 0;
	while (counter < num_points) {
		int pt = 0;
		std::cin >> pt;
		points.push_back(std::make_pair(counter, pt));
		++counter;
	}
```

```
	// Сортируем массив точек
	std::stable_sort(points.begin(), points.end(), [](const std::pair<int, int> &e1, const
std::pair<int, int> &e2) {return e1.second < e2.second;});
	// Сортируем массив отрезков по левому краю
	std::stable_sort(lines.begin(), lines.end(), [](const std::pair<int, int> &e1, const
std::pair<int, int> &e2) {return e1.first < e2.first;});
	// Создаем новый массив из отрезков и упорядочиваем его по правому краю
	std::vector<std::pair<int, int>> part;
	part.insert(part.begin(), lines.begin(), lines.end());
	std::stable_sort(part.begin(), part.end(), [](const std::pair<int, int> &e1, const
std::pair<int, int> &e2) {return e1.second < e2.second;});
	// Массив числа отрезков содержащих каждую точку
	std::vector<std::pair<int, int>> numbers;
	num_points = points.size();
	// Выбираем первую точку
	auto i_pt = points.begin();
	// Оптимизируем для одинаковыхх точек ????
	int prev_pt = (*i_pt).second;
	int prev_num = 0;
	bool is_prev = false;
	auto prev_iter_left = lines.begin();
	auto prev_iter_right = part.begin();
	while (num_points > numbers.size()) { // Пока пройдены не все точки
		if (is_prev && (*i_pt).second == prev_pt) {
			numbers.push_back(std::make_pair((*i_pt).first, prev_num));
			i_pt++;
			continue;
		}
		// Берем первую точку и находим позицию первого отрезка левый край которого
более точки
		auto pos_left_line = std::find_first_of(prev_iter_left, lines.end(), i_pt, i_pt
+ 1, [](const std::pair<int, int> &e1, const std::pair<int, int> &e2) {return e1.first >
e2.second;});
		if (pos_left_line == lines.begin()) { // Начала всех отрезков правее точки -
пересечений нет
			numbers.push_back(std::make_pair((*i_pt).first, 0));
			is_prev = true;
			prev_pt = (*i_pt).second;
			prev_num = 0;
			i_pt++;
			continue;
		}
		prev_iter_left = lines.begin() == pos_left_line ? pos_left_line : pos_left_line
- 1;
		// Находим позицию первого отрезка правый край которого более либо равен точки
		auto pos_right_line = std::find_first_of(prev_iter_right, part.end(), i_pt,
i_pt + 1, [](const std::pair<int, int> &e1, const std::pair<int, int> &e2) {return e1.second
>= e2.second;});
		prev_iter_right = part.begin() == pos_right_line ? pos_right_line :
pos_right_line - 1;
		// Сохраняем найденное число отрезков
		numbers.push_back(std::make_pair((*i_pt).first, (pos_left_line - lines.begin())
- (pos_right_line - part.begin())));
		is_prev = true;
		prev_pt = (*i_pt).second;
		prev_num = (pos_left_line - lines.begin()) - (pos_right_line - part.begin());
		i_pt++;
	}
	// Выводим количество найденных (оставшихся отрезков)
	std::ostringstream oss;
	// Сортируем массив точек
	std::stable_sort(numbers.begin(), numbers.end(), [](const std::pair<int, int> &e1,
const std::pair<int, int> &e2) {return e1.first < e2.first;});
	for (auto elem : numbers) { oss << elem.second << " "; }
	std::cout << oss.str() << std::endl;

	return 0;
}
```

Задача 4: сортировка подсчетом

Первая строка содержит число $1 \le n \le 10^4$, вторая — n натуральных чисел, не превышающих 10. Выведите упорядоченную по неубыванию последовательность этих чисел.

Sample Input:

```
5
2 3 9 2 9
```

Sample Output:

```
2 2 3 9 9
```

Исходный код к задаче 4

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


int main()
{
	int n = 0;
	std::cin >> n;
	std::vector<unsigned int> b(11);
	while (--n >= 0) {
		unsigned int value = 0;
		std::cin >> value;
		b[value] += 1;
	}
	std::ostringstream oss;
	for (unsigned int i = 0; i < 11; ++i)
		for (unsigned int j = 0; j < b[i]; ++j) oss << i << " ";
	std::cout << oss.str() << std::endl;
	return 0;
}
```