

УДК 004.021

DOI 10.21661/r-561596

Невский Алексей Александрович

бакалавр, ведущий инженер-программист

Glovo

г. Минск, Республика Беларусь

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ПРОБЛЕМ СВЯЗНОГО СПИСКА С ИСПОЛЬЗОВАНИЕМ ТЕХНИКИ БЫСТРОГО И МЕДЛЕННОГО УКАЗАТЕЛЕЙ

Аннотация: в статье рассмотрена комбинация метода быстрого и медленного указателей и других техник на языке высокоуровневого программирования Java для эффективного решения популярных алгоритмических проблем связанного списка. Результат включает в себя описание проблемы, реализацию высокопроизводительного алгоритма и решение вычислительной задачи.

Ключевые слова: быстрый указатель, медленный указатель, алгоритм, разработка программного обеспечения, программное обеспечение, структура данных, связанный список, язык программирования Java, обработка данных, эффективные вычисления.

Актуальность проблемы.

Linked List – одна из наиболее популярных структур данных, где каждый узел имеет какое-либо заданное значение и связан со следующим узлом указателем на область памяти.

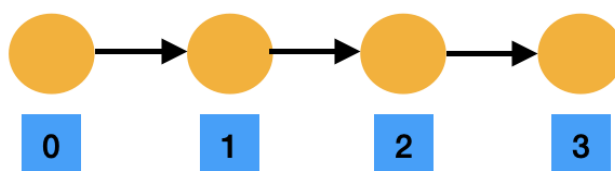


Рис. 1. Структура данных *Linked List* из 4 элементов

Умение эффективно использовать и манипулировать указателями структур данных и областей компьютерной памяти, придерживание общепринятых соглашений по именованию [1] критически важно для любого программиста, занимающегося созданием эффективных и отлаженных программ. Широкий класс задач, включающий в себя структуру данных Linked List, может быть эффективно решен, используя метод быстрого и медленного указателей, а также таких техник как упорядочивание в обратном порядке, определение наличия цикла и нахождение серединного элемента. Разумная комбинация и знание особенностей реализаций позволяет создавать эффективные алгоритмы для решения широкого класса таких задач [5].

Например, в задаче «Максимальная сумма пары из начала и конца связного списка» запрашивается максимальная сумма такой пары, где это одна из пар, состоящих из таких узлов связного списка как первый и последний узел, второй и предпоследний узел, третий и третий предпоследний узел и т. д.

Описание предложенного решения.

Предположим, что дан связный список, состоящий из следующих элементов: [4, 2, 2, 3].

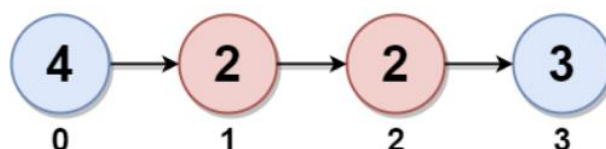


Рис. 2. Входные данные для примера задачи из класса «Максимальная сумма пары из начала и конца связного списка»

Максимальная сумма будет равна 7, так как.

1. Первая пара состоит из узлов с индексом 0 и 3, их сумма $4 + 3 = 7$.
2. Вторая пара состоит из узлов с индексом 1 и 2, их сумма $2 + 2 = 4$.

Таким образом, максимальная сумма пары из начала и конца связного списка равна $\max(7, 4) = 7$.

Тривиальным решением было бы преобразовать связанный список в массив, чтобы мы могли легко получить доступ к парам путем индексации.

Более элегантный и эффективный алгоритм выглядит следующим образом:

– шаг 1: найдем середину связного списка, используя технику быстрого и медленного указателей;

– шаг 2: оказавшись в середине связного списка, выполним разворот второй половины списка, т.е. упорядочим вторую часть в обратном порядке;

После упорядочивания второй половины каждый узел располагается на расстоянии $(n / 2)$ от своего парного узла, где (n) – количество узлов в списке, который мы можем найти на шаге (1).

– шаг 3: создадим еще один быстрый указатель на расстоянии $(n / 2)$ впереди медленного;

– шаг 4: выполним итерацию $(n / 2)$ раз от начала списка, чтобы найти сумму каждой пары указателей на узлы `slow.val + fast.val`.

Реализуем алгоритм на языке программирования Java:

```

1  /**
2  * Definition for singly-linked list.
3  * public class ListNode {
4  *     int val;
5  *     ListNode next;
6  *     ListNode() {}
7  *     ListNode(int val) { this.val = val; }
8  *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
9  * }
10 */

```

Рис. 3. Определение структуры данных *Linked List* на языке программирования *Java*

```

11
12 class Solution {
13     public int pairSum(ListNode head) {
14         // find middle
15         ListNode slow = head;
16         ListNode fast = head.next;
17         ListNode middle = null;
18         ListNode tail = null;
19         while(fast != null && fast.next != null) {
20             slow = slow.next;
21             fast = fast.next.next;
22         }
23         middle = slow.next;
24         tail = fast;
25
26         // reverse 2nd part of list
27         slow.next = reverse(middle);
28
29         // iterate from the beggining and middle
30         middle = tail;
31         int ans = Integer.MIN_VALUE;
32         while (middle != null) {
33             ans = Math.max(ans, head.val + middle.val);
34             head = head.next;
35             middle = middle.next;
36         }
37
38         return ans;
39     }

```

Рис. 4. Ключевая реализация алгоритма с использованием техники медленного и быстрого указателей на языке программирования Java

```

40
41     public ListNode reverse(ListNode head) {
42         ListNode prev = null;
43         while (head != null) {
44             ListNode next = head.next;
45             head.next = prev;
46             prev = head;
47             head = next;
48         }
49         return prev;
50     }
51 }

```

Рис. 5. Упорядочивание в обратном порядке структуры данных *Linked List* на языке программирования Java

Вывод.

Представленный алгоритм на основе комбинации метода быстрого и медленного указателей и других техник на языке высокоуровневого программирования Java корректен [2], интуитивен, прост для понимания [3] и высокоэффективен, так как обладает асимптотической сложностью по времени $O(n)$ и компьютерной памяти $O(1)$.

Используя данный алгоритм для связного списка, можно легко улучшить производительность компьютерных вычислений, сократить время исполнения, потребление памяти и повысить эффективность разрабатываемого приложения, что является одним из важнейших факторов правильно спроектированной программы и демонстрацией правильного использования подходящих структур данных, а также техник написания высокопроизводительных алгоритмов [4].

Список литературы

1. Блох Дж. Java. Эффективное программирование / Дж. Блох. – 3-е изд. – М.: Диалектика, 2019. – 464 с.
2. Клеппман М. Высоконагруженные приложения. Программирование, масштабирование, поддержка / М. Клеппман. – СПб.: Питер, 2020. – С. 30.
3. Мартин Р. Чистый код. Создание, анализ и рефакторинг / Р. Мартин. – СПб.: Питер, 2019. – С. 60.
4. Heather Adkins, Betsy Beyer, Paul Blankinship, Piotr Lewandowski, Ana Oprea & Adam Stubblefield. Building Secure & Reliable Systems / CA, US.: O'Reilly Media, 2020. – p. 95.
5. Robert Sedgewick, Kevin Wayne. Algorithms, 4th Edition / US.: Addison-Wesley Professional, 2011. – p. 5.