

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»**

Журнал практики

Обучающийся Иванов Иван Иванович

Факультет №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

Учебная группа М8О-199Б-29

Направление подготовки (специальность) 01.03.00
(шифр)

«Информационные технологии и прикладная математика»
(наименование направления, специальности)

Вид практики ознакомительная
(учебной, производственной, преддипломной или другой вид практики)

Руководитель практики от МАИ

Крылов С. С. _____
(фамилия, имя, отчество) (подпись)

_____ / Иванов И. И. / 12 июля 2022 г.
(подпись обучающегося) (дата)

Москва, 2022

1 Место и сроки проведения практики

Сроки проведения практики

-дата начала практики 17.09.2021

-дата окончания практики 12.07.2022

Наименование организации:

Московский авиационный институт (национальный исследовательский университет)

Название структурного подразделения (отдел, лаборатория):

кафедра №806 «Вычислительная математика и программирование»

2 Инструктаж по технике безопасности

_____ / Крылов С. С. / 17 сентября 2021 г.
(подпись проводившего) (дата проведения)

3 Индивидуальное задание обучающегося

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 МАИ:

Крылов С. С. / _____ / 17 сентября 2021 г.

Руководитель от организации:

_____ / _____ / 17 сентября 2021 г.

_____ / Иванов И. И. / 17 сентября 2021 г.
(подпись обучающегося) (дата)

4 План выполнения индивидуального задания

№	Тема	Дата
1	Основы C++	17.09.2021
2	Основы C++	24.09.2021
3	Библиотека C++	01.10.2021
4	Библиотека C++	08.10.2021
5	Теория чисел	15.10.2021
6	Основы ДП	22.10.2021
7	Арифметика в кольце, комбинаторика, функция Эйлера	22.10.2021
8	Префиксные суммы, сортировка событий, метод двух указателей	05.11.2021
9	Двумерное ДП, задача о рюкзаке	12.11.2021
10	Геометрия, тернарный поиск	19.11.2021
11	Осенняя олимпиада первого курса	05.12.2021
12	Основы теории графов	11.02.2022
13	Кратчайшие пути во взвешенных графах	18.02.2022
14	СНМ, минимальное остовное дерево	25.02.2022
15	Деревья, наименьший общий предок	04.03.2022
16	Паросочетания в двудольном графе, потоки в транспортной сети	11.03.2022
17	Строки, Z-функция, хеши, префиксное дерево	18.03.2022
18	ДП по подмножествам, ДП по профилю	25.03.2022
19	Теория игр, функция Шпрага-Гранди	01.04.2022
20	Дерево отрезков	08.04.2022
21	Дерево отрезков с отложенными обновлениями	15.04.2022
22	Декартово дерево	22.04.2022
23	Весенняя олимпиада первого курса	15.05.2022
24	Оформление журнала с электронным приложением	11.07.2022
25	Защита практики	12.07.2022

_____ / Иванов И. И. / 17 сентября 2021 г.
(подпись обучающегося) (дата)

5 Отзыв руководителя практики от организации

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

Руководитель от организации:

_____/_____/_____/ 12 июля 2022 г.
(подпись) (фамилия, имя, отчество) М.П. (печать)

6 Отчёт обучающегося 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

входные данные	Скопировать
7 1 2 1 2 7 4 7	
выходные данные	Скопировать
2	

Идея решения

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

Например

Переборное решение работает $O(n!)$, это очень долго. Использую 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 контексты 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).