НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Продвинутые алгоритмы и структуры данных

Отчет о задаче № A

Выполнила
студентка

Ершова А. И.

Группа № P4115

Преподаватель: Косяков Михаил Сергеевич

г. Санкт-Петербург

2024

Условие задачи

A. K-ый ноль

Ограничение времени	0.5 секунд
Ограничение памяти	128Mb
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Реализуйте эффективную структуру данных, позволяющую изменять элементы массива и вычислять индекс k-го слева нуля на данном отрезке в массиве.

Формат ввода

В первой строке вводится одно натуральное число N ($1 \leq N \leq 200\ 000$) — количество чисел в массиве. Во второй строке вводятся N чисел от 0 до $100\ 000$ — элементы массива. В третьей строке вводится одно натуральное число M ($1 \leq M \leq 200\ 000$) — количество запросов. Каждая из следующих M строк представляет собой описание запроса. Сначала вводится одна буква, кодирующая вид запроса (`s` — вычислить индекс k-го нуля, `u` — обновить значение элемента). Следом за `s` вводится три числа — левый и правый концы отрезка и число k ($1 \leq k \leq N$). Следом за `u` вводятся два числа — номер элемента и его новое значение.

Формат вывода

Для каждого запроса s выведите результат. Все числа выводите в одну строку через пробел. Если нужного числа нулей на запрашиваемом отрезке нет, выводите -1 для данного запроса.

Пример

Ввод	Вывод
5 0 0 3 0 2 3 u 1 5 u 1 0 s 1 5 3	4

Решение:

```
#include <cmath>
#include <fstream>
#include <iostream>
#include <vector>

using namespace std;

int n;
vector<int> arr;
int block_size;
vector<int> blocks;

void BuildBlocks(int n) {
 for (int i = 0; i < n; i++) {
   if (arr[i] == 0) {
     int index = i / block_size;
     blocks[index]++;
   }
 }
}

void Update(const int pos, const int new_val) {
 int block_index = pos / block_size;
 if (arr[pos] == 0) {
   blocks[block_index]--;
 }
 if (new_val == 0) {
   blocks[block_index]++;
 }
 arr[pos] = new_val;
}

int FindKthZero(const int left_border, const int right_border, const
int k) {
 if (k > n) {
   return -1;
 }
 int count_zeros = 0;

 int const first_block = left_border / block_size;
 int const last_block = right_border / block_size;

 for (int i = first_block; i <= last_block; i++) {
   int const first_block_index = i * block_size;
   int const last_block_index = min(((i + 1) * block_size) - 1, n -
1);
```

```
   if (first_block_index >= left_border && last_block_index <=
right_border) {
     if (count_zeros + blocks[i] < k) {
       count_zeros += blocks[i];
     } else {
       for (int j = first_block_index; j <= last_block_index; j++) {
         if (arr[j] == 0) {
           count_zeros += 1;
           if (count_zeros >= k) {
             return j;
           }
         }
       }
     }
   } else {
     int start_check_index = max(first_block_index, left_border);
     int end_check_index = min(last_block_index, right_border);
     for (int j = start_check_index; j <= end_check_index; j++) {
       if (arr[j] == 0) {
         count_zeros += 1;
         if (count_zeros >= k) {
           return j;
         }
       }
     }
   }
 }
 return -1;
}

int main() {
 ifstream fin("../input.txt");
 fin >> n;
 arr.resize(n);
 for (int i = 0; i < n; i++) {
   fin >> arr[i];
 }

 block_size = sqrt(n);
 blocks.resize((n + block_size - 1) / block_size, 0);
 BuildBlocks(n);

 int m;
 fin >> m;
 string answer = "";
 for (int i = 0; i < m; i++) {
   char type;
   fin >> type;
   if (type == 'u') {
     int index;
```

```
      int new_val;
      fin >> index >> new_val;
      Update(index - 1, new_val);
    } else if (type == 's') {
      int left_border = -1;
      int right_border = -1;
      int k = -1;
      fin >> left_border >> right_border >> k;
      int result = FindKthZero(left_border - 1, right_border - 1, k);
      if (result == -1) {
        answer.append("-1 ");
      } else {
        answer.append(to_string(result + 1));
        answer.append(" ");
      }
    }
  }
  fin.close();
  cout << answer << '\n';
  return 0;
}
```

Сложность:

Инициализация: O(N).

Обработка m запросов: $O(m * \sqrt{N})$.

Общая сложность: $O(n + m * \sqrt{N})$.

Объяснение решения:

Для решения данной задачи был использован метод корневой декомпозиции массива.

Для определения размера блока берется корень длины входного массива в соответствии с алгоритмом Мо. Далее используется функция BuildBlocks, создающая массив blocks, где каждому значению соответствует количество нулей в блоке входного массива.

При введении команды “u” вызывается функция Update, которая определяет в каком из блоков происходят изменения. В соответствии с этим она либо уменьшает, либо увеличивает на 1 значение в массиве blocks, также она производит изменения в оригинальном массиве arr.

При введении команды “s” вызывается функция FindKthZero. Если k было введено больше чем всего чисел в массиве, то функция сразу возвращает “-1”. Далее она определяет номер первого и последнего блока, в которых располагается необходимый диапазон. После этого функция проходится по каждому из выбранных блоков, определяя находится ли блок полностью в диапазоне или частично. Если

частично, то тогда функция считает количество нулей в блоке, опираясь на левую и правую границу диапазона. Возвращается индекс k-того нуля. В основной функции main к результату добавляется 1, так как по условию задачи нумерация заданного массива идет с единицы.

Вывод:

Был изучен и рассмотрен метод корневой декомпозиции массива и алгоритм Мо.