ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «КАЗАНСКИЙ(ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ

ОТЧЕТ ПО ТЕМЕ:

**«ДЕРЕВО ОТРЕЗКОВ»**

Выполнила: Федорчук Виктория, гр. 11-002

Казань 2021г.

1.Дерево отрезков.

2. *Основной принцип устройства. Особенности.*

Дерево отрезков - это структура данных, позволяющая находить значение некоторой ассоциативной функции f на произвольном отрезке a[i], a[i+1],…,a[j] массива за асимптотику O(log n). Наиболее часто в качестве f берутся функции суммы, произведения, максимум и минимум.

Дерево отрезков представляет собой корневое дерево, листьями которого являются элементы исходного массива, а другие вершины имеют по 2 ребенка. Каждой вершине в соответствие поставлен интервал, являющийся объединением интервалов ее детей (если у вершины есть дети), либо интервал, содержащий конкретный элемент массива (для листьев). Кроме того, для каждой вершины хранится значение некоторой ассоциативной функции f на данном интервале. Данное дерево будет иметь логарифмическую высоту, так как количество уровней не будет превышать log2 n.

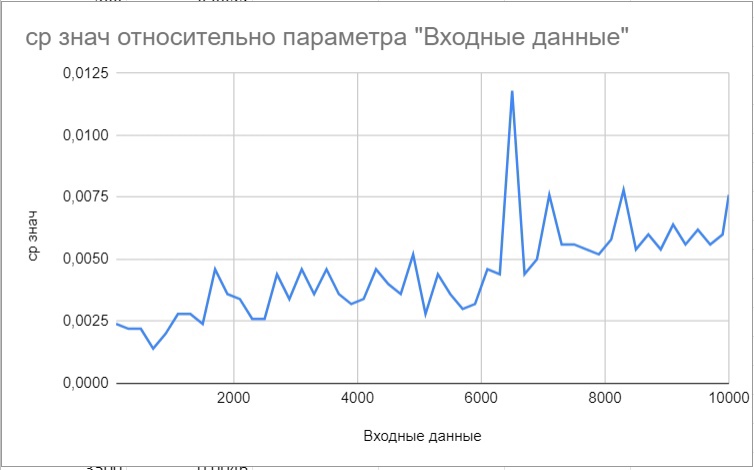
*3. Оценка времени*

Процесс построения дерева заключается в заполнении массива t. Заполним этот массив таким образом, чтобы i-й элемент являлся бы результатом некоторой бинарной операции (для каждой конкретной задачи своей) от элементов c номерами 2i+1 и 2i+2, то есть родитель являлся результатом бинарной операции от своих сыновей (обозначим в коде эту операцию как "∘"). Один из вариантов — делать рекурсивно. Пусть у нас имеются исходный массив a, а также переменные tl и tr, обозначающие границы текущего полуинтервала. Запускаем процедуру построения от корня дерева отрезков (i=0, tl=0, tr=n), а сама процедура построения, если её вызвали не от листа, вызывает себя от каждого из двух сыновей и суммирует вычисленные значения, а если её вызвали от листа — то просто записывает в себя значение этого элемента массива (Для этого у нас есть исходный массив a). Асимптотика построения дерева отрезков составит, таким образом, O(n).

Чтобы найти сумму на любом из отрезков массива, при построении дерева на каждой его вершине будем хранить сумму на соответствующем отрезке. Выполняя данные действия рекурсивно, получим сложность O(log n).

*5. Графики*

Построение дерева:



Подсчет суммы(в контексте задачи):



*6. Выводы. Плюсы и минусы структуры данных, ее применимость.*

Основные плюсы дерева отрезков - малые константы времени и памяти, а также простота реализации.

Минус - подходит не для всех типов задач.

Дерево отрезков - очень гибкая структура, с помощью которой можно решать большое количество задач, например:

* поиск минимума/максимума(и количество раз, которое он встречается);
* поиск наибольшего общего делителя/наименьшего общего кратного;
* подсчет количества нулей, поиск k-го нуля;
* поиск подотрезка с максимальной суммой;
* обновление на отрезке
* прибавление на отрезке и т.д.

7. *Список литературы.*

* [*https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE\_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%BE%D0%B2*](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%BE%D0%B2)
* [*https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE\_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%BE%D0%B2.\_%D0%9F%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B5%D0%BD%D0%B8%D0%B5*](https://neerc.ifmo.ru/wiki/index.php?title=%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BE%D1%82%D1%80%D0%B5%D0%B7%D0%BA%D0%BE%D0%B2._%D0%9F%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B5%D0%BD%D0%B8%D0%B5)
* [*https://algorithmica.org/ru/segtree*](https://algorithmica.org/ru/segtree)
* [*http://e-maxx.ru/algo/segment\_tree*](http://e-maxx.ru/algo/segment_tree)

*8. Код программы.*

[#include](https://vk.com/im?sel=2000000038&st=%23include) <iostream>   
   
using namespace std;   
   
struct tree {   
    long \*tr;   
   
    tree(int n) {   
        tr = new long[4 \* n + 1];   
    }   
 

void build(int \*arr, int v, int l, int r) {   
        if (l == r) tr[v] = arr[l];   
        else {   
            int mid = (l + r) / 2;   
            build(arr, v \* 2, l, mid);   
            build(arr, v \* 2 + 1, mid + 1, r);   
            tr[v] = tr[v \* 2] + tr[v \* 2 + 1];   
        }   
    }   
   
    void update(int v, int l, int r, int pos, int val) {   
        if (l == r) tr[v] = val;   
        else {   
            int mid = (l + r) / 2;   
            if (pos <= mid) update(v \* 2, l, mid, pos, val);   
            else update(v \* 2 + 1, mid + 1, r, pos, val);   
            tr[v] = tr[v \* 2] + tr[v \* 2 + 1];   
        }   
    }   
   
    int find(int v, int l, int r, int max) {   
        if (l == r)   
            return (tr[v] <= max);   
        int mid = (l + r) / 2;   
        if (tr[v \* 2] > max)   
            return find(v \* 2, l, mid, max);   
        else   
            return mid - l + 1 + find(v \* 2 + 1, mid + 1, r, max - tr[v \* 2]);   
    }   
};   
   
int main() {   
    int n, \*arr;   
    cin >> n;   
    arr = new int[n + 1];   
    for (int i = 1; i <= n; ++i) cin >> arr[i];   
    tree tr(n);   
   tr.build(arr, 1, 1, n);   
    int v;   
    cin >> v;   
    cout << tr.find(1, 1, n, v) << '\n';   
    return 0;   
}

9. *Входные данные*

<https://github.com/Viktoria223/JAVA/tree/main/Tests>

4*.Таблица полученных значений в зависимости от размера данных*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Построение дерева | | | | | | Подсчет суммы | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | Ср знач | 1 | 2 | 3 | 4 | 5 | Ср знач |
| 100 | 0,002 | 0,003 | 0,003 | 0,002 | 0,002 | 0,0024 | 0,001 | 0 | 0 | 0,001 | 0,001 | 0,0006 |
| 300 | 0,002 | 0,002 | 0,003 | 0,002 | 0,002 | 0,0022 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 500 | 0,002 | 0,002 | 0,002 | 0,003 | 0,002 | 0,0022 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 700 | 0,001 | 0,002 | 0,001 | 0,001 | 0,002 | 0,0014 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 900 | 0,001 | 0,003 | 0,002 | 0,003 | 0,001 | 0,002 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 1100 | 0,003 | 0,003 | 0,002 | 0,003 | 0,003 | 0,0028 | 0 | 0,001 | 0 | 0,001 | 0,001 | 0,0006 |
| 1300 | 0,003 | 0,003 | 0,003 | 0,002 | 0,003 | 0,0028 | 0,001 | 0,001 | 0 | 0,001 | 0,001 | 0,0008 |
| 1500 | 0,003 | 0,002 | 0,002 | 0,002 | 0,003 | 0,0024 | 0,002 | 0 | 0,001 | 0 | 0,002 | 0,001 |
| 1700 | 0,004 | 0,005 | 0,005 | 0,005 | 0,004 | 0,0046 | 0 | 0,001 | 0,001 | 0,001 | 0 | 0,0006 |
| 1900 | 0,004 | 0,004 | 0,003 | 0,004 | 0,003 | 0,0036 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 2100 | 0,004 | 0,003 | 0,003 | 0,003 | 0,004 | 0,0034 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 2300 | 0,003 | 0,002 | 0,003 | 0,003 | 0,002 | 0,0026 | 0,001 | 0,001 | 0 | 0,001 | 0,001 | 0,0008 |
| 2500 | 0,003 | 0,003 | 0,002 | 0,002 | 0,003 | 0,0026 | 0 | 0,001 | 0,001 | 0,001 | 0,001 | 0,0008 |
| 2700 | 0,006 | 0,003 | 0,004 | 0,004 | 0,005 | 0,0044 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 2900 | 0,004 | 0,003 | 0,003 | 0,004 | 0,003 | 0,0034 | 0,002 | 0,001 | 0,001 | 0,002 | 0,002 | 0,0016 |
| 3100 | 0,003 | 0,006 | 0,006 | 0,004 | 0,004 | 0,0046 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 3300 | 0,004 | 0,003 | 0,004 | 0,004 | 0,003 | 0,0036 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 | 0,002 |
| 3500 | 0,004 | 0,005 | 0,004 | 0,005 | 0,005 | 0,0046 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 3700 | 0,004 | 0,004 | 0,003 | 0,004 | 0,003 | 0,0036 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 3900 | 0,003 | 0,003 | 0,004 | 0,003 | 0,003 | 0,0032 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 4100 | 0,004 | 0,003 | 0,003 | 0,004 | 0,003 | 0,0034 | 0,001 | 0,001 | 0,001 | 0,002 | 0 | 0,001 |
| 4300 | 0,005 | 0,005 | 0,004 | 0,005 | 0,004 | 0,0046 | 0 | 0,001 | 0 | 0 | 0,001 | 0,0004 |
| 4500 | 0,003 | 0,005 | 0,003 | 0,004 | 0,005 | 0,004 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 4700 | 0,003 | 0,004 | 0,004 | 0,004 | 0,003 | 0,0036 | 0 | 0,001 | 0 | 0,001 | 0,001 | 0,0006 |
| 4900 | 0,004 | 0,006 | 0,006 | 0,005 | 0,005 | 0,0052 | 0 | 0,002 | 0,001 | 0,002 | 0,002 | 0,0014 |
| 5100 | 0,004 | 0,002 | 0,002 | 0,003 | 0,003 | 0,0028 | 0,001 | 0,001 | 0,001 | 0,002 | 0,001 | 0,0012 |
| 5300 | 0,003 | 0,006 | 0,004 | 0,004 | 0,005 | 0,0044 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 5500 | 0,004 | 0,003 | 0,004 | 0,004 | 0,003 | 0,0036 | 0,001 | 0 | 0 | 0,001 | 0,001 | 0,0006 |
| 5700 | 0,004 | 0,002 | 0,004 | 0,002 | 0,003 | 0,003 | 0,001 | 0 | 0,001 | 0 | 0 | 0,0004 |
| 5900 | 0,004 | 0,003 | 0,003 | 0,003 | 0,003 | 0,0032 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 6100 | 0,004 | 0,005 | 0,004 | 0,005 | 0,005 | 0,0046 | 0,001 | 0,001 | 0 | 0 | 0,001 | 0,0006 |
| 6300 | 0,004 | 0,005 | 0,005 | 0,004 | 0,004 | 0,0044 | 0,001 | 0,001 | 0,001 | 0,002 | 0 | 0,001 |
| 6500 | 0,003 | 0,006 | 0,04 | 0,005 | 0,005 | 0,0118 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 6700 | 0,005 | 0,004 | 0,004 | 0,004 | 0,005 | 0,0044 | 0,003 | 0,001 | 0,002 | 0,002 | 0,002 | 0,002 |
| 6900 | 0,004 | 0,006 | 0,006 | 0,004 | 0,005 | 0,005 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 7100 | 0,009 | 0,006 | 0,007 | 0,008 | 0,008 | 0,0076 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 | 0,001 |
| 7300 | 0,005 | 0,006 | 0,006 | 0,006 | 0,005 | 0,0056 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 7500 | 0,005 | 0,006 | 0,005 | 0,006 | 0,006 | 0,0056 | 0,001 | 0,001 | 0,001 | 0,001 | 0 | 0,0008 |
| 7700 | 0,005 | 0,006 | 0,006 | 0,005 | 0,005 | 0,0054 | 0 | 0 | 0,001 | 0 | 0 | 0,0002 |
| 7900 | 0,005 | 0,006 | 0,005 | 0,005 | 0,005 | 0,0052 | 0,001 | 0,001 | 0,001 | 0 | 0,001 | 0,0008 |
| 8100 | 0,005 | 0,007 | 0,006 | 0,006 | 0,005 | 0,0058 | 0,002 | 0,001 | 0,001 | 0,001 | 0,002 | 0,0014 |
| 8300 | 0,007 | 0,009 | 0,007 | 0,007 | 0,009 | 0,0078 | 0,001 | 0,002 | 0,002 | 0,001 | 0,001 | 0,0014 |
| 8500 | 0,004 | 0,007 | 0,004 | 0,007 | 0,005 | 0,0054 | 0,001 | 0,001 | 0,001 | 0,002 | 0,001 | 0,0012 |
| 8700 | 0,006 | 0,006 | 0,006 | 0,006 | 0,006 | 0,006 | 0,001 | 0,002 | 0,001 | 0,002 | 0,002 | 0,0016 |
| 8900 | 0,004 | 0,006 | 0,005 | 0,006 | 0,006 | 0,0054 | 0,001 | 0,002 | 0,001 | 0,001 | 0,001 | 0,0012 |
| 9100 | 0,007 | 0,006 | 0,006 | 0,006 | 0,007 | 0,0064 | 0,001 | 0,001 | 0 | 0,001 | 0,001 | 0,0008 |
| 9300 | 0,006 | 0,006 | 0,005 | 0,005 | 0,006 | 0,0056 | 0,001 | 0,001 | 0,001 | 0 | 0 | 0,0006 |
| 9500 | 0,004 | 0,008 | 0,006 | 0,006 | 0,007 | 0,0062 | 0 | 0,001 | 0 | 0 | 0,001 | 0,0004 |
| 9700 | 0,006 | 0,006 | 0,005 | 0,005 | 0,006 | 0,0056 | 0 | 0,001 | 0,001 | 0,001 | 0,001 | 0,0008 |
| 9900 | 0,004 | 0,008 | 0,006 | 0,008 | 0,004 | 0,006 | 0 | 0,001 | 0,001 | 0 | 0,001 | 0,0006 |
| 10000 | 0,007 | 0,008 | 0,008 | 0,007 | 0,008 | 0,0076 | 0,001 | 0,002 | 0,002 | 0,002 | 0,002 | 0,0018 |