

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего
образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 2

"Задачи 1296, 1494"

по дисциплине Алгоритмы и структуры данных

Выполнила: студентка гр. **R3238**
поток **2.1**

Нечаева А. А.

Преподаватель: *Тропченко Андрей Александрович*

Санкт-Петербург, 2024

1 Цель

Разработать и реализовать алгоритмы для решения задач 1296, 1494.

2 Задача 1296

1296. Гиперпереход

Ограничение времени: 1.0 секунды
Ограничение памяти: 64 МБ

Гиперпереход, открытый ещё в начале XXI-го века, и сейчас остаётся основным способом перемещения на расстояния до сотен тысяч парсеков. Но совсем недавно физиками открыто новое явление. Оказывается, длительностью альфа-фазы перехода можно легко управлять. Корабль, находящийся в альфа-фазе перехода, накапливает гравитационный потенциал. Чем больше накопленный гравитационный потенциал корабля, тем меньше энергии потребуется ему на прыжок сквозь пространство. Ваша цель — написать программу, которая позволит кораблю за счёт выбора времени начала альфа-фазы и её длительности накопить максимальный гравитационный потенциал.

В самой грубой модели гравитационность — это последовательность целых чисел p_i . Будем считать, что если альфа-фаза началась в момент i и закончилась в момент j , то накопленный в течение альфа-фазы потенциал — это сумма всех чисел, стоящих в последовательности на местах от i до j .

Исходные данные

В первой строке записано целое число N — длина последовательности, отвечающей за гравитационность ($0 \leq N \leq 60000$). Далее идут N строк, в каждой записано целое число p_i ($-30000 \leq p_i \leq 30000$).

Результат

Выведите максимальный гравитационный потенциал, который может накопить корабль в альфа-фазе прыжка. Считается, что потенциал корабля в начальный момент времени равен нулю.

Примеры

исходные данные	результат
10 31 -41 59 26 -53 58 97 -93 -23 84	187
3 -1 -5 -6	0

Автор задачи: Ден Раскавалов
Источник задачи: IX Открытое командное соревнование школьников по программированию (13.03.2004)

Рис. 1. Условие задачи 1296.

2.1 Основная идея

Задача сводится к поиску максимальной суммы подпоследовательности последовательности p_i .

2.2 Краткое описание алгоритма

- 1. Входные данные:** целое число N – длина последовательности, отвечающей за гравитационную интенсивность ($0 \leq N \leq 60000$). Далее идут N строк, в каждой записано целое число p_i ($-3000 \leq p_i \leq 30000$).
- 2.** Считываем построчно числа и записываем их сумму в переменную cur_sum .
- 3.** На каждой итерации цикла проверяем, что текущая сумма неотрицательна, иначе объявляем ее нулевой.
- 4.** На каждой итерации проверяем, что максимальная сумма не меньше текущей, иначе присваиваем значение текущей суммы максимальной.
- 5. Выходные данные:** целое неотрицательное число – максимальный гравитационный потенциал, который накопит корабль в альфа-фазе прыжка.

2.3 Листинг

Листинг 1. Исходный код для 1296

```
1 #include <iostream>
2
3
4 int main() {
5     int n;
6     std::cin >> n;
7
8     // task to search substring with max sum
9     int max_sum = 0;
10    int cur_sum = 0;
11
12    for (int i = 0; i < n; ++i) {
13        int cur_p;
14        std::cin >> cur_p;
15
16        cur_sum += cur_p;
17        cur_sum = cur_sum < 0 ? 0 : cur_sum;
18        max_sum = max_sum < cur_sum ? cur_sum : max_sum;
19    }
20 }
```

2.4 Результат



Online Judge

[О системе](#)
[Часто задаваемые вопросы](#)
[Новости сайта](#)
[Форум](#)
[Ссылки](#)

Задачи

[Длина задачи](#)
[Отправить на проверку](#)
[Состояние проверки](#)
[Руководство](#)

Авторы

[Регистрация](#)
[Исправить данные](#)
[Рейтинг авторов](#)

Соревнования

[Текущее соревнование](#)
[Расписание](#)
[Промежуточные соревнования](#)
[Правила](#)

Результаты проверки решений

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10553534	18:25:23 9 фев 2024	mistDragon	1296_Гиперперекосы	Visual C++ 2022 x64	Accepted		0.062	608 KB

Рис. 2. Результат отправки задачи 1296.

3 Задача 1494

1494. Монобильярд

Ограничение времени: 1.0 секунды

Ограничение памяти: 64 МБ

Стол для монобильярда, установленный в игровом доме уездного города N , оказался очень прибыльным вложением. До того, как в городе появился небезызвестный господин Чичиков. Раз за разом он выигрывал, и хозяин, подсчитывая убытки, понимал, что дело тут нечисто. Однако уличить подлеца в жульничестве не удавалось до прибытия в город N ревизора из Петербурга.

Правила игры в монобильярд очень просты: нужно последовательно закатить в единственную лузу шары с номерами $1, 2, \dots, N$ (именно в этом порядке). Пока господин Чичиков играл, ревизор несколько раз подходил к столу и забирал из лузы последний закатившийся туда шар. В конце концов, оказалось, что Чичиков закатил в лузу все шары, а ревизор все шары достал и обследовал. Аферист утверждал, что закатил шары в правильном порядке. Хозяин понял, что это его шанс: ревизор должен помнить, в каком порядке он доставал шары. Однако так ли легко будет доказать жульничество?

Исходные данные

В первой строке записано целое число N — количество бильярдных шаров ($1 \leq N \leq 100000$). В следующих N строках даны номера этих шаров в том порядке, в котором ревизор забирал их из лузы.

Результат

Выведите слово «Cheater», если Чичиков не мог закатить все N шаров в правильном порядке. Иначе выведите «Not a proof».

Примеры

исходные данные	результат
2 2 1	Not a proof
3 3 1 2	Cheater

Замечания

В первом примере Чичиков мог закатить шары в правильном порядке, если ревизор достал их оба по очереди уже после того, как Чичиков закатил второй шар. Во втором примере Чичиков мог закатить шары в любом порядке, кроме правильного 1-2-3.

Автор задачи: Алексей Самсонов

Источник задачи: XIII командный чемпионат школьников Свердловской области по программированию (14 октября 2006 года)

Рис. 3. Условие задачи 1494.

3.1 Основная идея

Основная идея состоит в том, чтобы на каждом шаге итерации проверять, является текущая подпоследовательность обратной с шагом 1.

3.2 Краткое описание алгоритма

1. Входные данные: количество шаров – целое число n , такое что $(1 \leq n \leq 100000)$. В следующих n строках даны номера этих шаров в том порядке, в котором ревизор забирал их из луны.

2. Введем переменные *temp* – отвечает за максимальный встреченный номер шара, тип *int*; *cheater* – отвечает за то, уличен ли Чичиков в читерстве, тип *bool*. Так же объявим *stack_of_balls* – помогает нам проверять, корректно ли загнаны в лунку шары.

3. При вводе каждого нового номера шара, проверяем, больше ли он *temp*, если, да, тогда складываем в стек номера с $temp + 1$ до текущего номера шара. Затем обновляем значение *temp*, присваивая ему значение текущего шара; иначе проверяем равенство текущего номера шара значению вершины стека, если это условие выполнено, то убираем верхний номер шара, иначе, присваиваем переменной *cheater* значение *true*, так как получена последовательность неудовлетворяющая условиям честной игры.

5. Выходные данные: строка "*Not a proof*" – для случая честной игры, иначе – "*Cheater*" .

3.3 Структуры данных

В данной работе была использована структура данных **стек**. Стек сравним со стопкой книг, когда мы складываем книги первая оказывается в самом низу. Книги можно забирать только с вершины, то есть мы не можем вытаскивать книги из случайного места стопки. Чтобы достичь книги, которую мы положили первой, необходимо снять все книги, которые были сложены в стопку после нее. Выполняется принцип **First In – Last Out (FILO)**.

3.4 Листинг

Листинг 2. Исходный код для 1494

```
1 #include <iostream>
2 #include <stack>
3
4
5 int main() {
6     int n;
7     std::cin >> n;
8     // special data structure stack (first in – last out)
9     // helps to check order of balls
10    std::stack<int> stack_of_balls;
11    bool cheater = false;
12    // the biggest number of ball was input
13    int temp = 0;
14    for (int i = 0; i < n; ++i) {
15        int cur_ball;
16        std::cin >> cur_ball;
17        // check and update stack of the balls
18        if (temp < cur_ball) {
19            for (int j = temp + 1; j < cur_ball; ++j) {
20                stack_of_balls.push(j);
21            }
22            temp = cur_ball;
23        } else {
24            // check if current ball is on the top of stack
25            if (cur_ball == stack_of_balls.top()) {
26                // update stack by pop the highest
27                stack_of_balls.pop();
28            } else {
29                cheater = true;
30            }
31        }
32    }
33
34    if (cheater) {
35        std::cout << "Cheater" << std::endl;
36    } else {
37        std::cout << "Not a proof" << std::endl;
38    }
39 }
```

3.5 Результат

Timus Online Judge			
Online Judge	Задачи	Авторы	Соревнования
О системе	Архив задач	Регистрация	Текущее соревнование
Часто задаваемые вопросы	Отправить на проверку	Исправить данные	Расписание
Новости сайта	Состояние проверки	Рейтинг авторов	Прошедшие соревнования
Форум	Руководство	<input type="text" value="Имя автора"/> <input type="button" value="Поиск"/>	Правила
Ссылки			

Результаты проверки решений

ID	Дата	Автор	Задача	Язык	Результат проверки	№ теста	Время работы	Выделено памяти
10553903	22:42:22 9 фев 2024	mistDragon	1494. Монобильярд	Visual C++ 2022 x64	Accepted		0.093	1 212 КБ

Рис. 4. Результат отправки задачи 1494.

4 Вывод по работе

В ходе выполнения данной лабораторной работы были реализованы алгоритмы для решения задач 1296, 1494. Для решения задачи 1494 была использована особая структура данных – стек.