

Министерство науки и высшего образования  
Российской Федерации

Московский авиационный институт  
(национальный исследовательский университет)

Институт компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Журнал по ознакомительной практике

Студент: Иванов И. И.

Группа: М8О-119Б-29

Оценка:

Дата:

Подпись:

Москва, 2022

# ИНСТРУКЦИЯ

## о заполнении журнала по производственной практике

Журнал по производственной практике студентов имеет единую форму для всех видов практик.

Задание в журнал вписывается руководителем практики от института в первые три-пять дней пребывания студентов на практике в соответствии с тематикой, утверждённой на кафедре до начала практики. Журнал по производственной практике является основным документом для текущего и итогового контроля выполнения заданий, требований инструкции и программы практики.

Табель прохождения практики, задание, а также технический отчёт выполняются каждым студентом самостоятельно.

Журнал заполняется студентом непрерывно в процессе прохождения всей практики и регулярно представляется для просмотра руководителям практики. Все их замечания подлежат немедленному выполнению.

В разделе «Табель прохождения практики» ежедневно должно быть указано, на каких рабочих местах и в качестве кого работал студент. Эти записи проверяются и заверяются цеховыми руководителями практики, в том числе мастерами и бригадирами. График прохождения практики заполняется в соответствии с графиком распределения студентов по рабочим местам практики, утверждённым руководителем предприятия. В разделе «Рационализаторские предложения» должно быть приведено содержание поданных в цехе рационализаторских предложений со всеми необходимыми расчётами и эскизами. Рационализаторские предложения подаются индивидуально и коллективно.

Выполнение студентом задания по общественно-политической практике заносится в раздел «Общественно-политическая практика». Выполнение работы по оказанию практической помощи предприятию (участие в выполнении спецзаданий, работа сверхурочно и т.п.) заносится в раздел журнала «Работа в помощь предприятию» с последующим письменным подтверждением записанной работы соответствующими цеховыми руководителями. Раздел «Технический отчёт по практике» должен быть заполнен

особо тщательно. Записи необходимо делать чернилами в сжатой, но вместе с тем чёткой и ясной форме и технически грамотно. Студент обязан ежедневно подробно излагать содержание работы, выполняемой за каждый день. Содержание этого раздела должно отвечать тем конкретным требованиям, которые предъявляются к техническому отчёту заданием и программой практики. Технический отчёт должен показать умение студента критически оценивать работу данного производственного участка и отразить, в какой степени студент способен применить теоретические знания для решения конкретных производственных задач.

Иллюстративный и другие материалы, использованные студентом в других разделах журнала, в техническом отчёте не должны повторяться, следует ограничиваться лишь ссылкой на него. Участие студентов в производственно-технической конференции, выступление с докладами, рационализаторские предложения и т.п. должны заноситься на свободные страницы журнала.

**Примечание.** Синьки, кальки и другие дополнения к журналу могут быть сделаны только с разрешения администрации предприятия и должны подшиваться в конце журнала.

Руководители практики от института обязаны следить за тем, чтобы каждый цеховой руководитель практики перед уходом студентов из данного цеха в другой цех вписывал в журнал студента отзывы об их работе в цехе.

Текущий контроль работы студентов осуществляется руководителями практики от института и цеховыми руководителями практики заводов. Все замечания студентам руководители делают в письменном виде на страницах журнала, ставя при этом свою подпись и дату проверки.

Результаты защиты технического отчёта заносятся в протокол и одновременно заносятся в ведомость и зачётную книжку студента.

**Примечание.** Нумерация чистых страниц журнала проставляется каждым студентом в своём журнале до начала практики.

С инструкцией о заполнении журнала ознакомлены:

«    » \_\_\_\_\_ 2021 г.  
(дата)

Студент Иванов И. И. \_\_\_\_\_  
(подпись)

## ЗАДАНИЕ

Принять участие в тренировках и соревнованиях по олимпиадному программированию для студентов первого курса в 2021/2022 учебном году: посетить и проработать установочные лекции, решать и дорешивать конкурсные задания, принять участие в разборе. Объем практики 108 часов.

Руководитель практики от института:

«    » \_\_\_\_\_ 2021 г.  
(дата)

\_\_\_\_\_  
(подпись)

## ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

№	Дата	Название контеста	Время проведения	Место проведения	Решено задач	Дорешано задач	Подпись
1	17.09.2021	Выдача задания, Основы C++ [1]	16:30 - 21:30	МАИ	1	2	
2	01.10.2021	Библиотека C++ [3]	16:30 - 21:30	Дистанционно	1	2	
3	21.11.2021	RuCode 4.0 Div A-B Champoinship	11:00 - 16:00	Дистанционно	1	2	
4	05.12.2021	Осенняя олимпиада первого курса	11:00 - 17:00	МАИ	1	2	
5	27.01.2022	Codeforces Round #768 (Div. 2)	17:35 - 19:35	Дистанционно	1	2	
6	11.03.2022	Паросочетания в двудольном графе, потoki в транспортной сети [15]	16:30 - 21:30	Дистанционно	1	2	
7	12.07.2022	Оформление журнала. Защита практики	9:00 - 18:00	МАИ			
		Итого часов	108				

## Отзывы цеховых руководителей практики

Принято участие в  $N$  конкурсах, прослушаны установочные лекции и разборы задач, решено  $M$  и дорешано  $K$  задач конкурсов, оформлен журнал практики с электронным приложением. Задание практики выполнено. Рекомендую оценку

Тренер Инютин М. А. \_\_\_\_\_  
(подпись)

## Работа в помощь предприятию

Встречи с представителями ИТ-компаний, сотрудничающих с МАИ.

# ПРОТОКОЛ

## ЗАЩИТЫ ТЕХНИЧЕСКОГО ОТЧЁТА

ПО ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ

студентом: Ивановым Иваном Ивановичем

[illegible]

Председатель: Зайцев В. Е. \_\_\_\_\_  
Члены: Сорокин С. А. \_\_\_\_\_  
Инютин М. А. \_\_\_\_\_

Дата: 12 июля 2022 г. *Если практика на другой кафедре, то состав комиссии меняется, но представитель кафедры 806 остаётся в комиссии и выступает в качестве цехового руководителя*

# ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

## Codeforces Round #768 (Div. 2)

### Е. Раскрась середину

ограничение по времени на тест: 2 секунды  
ограничение по памяти на тест: 256 мегабайт  
ввод: стандартный ввод  
вывод: стандартный вывод

Дано  $n$  элементов, пронумерованных от 1 до  $n$ . Элемент  $i$  имеет значение  $a_i$  и цвет  $c_i$ . Изначально  $c_i = 0$  для всех  $i$ .

Можно выполнять следующую операцию:

- Выбрать три элемента  $i, j$  и  $k$  ( $1 \leq i < j < k \leq n$ ) такие, что  $c_i, c_j$  и  $c_k$  равны 0 и  $a_i = a_k$ , и затем присвоить  $c_j = 1$ .

Найдите максимальное значение  $\sum_{i=1}^n c_i$ , которое можно получить, выполнив описанную операцию некоторое (любое) количество раз.

#### Входные данные

Первая строка содержит целое число  $n$  ( $3 \leq n \leq 2 \cdot 10^5$ ) — количество элементов.

Вторая строка содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq n$ ) — где  $a_i$  равно значению  $i$ -го элемента.

#### Выходные данные

Напечатайте одно целое число — максимальное значение  $\sum_{i=1}^n c_i$ , которое можно получить, выполнив описанную операцию некоторое (любое) количество раз.

#### Примеры

<b>входные данные</b>	Скопировать
7 1 2 1 2 7 4 7	
<b>выходные данные</b>	Скопировать
2	

### Идея решения

Описать идею решения, оценить сложность, сравнить с другими возможными идеями.

### Например

Переборное решение работает  $O(n!)$ , это очень долго. Используя метод динамического программирования,  $dp_i$  — это минимальное количество белочек при условии чего-то там для  $i$  веточек. Это позволяет решить задачу за  $O(n^2)$ . Дерево отрезков с отложенными обновлениями позволяет улучшить асимптотику до  $O(n \cdot \log n)$ , так как все операции с деревом совершаются за  $O(\log n)$ .

### Исходный код

Исходный код необходимо комментировать, но не более 25% строк

```
1 #include <bits/stdc++.h>
2
3 using pii = std::pair<int, int>;
4
```

```

5 | const int INF = 1e9;
6 |
7 | void set_pos(std::vector<int> & pos, int x, int i) {
8 |     if (pos[x] == -1) {
9 |         pos[x] = i;
10 |    }
11 | }
12 |
13 | struct seg_t {
14 |     int l, r;
15 |
16 |     seg_t() {
17 |         l = 1e9;
18 |         r = 1e9;
19 |     }
20 |
21 |     seg_t(int _l, int _r) {
22 |         l = _l;
23 |         r = _r;
24 |     }
25 |
26 |     friend bool operator == (const seg_t & lhs, const seg_t & rhs) {
27 |         return lhs.l == rhs.l and lhs.r == rhs.r;
28 |     }
29 |
30 |     friend bool operator < (const seg_t & lhs, const seg_t & rhs) {
31 |         if (lhs.l != rhs.l) {
32 |             return lhs.l < rhs.l;
33 |         } else {
34 |             return lhs.r < rhs.r;
35 |         }
36 |     }
37 | };
38 |
39 | /* KTO PROCHIAL, TOT ZAKROET SESSIU! */
40 |
41 | #ifndef SEGMENT_TREE
42 | #define SEGMENT_TREE
43 |
44 | template<class T>
45 | class segment_tree_t {
46 | private:
47 |     size_t _size;
48 |     std::vector<T> _data;
49 |     std::vector<T> _delay;
50 | public:
51 |     segment_tree_t(const size_t & n) : _size(n), _data(4 * n), _delay(4 * n) {}
52 |
53 |     ~segment_tree_t() = default;
54 |
55 |     void update_delay(size_t id, size_t l, size_t r) {
56 |         if (_delay[id] == T()) {
57 |             return;
58 |         }
59 |         _data[id] = std::min(_data[id], _delay[id]);
60 |         if (id * 2 < 4 * _size) {
61 |             _delay[id * 2] = std::min(_delay[id * 2], _delay[id]);
62 |         }
63 |         if (id * 2 + 1 < 4 * _size) {
64 |             _delay[id * 2 + 1] = std::min(_delay[id * 2 + 1], _delay[id]);

```



```

65     }
66     _delay[id] = T();
67 }
68
69 T operator [] (int id) {
70     return get(1, id, id, 1, _size);
71 }
72
73 T get(size_t ql, size_t qr) {
74     return get(1, ql, qr, 1, _size);
75 }
76
77 T get(size_t id, size_t ql, size_t qr, size_t l, size_t r) {
78     update_delay(id, l, r);
79     if (ql <= l and r <= qr) {
80         return _data[id];
81     }
82     size_t m = (l + r) / 2;
83     if (qr <= m) {
84         return get(id * 2, ql, qr, l, m);
85     }
86     if (ql > m) {
87         return get(id * 2 + 1, ql, qr, m + 1, r);
88     }
89     // return get(id * 2, ql, qr, l, m) + get(id * 2 + 1, ql, qr, m + 1, r);
90 }
91
92 void delay(size_t ql, size_t qr, const T & val) {
93     if (ql <= qr) {
94         delay(1, ql, qr, 1, _size, val);
95     }
96 }
97
98 void delay(size_t id, size_t ql, size_t qr, size_t l, size_t r, const T & val) {
99     update_delay(id, l, r);
100     if (ql <= l and r <= qr) {
101         _delay[id] = val;
102         return;
103     }
104     size_t m = (l + r) / 2;
105     if (qr <= m) {
106         delay(id * 2, ql, qr, l, m, val);
107     } else if (ql > m) {
108         delay(id * 2 + 1, ql, qr, m + 1, r, val);
109     } else {
110         delay(id * 2, ql, qr, l, m, val);
111         delay(id * 2 + 1, ql, qr, m + 1, r, val);
112     }
113 }
114 };
115
116 #endif /* SEGMENT_TREE */
117
118 int main() {
119     std::ios::sync_with_stdio(false);
120     std::cin.tie(0);
121
122     int n;
123     std::cin >> n;
124     std::vector<int> a(n);




```

```

125     for (int i = 0; i < n; ++i) {
126         std::cin >> a[i];
127     }
128     std::vector<int> pos(n + 1, -1);
129
130     for (int i = 1; i <= 2; ++i) {
131         set_pos(pos, a[i - 1], i);
132     }
133     segment_tree_t<seg_t> st(n);
134     std::vector<int> dp(n + 1, 0);
135     for (int i = 3; i <= n; ++i) {
136         int num = a[i - 1];
137         int pos_num = pos[num];
138         dp[i] = dp[i - 1];
139         if (pos_num != -1) {
140             seg_t cur_cov = seg_t(pos_num, i);
141             dp[i] = std::max(dp[i], dp[pos_num - 1] + (i - pos_num - 1));
142             st.delay(pos_num + 1, i - 1, cur_cov);
143             seg_t last_covered = st[pos_num];
144             // std::cout << last_covered.l << ' ' << last_covered.r << '\n';
145             if (last_covered.l <= pos_num and pos_num <= last_covered.r) {
146                 dp[i] = std::max(dp[i], dp[last_covered.r] - 1 + (i - last_covered.r));
147                 st.delay(last_covered.r + 1, i - 1, cur_cov);
148             }
149         }
150         set_pos(pos, a[i - 1], i);
151     }
152     // for (int i = 0; i <= n; ++i) {
153     //     std::cout << "i = " << i << ", dp = " << dp[i] << std::endl;
154     // }
155     std::cout << dp.back() << std::endl;
156 }

```

## Фрагмент турнирной таблицы контеста

Положение 		Совпадений: 1  mainyutin							
№	Кто	=	*	A 500	B 1000	C 1500	D 2250	E 2250	F 3000
384	 mainyutin	3893		492 00:04	882 00:17	1206 00:49		1313 01:33	

## Выводы

Задача решена. **ИЛИ** Задача дорешана. **ИЛИ** Не принята чекером.

Ошибки, неудачи, как они преодолевались.

## Например

Задача решена. Основные события процесса отладки: неправильный ответ на претесте 3, исправил дерево отрезков.

*Если задач много (более одной на контест), то часть отчёта может быть представлена в электронном виде (на компакт диске или на плоской флешке, оглавление прилагаемого носителя должно быть распечатано рекурсивным обходом и должно однозначно интерпретироваться как контесты и задачи). На носитель следует поместить журнал в формате pdf и в виде исходного кода (в случае LaTeX должен быть takefile). Для каждого контеста следует завести отдельную директорию (например: в папку «20220630» поместить файлы «a.cpp», «b.cpp» и условия задач).*