ЭФФЕКТИВНОЕ УДАЛЕНИЕ ЭЛЕМЕНТОВ ИЗ ОДНОМЕРНОГО МАССИВА

EFFECTIVE REMOVAL OF ELEMENTS FROM ONE-DIMENSIONAL ARRAY

A. Suvorov

Summary. This article is based on the solution of one Olympiad problem which was given to students of MGSU during the Olympiad in Informatics. The problem is related to removal of elements from the given array and determination of the last remaining number. The paper considers different extensions of this problem and its possible solutions.

Keywords: array, Fibonacci numbers, Matlab, C++.

Суворов Александр Павлович
К.т.н., старший преподаватель, Московский государственный строительный университет
suvorovap@mgsu.ru

Аннотация. В основу данной статьи легло решение одной олимпиадной задачи, которая была предложена студентам МГСУ на олимпиаде по информатике. Задача связана с удалением элементов из данного массива и определением последнего оставшегося числа. Рассматриваются различные расширения этой задачи и ее возможные решения.

Ключевые слова: массив, числа Фибоначчи, Матлаб, С++.

Рассмотрим следующую задачу. Дана последовательность (массив) натуральных чисел от 1 до N. Далее из этой последовательности чисел мы удаляем числа, номера которых являются числами Фибоначчи. Напомним, что числа Фибоначчи — это числа 0,1,2,... и каждое следующее число получается путем сложения двух предыдущих, т.е. 0,1,2,3,5,8,13,.... (Нумерацию элементов (индексацию) в данной последовательности чисел мы будем начинать с нуля, поэтому нулевой номер (индекс) имеет самое первое число в последовательности.) После удаления оставшиеся числа мы сдвигаем к началу массива. Такое удаление мы назовем удалением по Фибоначчи.

После этого мы удаляем числа с четными номерами (индексами): 0,2,4,.... Оставшиеся числа сдвигаем к началу массива.

Далее удаляем числа с нечетными номерами: 1,3,5,... и опять сдвигаем оставшиеся числа к началу массива.

Эту последовательность удалений и сдвигов (удаление по Фибоначчи, удаление чисел с четными номерами, удаление чисел с нечетными номерами) мы повторяем до тех пор, пока в последовательности не осталось одно число. Что это за число, если N = 200000?

Для решения этой задачи мы предлагаем написать программу для произвольного количества элементов N и далее взять N = 200000. Для произвольного N может получиться так, что перед удалением по Фибоначчи в последовательности осталось, например, 4 числа и тогда при удалении по Фибоначчи все числа должны быть удалены из этого массива, так как 4 числа массива имеют номера 0,1,2,3 (а это числа Фибоначчи). В этом случае ответом должно стать последнее число с номером 3. Этот факт учитывается в программе как специальный случай.

Рассмотрим пример, в котором N = 20.
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20.

После удаления по Фибоначчи остались числа:
5,7,8,10,11,12,13,15,16,17,18,19,20.

Далее после удаления чисел с четными номерами остались числа: 7,10,12,15,17,19. После удаления чисел с нечетными номерами остались числа: 7,12,17. Осталось три числа. При удалении по Фибоначчи мы должны удалить все из них, но мы оставляем последнее число в качестве ответа: 17.

Решение этой задачи с помощью программы, написанной на языке Matlab, представлено ниже.

```
clear all
N=200000;
%% это исходный массив
a=1: N;

tic
while 1
n = length(a);
%% специальные случаи, когда после удаления по
%% Фибоначчи ни один элемент не остается
%% или остается только один элемент
if n==1
   disp('оставшееся число');
  disp(a(1));
```

Таблица 1. Данные расчета по программе Matlab

Начальное количество элементов в массиве	Оставшееся число	Время [сек.]
200000	153932	0.0126
2000000	1922357	0.0986
20000000	15379102	0.8081
200000000	157635910	6.645

```
    break;
  end
  if n==2 | n==3 | n==4 | n==5
    disp('оставшееся число');
    disp(a(end));
    break;
  end
  if n==6
    disp('оставшееся число');
    disp(a(5));
    break;
  end
  %% закончим рассматривать специальные случаи
  %% L это массив, в котором хранятся числа Фибоначчи
  clear L
  L=[0 1];
  k=length(L);
  while 2
  %% проверяем, является ли очередное число Фибоначчи
  %% (плюс 1) больше количества элементов в массиве;
  %% прибавляем 1, чтобы правильно сделать сравнение
    if L(k)+L(k-1) + 1 > length(a)
      break;
    end
  %% записываем новое число Фибоначчи в массив L
    L=[L, L(k)+L(k-1)];
    k=k+1;
  end%% закрыть цикл while 2
  %% векторизованная операция удаления элементов;
  %% должны прибавить 1 к номеру из L,
  %% так как индексация элементов в Matlab начинается
  %% с 1
  a(L(1: end)+1)=[];%% исключение и сдвиг

  n=length(a);
  if n==1
    break;
  end
  %% удаляем из массива числа с четными номерами
  a(1:2: n)=[];%% исключение и сдвиг
  n=length(a);
  if n==1
    break;
  end
  %% удаляем из массива числа с нечетными номерами
  a(2:2: n)=[];%% исключение и сдвиг
  end%% закрыть цикл while 1
  toc
  n=length(a);
  if n==1
  disp('оставшееся число');
  disp(a);
  end
  if n>1
  disp('оставшиеся числа перед удалением по Фибоначчи');
  disp(a);
  end
```

Время измеряется при помощи часов, которые включаются по команде tic и выключаются по команде toc. Исключение элементов из массива происходит с помощью векторизованной операции

```
a(indx)=[];
```

где массив индексов indx содержит номера тех элементов массива, которые необходимо удалить.

Чтобы показать эффективность этой программы, попробуем выполнить расчет для массивов с большим количеством элементов N. Таблица 1, представленная ниже, приводит результат расчета — последнее оставшееся число в массиве — и указывает время, которое было затрачено на выполнение расчета.

Расчет производился на компьютере с оперативной памятью 8 Гб. Мы видим, что программа работает быстро даже для массивов с большим количеством элементов. Также отметим, что для компьютера с данным размером оперативной памяти (8 Гб) количество элементов в массиве не может превосходить примерно 200 млн. элементов, поэтому данные для массивов большей размерности отсутствуют. Это ограничение на размер массива возникает из-за переполнения памяти.

Далее выполним ту же задачу на языке C/C++. Код основной функции программы представлен ниже.

```
#include "stdafx.h"
#include <stdio.h>
#include <time.h>
// объявление функции, которая будет удалять
// элементы по Фибоначчи
int fib_remove(long int*, int);
using namespace System;
int main(array<System:: String ^> ^args)
{
   int n = 200000;
   long int *a;
   int i, j, k;
// формируем заданный массив
   a = new long int[n];
   for (i = 0; i < n; i++)
      a[i] = i + 1;
   clock_t begin=clock();
   while (1) {
      if (n <= 4) break;
       // удаляем из массива a числа по Фибоначчи;
       // функция возвращает количество оставшихся
      // элементов
      n = fib_remove(a, n);

      // удаляем из массива a числа с четными номе-
         рами

      if (n <= 1) break;
      k = 0;
      for (i = 1; i < n; i += 2) {
// оставляем число с нечетным номером и
// сдвигаем его влево
         a[k] = a[i];
         k++;
      }
   n = k;

//удаляем из массива a числа с нечетными номерами

      if (n <= 1) break;
      k = 0;
      for (i = 0; i < n; i += 2) {
// оставляем число с четным номером и
// сдвигаем его влево
         a[k] = a[i];
         k++;
      }
      n = k;
   }
   clock_t end=clock();
   printf(«\nКоличество оставшихся элементов%d\n»,
   n);
   printf(«\nОставшиеся числа:\n»);
   for (j = 0; j < n; j++) {
      printf("%d ", a[j]);
   }
   if (n>1)
      printf(«\n последнее число:%d «, a[n-1]);

   double ct = double(end-begin);
   printf("\n время:%f ", ct/CLOCKS_PER_SEC);
   getchar();
   return 0;
}
```

Обратим внимание на то, что переменная n постоянно меняет свое значение в этой программе, и оно всегда равно количеству оставшихся элементов в массиве a. Также отметим, что в действительности мы не удаляем числа из массива a и не меняем его размер, а лишь переносим нужные элементы влево.

Функция fib_remove представлена ниже.

```
// функция для удаления по Фибоначчи
// a — массив чисел,
// n — количество оставшихся элементов в нем

int fib_remove(long int* a, int n) {

   int j;
// новое количество оставшихся элементов
   int new_n = 0;
// s1, s2 — пара соседних чисел Фибоначчи
   int s1 = 3, s2 = 5;
// k — номер позиции числа s1 в ряду чисел
// Фибоначчи (номер позиции начинаем с нуля)

   int k = 3;

   while (1) {
// подошли к элементу с номером s1.
// цикл for: берем все нужные элементы
// и сдвигаем их влево на k+1 позиций
      for (j = s1 + 1; j < s2; j++) {
         a[j — (k + 1)] = a[j];
         new_n++;
      }
// далее нам придется сдвигать числа влево
// на одну позицию больше, так как мы подойдем к
// следующему элементу с номером,
// равным числу Фибоначчи, поэтому увеличиваем k
      k++;
// переопределяем s1, s2 как следующая пара
// чисел Фибоначчи
      int s_tmp = s2;
      s2 = s1 + s2;
      s1 = s_tmp;
// если s2 больше n, то предыдущий цикл for
// не применим
```

Таблица 2. Данные расчета по программе C

Начальное количество элементов в массиве	Оставшееся число	Время [сек.]
200000	153932	0.003
2000000	1922357	0.017
20000000	15379102	0.112
200000000	157635910	1.126

Таблица 3. Данные расчета по программе C при выполнении только удалений по Фибоначчи

Начальное количество элементов в массиве	Оставшееся число	Время [сек.]
200000	199996	2.06
2000000	1999999	176.2
20000000	?	?
200000000	?	?

```
        if (s2 >= n) break;
    }

// сейчас уже s2 больше n.
// рассматриваем элементы в хвосте:
// с номерами между s1 + 1 и n-1
// их тоже надо оставить
    for (j = s1 + 1; j < n; j++) {
        a[j — (k + 1)] = a[j];
        new_n++;
    }
    n = new_n;
    return n;
}
```

В этой функции мы также сдвигаем нужные элементы влево, записывая их на место ненужных элементов, если это необходимо. Как определить, насколько позиций необходимо сдвинуть нужное число? Назовем число, номер которого равен k-му числу Фибоначчи, k-м ненужным элементом, k=0,1,2,… Тогда понятно, что числа, которые стоят в массиве a между k-м ненужным элементом и k+1-м ненужным элементом, должны остаться в массиве a и их нужно сдвинуть на k + 1 позиций влево. Это и происходит в программе по команде a[j — (k + 1)] = a[j]. Другими словами, мы сдвигаем нужный элемент на такое количество позиций, которое равно количеству ненужных элементов, предшествующих ему.

Таблица 2. приводит результат расчета и время, которое было затрачено на выполнение расчета, используя данную программу C.

Мы видим, что время, потраченное на выполнение программы на языке C, меньше того времени, которое было потрачено на выполнение программы на языке Matlab, однако порядок цифр остается таким же.

Если мы в каждом цикле будем выполнять исключения только по Фибоначчи, то время расчета, значительно увеличится. Эти результаты представлены ниже для программы, написанной на языке C.

Для массивов большой размерности нам не удалось получить необходимый результат из-за затрат времени.

В заключение отметим, что различные интересные задачи на операции с массивами представлены в книгах [1–3].

ЛИТЕРАТУРА

1. Шень А. Программирование: теоремы и задачи. — М.: МЦНМО, 2014. — 296 с.
2. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ.: Пер. с англ. — М.: издательский дом «Вильямс», 2009. — 1296 с.
3. Стивенс Р. Алгоритмы. Теория и практическое применение. — М.: изд-во «Э», 2016. — 544 с.

Журнал «Современная наука: актуальные проблемы теории и практики»