Кротков Павел Андреевич,
Кучеренко Демид Сергеевич,
Ульянцев Владимир Игоревич

ЗАДАЧА «СКЛАД ОБИ-ВАНА КЕНОБИ»

Этой статьей мы продолжаем цикл публикаций олимпиадных задач для школьников по информатике. Решение таких задач и изучение разборов поможет повысить уровень практических навыков программирования и подготовиться к олимпиадам по информатике.

В статье рассматривается задача «Склад Оби-Вана Кеноби», которая предлагалась на Пятой командной интернет-олимпиаде по программированию в 2012–2013 учебном году. Материалы этой олимпиады можно найти на сайте http://neerc.ifmo.ru/school/io.

УСЛОВИЕ ЗАДАЧИ

Оби-Ван Кеноби – один из последних представителей ордена рыцарей джедаев. Он родом с планеты Стьюджон, где абсолютно все жители чтут порядок, и Оби-Ван Кеноби – не исключение.

Во всех эпизодах «Звездных войн» Оби-Ван Кеноби орудует синим световым мечом, но мало кто знает, что на самом деле у него их много, причем все его мечи пронумерованы. Оби-Ван хранит все свои мечи на очень длинном столе. Когда он хочет вооружиться, он берет самый правый меч со стола и идет по своим делам. После этого он орудует взятым мечом до тех пор, пока не потеряет или не сломает его.

Иногда кто-нибудь дарит Оби-Вану новый меч, и тогда он просто кладет его на стол справа от тех, что уже лежат там. Но самое страшное случается тогда, когда к столу подходит мама Оби-Вана Кеноби. Мама тоже житель планеты Стьюджон, и поэтому она всегда возмущается тем, что мечи лежат неупорядоченно. Чтобы исправить это, она забирает всю левую половину мечей (если мечей на столе было нечетное число $2m + 1$, то мама забирает m мечей) и начинает их класть справа от оставшихся на столе мечей. Причем сначала она кладет меч, который изначально был самым левым, потом правее от него кладет второй меч и так далее.

Определите, в каком порядке лежат мечи у Оби-Вана Кеноби на данный момент. У

каждого меча есть свой личный номер, который известен только Оби-Вану и вам.

Формат входного файла

В первой строке входного файла дано натуральное число n ($1 \le n \le 10^6$) – количество изменений, произошедших на столе.

В следующих n строках заданы описания изменений в следующем формате:

– если строка начинается со слова "*add*", то в той же строке через пробел находится число x ($1 \le x \le n$) – личный номер меча, подаренного Оби-Вану Кеноби. Гарантируется, что мечей с таким номером Оби-Вану раньше не дарили и больше никогда не подарят;

– строка, содержащая единственное слово "*take*", означает, что Оби-Ван забрал самый правый меч со стола с собой. Если на столе не было мечей, то ничего не произошло;

– строка, содержащая единственное слово "*mum!*", означает, что к столу подошла мама и переложила половину мечей слева направо. Если на столе было меньше двух мечей, то ничего не произошло.

Изначально стол был пуст.

Формат выходного файла

В первой строке выходного файла выведите число k – количество мечей, лежащих на данный момент на столе у Оби-Вана Кеноби. В следующей строке выведите k чисел через пробел – личные номера мечей, лежащих на столе, в том порядке, в котором они на нем лежат (слева направо).

Примеры входных и выходных данных

kenobi.in	kenobi.out
8 add 1 add 2 add 4 add 3 add 5 add 8 take mum!	5 4 3 5 1 2

Пояснение к примеру

После выполнения шести операций *add* и одной операции *take* на столе будут лежать пять мечей в следующем порядке: 1 2 3 4 5. После чего появится мама Оби-Вана Кеноби, возьмет мечи с номерами 1 и 2 и положит их справа от оставшихся на столе трех мечей.

РАЗБОР ЗАДАЧИ

Рассмотрим два различных решения данной задачи. Первое приведенное решение будет несколько проще, однако при данном в условии максимальном количестве операций не уложится в ограничение времени работы программы. Второе, более сложное, будет отвечать всем требованиям, предъявляемым к программе, решающей эту задачу.

Наивное решение (обработка запроса *mum!* с использованием $O(n)$ операций)

Будем хранить количество всех мечей, имеющихся у Оби-Вана, в переменной *count*, а номера мечей – в массиве *nambers*. Тогда реализовать все три операции будет достаточно просто. В листинге 1 приведен код процедур на языке Pascal, выполняющих то, что написано в условии задачи.

Используемый в программе массив *tmp* необходимо, как и массив *nambers*, объявить глобально, во избежание ошибки, связанной с переполнением стека.

Заметим, что обработка каждого запроса *mum!* требует $2 \times count$ операций, поэтому при многократном вызове этой процедуры и достаточно большом количестве мечей на столе программа будет работать слишком долго и превысит ограничение на максимальное время работы программы.

Верное решение (обработка запросов с использованием $O(1)$ операций)

Чтобы написать программу, работающую за корректное время, поменяем подход к хранению информации о мечах на столе. Будем использовать следующие переменные:

count – количество мечей на столе;

first – номер самого левого меча на столе;

Листинг 1. Наивная реализация обработки запросов

```
procedure add(new : longint);
begin
  count := count + 1;
  numbers[count] := new;
end;

procedure take;
begin
  if (count > 0) then count := count - 1;
end;

procedure mum;
var
  i, mid : longint;
begin
  if (count < 2) then exit;
  mid := count div 2;
  for i := mid + 1 to count do
    tmp[i - mid] := numbers[i];
  for i := 1 to mid do
    tmp[count - mid + i] := numbers[i];
  for i := 1 to count do
    numbers[i] := tmp[i];
end;
```

last – номер самого правого меча на столе;

mid – номер самого правого меча, который мама переложит, если подойдет к столу прямо сейчас;

next [ ] – массив, *i*-ый элемент которого содержит номер меча, лежащего правее меча с номером *i*;

prev [ ] – массив, *i*-ый элемент которого содержит номер меча, лежащего левее меча с номером *i*.

Сразу оговоримся о том, что если меча номер *i* на столе в данный момент нет, то для программы не существенно то, что записано в *next* [*i*] и *prev* [*i*], а также о том, что, если соответствующего какой-то переменной меча в принципе не может быть (например, *mid* при *count* = 1 или *next* [*last*], эта переменная будет равна –1.

Теперь подробно рассмотрим то, как будет выполняться каждый запрос. Начнем с запроса на добавление нового меча. Несложно понять то, как изменятся массивы *next* и *prev* и переменная *last* после добавления одного меча справа от самого правого. Однако, важно не забыть и о том, как поменяется переменная *mid*. Из условия задачи понятно, что при четном значении переменной *count* добавление одного меча справа не меняет значение переменной *mid*, а при нечетном необходимо сделать операцию *mid* := *next* [*mid*]. В листинге 2 приведен соответствующий процедуре добавления код.

Заметим, что в приведенном коде отдельно обрабатываются случаи, когда меч добавляется на пустой стол или на стол, на котором лежит только один меч. Это сделано для соблюдения правила о –1 в переменных, для которых соответствующего им меча не существует.

Запрос на удаление самого правого меча со стола можно реализовать по аналогии с запросом на добавление меча с той разницей, что переменная *mid* будет изменяться в случае четного числа оставшихся на столе мечей, и ее новое значение будет равно *prev* [*mid*]. В листинге 3 приведен код процедуры, реализующий запрос на удаление.

Наконец, научимся обрабатывать запрос на перемещение половины мечей со стола

слева направо. Заметим, что меч с номером, записанным в переменную *mid*, теперь станет на столе самым правым, а меч, лежавший правее меча с номером *mid*, то есть *next* [*mid*], станет самым левым. При этом в середине стола окажутся те мечи, которые раньше были самым левым и самым правым. Соответствующим образом изменим значения всех переменных, выполнив запрос. Программная реализация данного запроса приведена в листинге 4.

Только что реализованная нами структура данных, на самом деле, является двусвязным списком [1] с указателем на середину. Сам по себе двусвязный список является достаточно известной и используемой структурой данных и может оказаться полезным при реализации таких структур данных, как стек, очередь или дек [2].

Листинг 2. Обработка запроса на добавление меча с номером `new`

```
procedure add(new : longint);
begin
  count := count + 1;
  if (count = 1) then begin
    first := new;
    last := new;
    prev[new] := -1;
    next[new] := -1;
  end else begin
    next[last] := new;
    prev[new] := last;
    last := new;
    next[new] := -1;
  end;
  if (count mod 2 = 0) then
    if (count = 2) then
      mid := first
    else
      mid := next[mid];
end;
```

Листинг 3. Обработка запроса на удаление со стола самого правого меча

```
procedure take;
begin
  if (count = 0) then
    exit;
  if (count = 1) then begin
    count := 0;
    first := -1;
    last := -1;
    mid := -1;
    exit;
  end;
  count := count - 1;
  last := prev[last];
  next[last] := -1;
  if (count mod 2 = 1) then
    mid := prev[mid];
end;
```

Листинг 4. Обработка запроса на перемещение половины мечей слева направо

```
procedure mum;
var
  oldLast, oldFirst : longint;
begin
  if (count < 2) then
    exit;
  oldLast := last;
  oldFirst := first;
  last := mid;
  first := next[mid];
  prev[first] := -1;
  next[last] := -1;
  next[oldLast] := oldFirst;
  prev[oldFirst] := oldLast;
  if (count mod 2 = 0) then
    mid := oldLast
  else
    mid := prev[oldLast];
end;
```

Источники:

1. *Кнут, Дональд Эрвин.* Исскуство программирования, том 1. Основные алгоритмы. 3-е изд.: Пер. с англ.: Уч. пос. М.: Издательский дом «Вильямс», 2010.

2. *Скиена С.* Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. СПб.: БХВ-Петербург, 2011.

Кротков Павел Андреевич,
студент третьего курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике,

Кучеренко Демид Сергеевич,
студент второго курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике,

Ульянцев Владимир Игоревич,
студент шестого курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике.