Обзор биномиальных пирамид на основе различных систем счисления

В.К. Гулаков, К.В. Гулаков

Аннотация **- В статье дается обзор биномиальных структур данных на основе различных систем счисления. Такой подход позволяет улучшить производительность операций над биномиальными пирамидами, такими как увеличение и уменьшение числа элементов, вставку элементов, слияния пирамид. Рассматриваются различные системы счисления: бинарные, избыточные, без нулевые, регулярные, косые и их комбинации. Структура биномиальных пирамид хорошо описывается бинарным числом, равным количеству элементов в пирамиде. Работая с этими числами, можно эффективно выполнять различные операции над пирамидальными структурами. Проблема представления бинарного числа заключается в том, что увеличение или уменьшение числа на единицу может привести к каскадному изменению многих цифр числа, что влияет на результат в худшем случае. Для того, чтобы уйти от этой ситуации, используются другие системы счисления. В статье кратко рассматриваются различные системы счисления, их комбинации и их возможности.**

Ключевые слова: **деревья, биноминальная пирамида, избыточные системы счисления, сложность алгоритма.**

Введение

Совершенствование и развитие пирамидальных структур данных как правило, базируется на двух основных направлениях. Первое заключается в создании новых форм хранения данных и новой структурной архитектуре, что в совокупности влияет на время работы различных операций, производимых над пирамидальной структурой данных. Второе направление – ослабление основного свойства пирамиды и ослабление правил и ограничений, накладываемых на пирамидальную структуру. Примером может служить AVL-дерево. В целом анализ существующих тенденций в развитии пирамидальных структур рассмотрен в работах [1, 7].

```
  11111111
   99999999
+         1
 1000000000
```

Рис.1. Каскадное сложение

Новый подход в развитии пирамидальных структур - взгляд на пирамиды с позиции других систем счисления. Взаимосвязь между системами счисления и структурами данных достаточно проста и впервые описана в записках семинара Кленси и Кнута [6].

Предполагается, что представление биномиальных пирамид в других системах счисления приведет к уменьшению количества сравнений и количества слияний биномиальных деревьев.

В десятичной системе счисления одно приращение может привести к каскадным сложениям (Рис.1). У бинарной системы счисления есть та же проблема.

Мы ищем систему, в которой приращение и уменьшение может быть выполнено с постоянным количеством изменений цифр.

Каждой биномиальной пирамиде в соответствующей системе счисления соответствует некоторое число d, непосредственно обозначающее количество элементов в пирамиде. Введем строку $<d_{k-1}, \ldots, d_1, d_0>$, которая будет являться представлением числа d в какой-либо из представленных ниже систем счисления. Тогда значение числа d будет рассчитываться как $\sum_{i=0}^{k-1} d_i * w_i$, где:

- i – индекс цифры в строке числа или ранг биномиального дерева или степень исхода из вершины соответствующего биномиального дерева;
- d_i – цифра в представлении числа d, соответствующая количеству деревьев i-го ранга в пирамиде;
- w_i – значение i-й цифры в представлении числа, которое соответствует количеству элементов в дереве i-го ранга в пирамиде.

Для каждого представления пирамиды в различных системах счисления её размер n содержит d_i биномиальных деревьев размером $w_i=2^i$ или d_i косых биномиальных деревьев размером $w_i=2^{i+1}-1$.

Требования к системам счисления:

- $|\{d_i|i \in \{0, 1, \ldots, k-1\}\}|$ минимальная, насколько это возможно для всех d_i;
- Увеличение в любой позиции i (increment (d, i)) генерирует как можно меньше изменений цифр в худшем случае;
- Уменьшение в любой позиции i (decrement (d, i)) генерирует как можно меньше изменений цифр в худшем случае.

Для реализации этих и других улучшений используются свойства различных систем счисления: избыточных, безнулевых, регулярных, косых и их комбинаций.

В данный момент для биномиальных пирамид рассматриваются следующие системы счисления:

- бинарная (binary) $d_i \in \{0,1\}$, $w_i = 2^i$ [14];
- избыточная бинарная (redundant binary) $d_i \in \{0,1,2\}$, $w_i = 2^i$ [5];
- безнулевая бинарная (zeroless binary) $d_i \in \{1,2\}$, $w_i = 2^i$ [14];
- избыточная безнулевая (redundant zeroless) $d_i \in \{1,2,3,4\}$, $w_i = 2^i$ [8, 11];
- избыточная регулярная (redundant regular - RR): $d_i \in \{0,1,2\}$, $w_i = 2^i$. [6];

- расширенная регулярная система (extended regular): $d_i \in \{0,1,2,3\}$. Каждой цифре 3 предшествует по крайней мере одна цифра {0,1} до предыдущей цифры 3 или конца цифр. Также каждому 0 предшествует по крайней мере одна цифра {2,3} до предыдущего нуля или конца цифр [6, 17];
- Двойное последовательное использование регулярной системы: $d_i \in \{0,1,2,3,4,5\}$; позволяет выполнять увеличение и уменьшение со сложностью O(1) с помощью одного изменения цифры [17, 18, 3];
- Безнулевая регулярная: (zeroless regular) $d_i \in \{1,2,3\}$; $w_i = 2^i$. Каждая последовательность цифр имеет следующую форму: (1 | 2 | 12*3)* [2];
- Строго-регулярная (strictly-regular) $d_i \in \{0,1,2\}$; $w_i = 2^i$. [9];
- Каноническая косая (canonical skew) : $d_i \in \{0,1,2\}$, $w_i=(2^{i+1}-1)$ и первая не нулевая цифра может быть 2. Каждая последовательность цифр имеет следующую форму: 0*(ε | 2)(0 | 1), [18, 19];
- Магическая косая (magical skew) $d_i \in \{0,1,2,3,4\}$, $w_i =(2^{i+1}-1)$ [8];
- Регулярная косая (regular skew) $d_i \in \{0,1,2\}$, $w_i =(2^{i+1}-1)$ [10];
- In-place счетчики [11, 14].

Рассмотрим более подробно каждую систему счисления применительно к биномиальной пирамиде.

БИНАРНАЯ СИСТЕМА СЧИСЛЕНИЯ

Классическая бинарная система счисления, содержащая в представлении цифры d_i , значения которых n= $\sum_{i=0}^{\lg n} d_i 2^i$, где $d_i \in \{0,1\}$

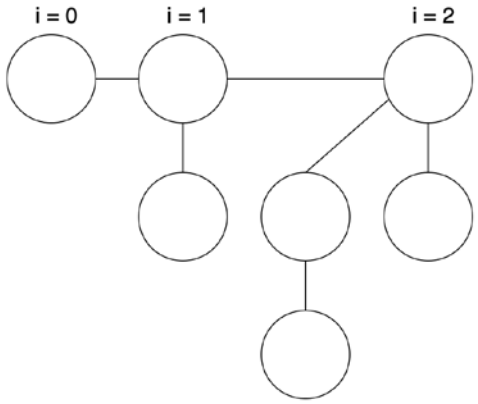

Рис. 2. Пример: $7_{10} = 111_2$. Соответствующая биномиальная пирамида состоит из трех биномиальных деревьев

Операция вставки (Insert) выполняется за время O($log_2 n$) в худшем случае. Возможна ситуация, когда операция выполняется за O(1), если в пирамиде отсутствует дерево ранга 0.

Операция слияния (Merge) выполняется за время O($log_2 n$), и при этом необходимо слияние деревьев одинакового ранга.

Причины высокой сложности в наихудшем случае состоят в том, что увеличение числа может привести к каскадному переносу, а уменьшение сразу после увеличения может привести к каскадному заимствованию. Следствием является то, что вставка узла может привести к последовательности слияний деревьев и удаление узла может привести к последовательности разбиений и слияний деревьев. Чередующаяся последовательность вставки и удаления узлов приводит к последовательности операций, все из которых имеют сложность O($log_2 n$). Все остальные операции также имеют сложность O($log_2 n$).

Решить проблему дорогостоящих чередующихся операций увеличения и уменьшения можно использованием дополнительной (избыточной) цифры в представлении.

ИЗБЫТОЧНАЯ БИНАРНАЯ СИСТЕМА СЧИСЛЕНИЯ

Система счисления называется *избыточной* (redundant), если некоторые числа могут быть представлены более, чем одним способом. Например, можно получить избыточную систему бинарного счисления, взяв $w_i = 2^i$ и $d_i = \{0, 1, 2\}$. Тогда десятичное число 13 можно будет записать как 1011, 1201 или 122. Мы запрещаем нули в конце числа, поскольку иначе почти все системы счисления будут тривиально избыточны.

Использование дополнительной (избыточной) цифры в представлении может решить проблему дорогостоящих чередующихся операций увеличения и уменьшения. Наличие избыточного представления позволяет иметь значение (d)= значению (d′), используя две разные строки. Таким образом, использование цифр 0, 1, 2 позволит избежать попадания в одну и ту же строку цифр при увеличении и уменьшении числа, тем самым избегая каскадной последовательности переносов и заимствований.

В статье Д. Кнута и Дж. Кленси [6] указывается, что представление чисел в избыточной системе необходимо для выполнения операции Insert за время O(1).

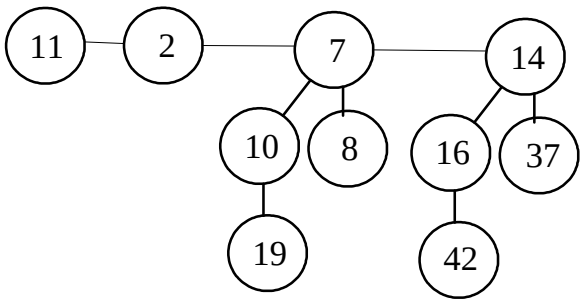

Рис. 3. Биномиальная пирамида с использованием избыточного бинарного представления d = ⟨202⟩

Чтобы получить все варианты представления числа в избыточной бинарной системе счисления необходимо:

1. Представить число в бинарной системе счисления

Таблица 1

…10…	…02…

2. Для представления числа в избыточной системе счисления воспользуемся таблицей 1. Найти в представлении числа из левого столбца таблицы 1, если они существуют, и заменить их на соответствующие элементы из правого столбца таблицы 1 [6, стр. 54]
3. Для получения всех избыточных чисел повторить шаг 2 до тех пор, пока это возможно. Все полученные в результате варианты (в том числе и изначальный) будут представлениями числа в избыточной бинарной системе счисления.

Например, число 10_{10} в избыточной бинарной системе счисления может быть представлено четырьмя способами: 1010, 1002, 210, 202 (рис. 4).

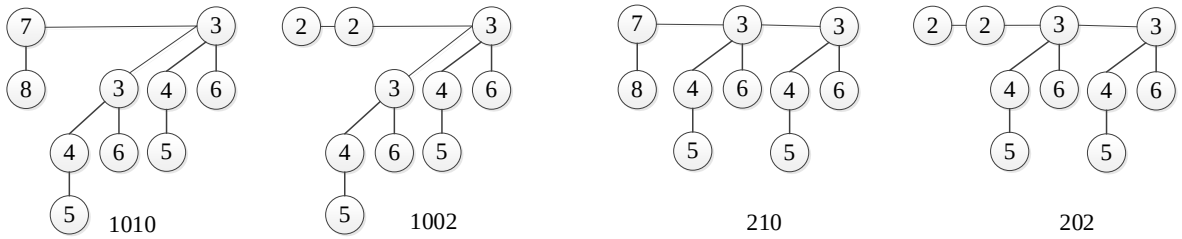

Рис. 4. Варианты представления числа 10 в избыточной системе счисления и соответствующие биномиальные пирамиды

При операции Insert, если имеются более двух деревьев одинакового ранга, сливаются два последних дерева наименьшего ранга

Для последующих операций Merge, FindMin, DeleteMin, Decrease, Delete в избыточной бинарной системе счисления минимальную сложность в наихудшем случае обеспечивает представление, не содержащее в себе цифр 2, что дает дополнительный элемент в корневом списке, т.е. представление в бинарной системе счисления. Данные операции выполняются аналогично операциям в бинарной системе счисления и имеют сложность O($log_2 n$) в худшем случае.

К достоинствам можно отнести:

- Операция Insert выполняется за время O(1);
- Цифра 2 в представлении избавляет от необходимости сливать все деревья одинакового ранга для тех операций, где она используется: Merge, DeleteMin, Delete. Одинаковые деревья сливаются лишь до тех пор, пока их не останется два или менее.

К недостаткам следует отнести отсутствие единства представления. Может потребоваться дополнительное время на перевод из одного представления в другое для уменьшения времени выполнения операции, если это необходимо.

Безнулевая бинарная система счисления

Безнулевые бинарные числа строятся из единиц и двоек, т.е. представление числа содержит цифры $d_i \in \{1,2\}$, значение которых $w_i = 2^i$. Так, например, десятичное число 16 можно записать как 2111 вместо 00001. Основное предназначение данного представления – списочные операции head (получить голову списка) и tail (получить хвост/последний элемент списка). Эти операции будут выполняться за время O(1). 0 в записи числа дает нам нулевой указатель(*null), соответственно, отсутствие нулей в записи числа избавляет нас от проверки на нулевой указатель, что позволяет сразу получить голову или конец списка. Запись числа в этой системе счисления является единственной.

Функция добавления единицы на безнулевых бинарных числах реализуется так:

Чтобы получить представление числа в безнулевой бинарной системе счисления необходимо:

1. Представить число в бинарной системе счисления
2. Найти в представлении элементы из левого столбца таблицы 2, если они существуют, и заменить их на соответствующие элементы из правого столбца.
3. Повторить шаг 2 до тех пор, пока это возможно.
4. Последний вариант, в котором будет невозможно что-либо заменить и будет представлением числа в безнулевой бинарной системе.

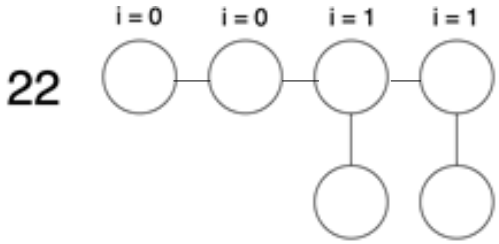

Рис. 5. Биномиальная пирамида в безнулевой системе счисления, соответствующая числу 610

Таблица 2

…10…	…02…
…02…	…10…
…20…	…12…

Для выполнения операции Insert в объединенном корневом списке необходимо слить два последних из трех деревьев одинакового ранга, если такие имеются.

Сложность операции составляет O($log_2 n - 1$) в худшем случае.

Операция Merge выполняется за время O($2*log_2 n - 2$) и у нее в объединенном корневом списке, если имеются более двух деревьев одинакового ранга, сливаются два последних дерева.

Также поддерживаются операции FindMin за время O($2*log_2 n - 2$) в худшем случае и DeleteMin, Decrease и Delete за время O($log_2 n$) в худшем случае.

Достоинства безнулевой бинарной системы счисления - цифра 2 в представлении избавляет от необходимости сливать все деревья одинакового ранга для тех операций, где оно используется: Merge, DeleteMin, Delete. Одинаковые деревья сливаются лишь до тех пор, пока их не останется одно или два.

Недостаток - максимальная длина корневого списка $2(log_2 n - 1)$, что дает высокую сложность операций, зависящих от него.

Избыточное безнулевое представление

Избыточное безнулевое представление объединяет возможности избыточного и безнулевого представлений.

Безнулевое представление: n=$\sum_{i=0}^{k} d_i 2^i$, где k ∈ {−1, 0, . . . , ⌊lg n⌋} и d_i∈ {1, 2, 3, 4} для всех i ∈{0, . . . , k }. Если k = −1, то пирамида пуста.

- Сложение n + 1 соответствует вставке узла в биномиальную пирамиду
- Вычитание n–1 соответствует удалению узла из биномиальной пирамиды

Когда используется стек, чтобы обеспечить функциональные возможности в контексте биномиальных пирамид, вставка и извлечение может быть выполнена при постоянной сложности для наихудшего случая.

Например, число 10 может быть представлено следующими способами: 34, 122, 42. (рис.7)

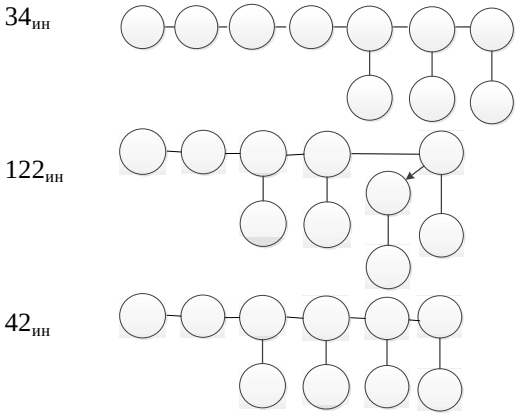

Рис. 7. Избыточное безнулевое представление числа 10

Данное представление обладает преимуществами избыточного бинарного и безнулевого представлений:

1) Для некоторых пирамид, числовую запись которых можно представить несколькими способами, операция insert работает за O(1);
2) При объединении некоторых пирамид не нужно лишний раз сливать деревья, т.к. может быть до 4 одинаковых деревьев;
3) списочные операции head, tail и cons выполняются за O(1).

Выполнение операций осуществляется следующим образом:

1. операция merge (слияния) пирамид объединяет корневые списки пирамид в один список, в котором корни упорядочиваются по рангу узла. Далее проходим от начала до конца списка и сливаем два последних одинаковых по рангу дерева. В среднем в 37.5% случаев не нужно сливать деревья;
2. insert – вставка элемента. Создаётся биномиальная пирамида с единственным узлом, содержащим этот элемент, и объединяется с исходной биномиальной пирамидой. Сложность O(log n) в худшем случае, O(1) – в лучшем (вероятность – 75%, т.к. последняя цифра с вероятностью 75% будет равно 1, 2 или 3);
3. findMin – получение минимума. Пройти по корневому списку и найти минимум. Сложность O(log n);
4. deleteMin – удаление минимального элемента. Удаляем его из корневого списка. Список его потомков включаем в корневой список исходной пирамиды. Сложность O(log n);
5. decrease – уменьшить значение в данной вершине. Уменьшаем значение. Если свойство пирамиды будет нарушено для данной вершины и ее родителя, то меняем их местами. Продолжаем процесс, пока наша вершина не “всплывет” на свое место. Сложность O(log n) в худшем случае;
6. delete – удаление произвольного элемента. Сначала для этого элемента делаем decrease (чтобы он «всплыл» в корень), затем делаем deleteMin. Сложность O(log n).

Регулярные представления.

Ещё одна разновидность бинарных чисел, дающая показатели O(1) для ряда операций в худшем случае — регулярные (regular) бинарные числа.

Регулярные бинарные числа объединяют непрерывные последовательности одинаковых цифр в блоки, так что мы можем применить перенос или заимствование к целому блоку за один шаг. Представляем регулярные бинарные числа как список чередующихся блоков из единиц и нулей.

Целое число в каждом блоке представляет длину блока. Добавляем новые блоки к началу списка блоков с помощью вспомогательных функций zeros (нули) и ones (единицы). Эти функции сливают идущие подряд блоки одинаковых цифр и отбрасывают пустые блоки. Кроме того, zeros отбрасывает нули в конце записи числа.

Заметим, что увеличение бинарного числа требует k шагов, когда число начинается с последовательности в k единиц. Подобным образом, уменьшение бинарного числа требует k шагов, когда число начинается с k нулей.

Регулярная система [6], называемая сегментированной системой [4], состоит из цифр {0,1,2} с тем ограничением, что каждая цифра 2 стоит за 0, между которыми может быть любое количество 1. Используя синтаксис регулярных выражений (для примера смотри [2, раздел 3.3]), каждое регулярное выражение имеет вид (0 | 1 | 01*2). Регулярная система позволяет увеличивать любое число за O(1) замен цифр [19, 6], этот факт может использоваться для того, чтобы видоизменить биноминальные пирамиды так, чтобы добиться выполнения *вставки* за O(1) в худшем случае. Числовое представление может иметь ограничения, которые ограничивают форму строки. Одним из способов выражения этих ограничений является использование синтаксиса регулярных выражений.

Теперь при увеличении регулярного бинарного числа смотрим на первый блок цифр (при условии, конечно, что он вообще есть). Если первый блок содержит нули, то заменяем первый из этих нулей на единицу, создавая новый единичный блок единиц, а длину блока нулей уменьшая на один. Если же первый блок содержит r единиц, то за один шаг проделываем *i* переносов, заменяя единицы на нули, и увеличивая следующую цифру.

Рекурсивный вызов процедуры «увеличить» не зацикливается, поскольку если следующий блок присутствует, он будет содержать нули. Вспомогательная функция позаботится об особом случае, когда первый блок содержит единственный ноль.

Уменьшение регулярного бинарного числа выглядит почти точно так же, только роли единиц и нулей меняются. Здесь тоже рекурсивный вызов не зацикливается, потому что в следующем блоке должны быть единицы.

К сожалению, несмотря на то, что регулярные бинарные числа поддерживают операции увеличение и уменьшение за время O(1) в худшем случае, числовые представления, основанные на них, оказываются слишком сложными, чтобы иметь какое-либо практическое значение. Проблема заключается в том, что понятие перевода целого блока единиц в нули и наоборот плохо переводится на язык операций с деревьями. Более практичные решения, однако, можно получить, если сочетать регулярность с избыточными бинарными числами. При этом можно снова обрабатывать цифры (а следовательно, и деревья) по одной. Регулярность позволяет нам обрабатывать цифры в середине последовательности, а не только в начале.

Рассмотрим, например, избыточное представление, в котором блоки единиц рассматриваются как единый блок. Определяем вспомогательную функцию ones, обрабатывающую блоки, идущие друг за другом, и уничтожающую пустые блоки.

Полезно рассматривать цифру 2 (два) как незаконченный перенос. Чтобы не было каскадов переносов, нам надо гарантировать, что две двойки никогда не идут подряд. Будем поддерживать свойство, что последняя не равная единице цифра перед каждой двойкой равна нулю. Это свойство можно записать как регулярное выражение **(0|1|01*2)*** либо, если ещё учесть отсутствие нулей в конце, **(0*1|0+ 1*2)*.** Заметим, что двойка никогда не является первой цифрой. Таким образом, можно увеличить число на единицу за время O(1) в худшем случае, просто увеличивая первую цифру.

Избыточная регулярная система счисления

Избыточной регулярной системой счисления называется система, в которой $d_i \in \{0,1,2\}$, $w_i = 2^i$, Каждая последовательность цифр имеет следующую форму (0 | 1 | 01*2)*. [6]

Последовательное использование избыточной и регулярной систем позволяет выполнять операции слияния и уменьшения за постоянное время в худшем случае. [3]. Другое последовательное использование избыточной и регулярной систем - чисто функциональный связный дек с постоянной сложностью в наихудшем случае. [16].

Представление числа в избыточной регулярной бинарной системе счисления с соблюдением условия регулярности (между двумя цифрами 2 в представлении числа должен быть как минимум один 0) обеспечивает для операции Insert время O(1).

Избыточная регулярная система счисления может использоваться, чтобы избежать сложности $O(\log_2 n)$ одного приращения. Подстрока вида 01*2 называется блоком. В качестве альтернативы регулярное представление также может быть определено с использованием более свободного ограничения, утверждающего, что между каждыми цифрами 2 существует ноль или более формально заявленных $d_i=2$ и $d_k=2$ и $i<k$, тогда $d_j=0$ для некоторых $j \in \{i+1,...,k-1\}$. Ограничения регулярности гарантируют, что приращение может быть выполнено со сложностью O(1) в наихудшем случае. Следовательно, если регулярное представление используется применительно к биномиальным пирамидам, операция вставки может выполняться со сложностью O(1) в худшем случае.

В регулярном представлении каждая цифра 2 соответствует отложенному переносу, и поддержание регулярности строки может быть выполнено с использованием исправления. Исправление устанавливает $d_i \leftarrow d_i-2$ и $d_{i+1} \leftarrow d_{i+1}+1$. Поддержание регулярности строки в связи с каждым увеличением выполняется путём фиксации ближайшего переноса с задержкой выше, чем i в последовательности переносов (последовательность переносов, упорядоченная по индексу цифр i, с которыми связаны переносы).

Задержанные переносы могут поддерживаться по правилу «первым пришёл – последним вышел», когда приращение должно поддерживаться с последней значащей цифрой.

Регулярная система счисления также может использоваться для поддержки приращения в произвольной цифре строки [6].

Для поддержки как увеличения, так и уменьшения при сложности O(1) в худшем случае используется расширенная регулярная система

Расширенная регулярная система (extended regular).

Чтобы выполнять уменьшение со сложностью O(1) числом замен цифр, было предложено расширение в [19, 6]. Такая расширенная-регулярная система состоит из {0, 1, 2, 3} с тем ограничением, что каждая цифра 3 стоит за 0 или 1, между которыми [3 и 0/1] может быть любое количество цифр 2, что каждая цифра 0 стоит за 2 или 3, между которыми [0 и 2/3] может быть любое количество цифр 1. Примеры структур, использующих расширенную-регулярную систему приведены в [6, 10].

n= $\sum_{i=0}^{\lg n} d_i 2^i$, где $d_i \in \{0,1,2,3\}$. Расширяется число цифр и появляется возможность рассматривать больше регулярных блоков. По-прежнему, d_i число деревьев ранга i.

Блоки: 12*3 02*3 21*0 31*0,

0 и 3 являются отличительными цифрами блоков

fix-carry: 3 x → 1 (x+1) (нет изменений в значении)

Пример: d = 1222232221031121101

increment (d, 2) : d = 123223222103112110**1** ???

fix -carry (d, 5) : d = 12322**13**221031121101

increment (d, 4) : d = 1232**3**13221031121101 ???

fix -carry (d, 4) : d = 1232**12**3221031121101

fix -borrow : 0 x → 2 (x − 1) (значение не меняется)

decrement (d, 15) : d = 12321232210311**2**0101 ???

fix -borrow (d, 17) : d = 123212322103112012

decrement (d, 11), decrement (d, 11), decrement (d, 11) :

d = 123212322100112012

fix -borrow (d, 11) : d = 123212322102012012

Процедура fix-carry соответствует объединению b объектов ранга j в один объект ранга j +1.

Процедура fix-borrow соответствует разделению объекта a_n ранга j на b объектов ранга j − 1

Каждой цифре 3 предшествует по крайней мере одна из цифр {0,1} до цифры 3 или конца цифр. Также каждому 0 предшествует по крайней мере одна из цифр {2,3} до предыдущего нуля или конца цифр. [6, 17]

При реализации:

- у каждой цифры есть указатель на отличительную цифру своего блока;
- если цифра находится не в блоке, ее указатель указывает на произвольную цифру.

Это работает потому, что:

1. Ненужный fix не вредит (когда блок уничтожается, нет необходимости менять указатели)
2. Блок расширяется на одну цифру спереди.
3. Вновь созданный блок состоит из двух цифр.

Безнулевая регулярная система счисления.

Бродал [2] использовал “не содержащий нуля” вариант регулярной системы, включающий $d_i \in \{1,2,3\}$; $w_i=2^i$. для того, чтобы гарантировать то, что размеры его деревьев экспоненциальные в отношении их рангов. Для дополнительных примеров использования регулярных систем смотри [8,9]. Каждая последовательность цифр имеет следующую форму (1 | 2 | 12*3)* [2].

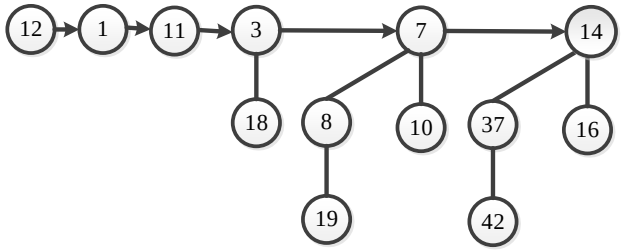

Рис. 8. Биномиальная очередь с использованием ненулевого регулярного представления d= ⟨312⟩

Безнулевая система счисления гарантирует, что число поддеревьев, имеющих одинаковый ранг, находятся в диапазоне {1, 2, 3}, что гарантирует, что размеры деревьев экспоненциальные по отношению к их рангам. Все поддеревья в безнулевой системе счисления имеют ранги меньше ранга дерева-родителя. Разрешение одному поддереву иметь ранг меньше ранга дерева-родителя облегчает достижение сложности O(1) в наихудшем случае для операции слияния. Сложность сравнений для операций удаления и удаления минимума в пирамидах, описанных в [[2] составляет 7lgn+O(1)

В качестве основного применения этой системы счисления, реализация эффективной сливаемой пирамиды. Наилучший верхний предел это 2 lg n + O(1) элементарных сравнений для удаления, что достигается изменением пирамиды, описанной в [2].

Строго регулярная бинарная система счисления

Система, в представлении которой соблюдается условие регулярности: если $d_i = 2$ и $d_k = 2$ и $i < k$, тогда $d_j = 0$ для некоторых $j \in \{i+1, \dots, k-1\}$ (между двумя цифрами 2 в представлении числа в строго-регулярной системе счисления должен быть хотя бы один ноль). Представление содержит цифры $d_i \in \{0,1,2\}$, значения которых $w_i=2^i$. Последовательность от наименее значимой до самой значимой цифры имеет вид $(1^+ \mid 01{*}2)^*(\varepsilon \mid 01^+)$

Экстремальные цифры: 0 и 2. Данная система является частным случаем избыточной системы.

Пример увеличения

Обозначения: цифра $\boldsymbol{d_i}$, которая будет увеличиваться отображается **более жирно**. d_a – первая экстремальная цифра после d_i (обозначается курсивом), k – неотрицательное целое, a обозначает любую комбинацию 1^+ и 01*2 блоков, и ω обозначает любую комбинацию 1^+ и 01*2 блоков, за которыми следует не более одного блока 01^+

Начальная конфигурация: α01***1**1*21^kω

Действие: $d_i \leftarrow 2$; $d_a \leftarrow d_a - 2$; $d_{a+1} \leftarrow d_{a+1}+1$

Финальная конфигурация: α01***2**1*02^kω

Примечание: Это один из 19 случаев, рассмотренных в работе [9]

Чтобы получить представление числа в строго-регулярной системе счисления, необходимо:

1. Представить число в бинарной системе счисления
2. Найти в представлении элементы из левого столбца таблицы 3, если они существуют, и заменить их на соответствующие элементы из правого столбца, при этом следя, чтобы условие регулярности не нарушалось.

3. Повторить шаг 2 до тех пор, пока это возможно.
4. Последний вариант, в котором будет невозможно что-либо заменить и будет представлением числа в строго-регулярной бинарной системе.

Таблица 3

…10…	…02…
…20…	…12…

Свойства:

- увеличение, уменьшение, объединение и разделение включают O(1) обмен цифр в худшем случае
- Сложение 2-х k-значных чисел включает не более k заимствований
- Сумма цифр k-значного числа равна как минимум φ^k-1 где φ – золотое сечение (golden ratio) (свойство экспоненциальности)

Сливаемые пирамиды с использованием строго регулярной системы счисления имеют следующие ограничения: Операции поиска минимума, вставки, слияния выполняется так же, как в избыточной бинарной системе счисления, и их сложность составляет O(1) в худшем случае. Операция удаления имеет сложность O(lg n) в худшем случае, где n размер структуры данных. Количество сравнений элементов при этом составляет 2lgn + O(1).

Алгоритмы вышеперечисленных операций описаны в [9].

К достоинствам можно отнести:

- Операция Insert выполняется за время O(1)

К недостаткам следует отнести сложные в реализации алгоритмы для операций Merge, DeleteMin и Delete из-за необходимости соблюдения условия регулярности.

Косые системы счисления.

Косое бинарное исчисление на практике обычно приводит к более простым и быстрым программам, однако этот метод связан с более радикальным отходом от обыкновенных бинарных чисел.

Компьютерные представления позиционных систем счисления могут быть *плотными* (dense) или *разреженными* (sparse). Плотное представление — это просто список (или какая-то другая последовательность) цифр, включая нули. Напротив, при разрежённом представлении нули пропускаются. В таком случае требуется хранить информацию либо о ранге (то есть индексе), либо о весе каждой не нулевой цифры.

Для косых бинарных чисел нельзя использовать плотное представление, потому что тогда поиск первой ненулевой цифры займёт больше времени, чем O(1). Поэтому выбираем разрежённое представление, благодаря чему всегда имеем непосредственный доступ к младшей ненулевой цифре.

Целые числа представляют либо ранг, либо вес не нулевых цифр. Пока используем веса. Веса хранятся в порядке возрастания, но два наименьших веса могут быть одинаковы, показывая, что младшая ненулевая цифра равна двум. Уменьшение косого бинарного числа на единицу столь же просто, как увеличение. Если младшая цифра не равна нулю, просто уменьшаем эту цифру с двух до единицы или с единицы до нуля. В противном случае уменьшаем самую младшую ненулевую цифру, а предыдущий ноль заменяем двойкой.

Ясно, что уменьшение также работает за время O(1) в худшем случае.

Каноническая косая система счисления

Каноническая косая система счисления [18, 19] называется косой потому, что значения цифр $d_i \in \{0,1,2\}$ в представлении равны w^i=d_i $*(2^{i+1}-1)$. Веса деревьев в бинарном представлении (2^i): 1, 2, 4, 8, 16, 32, … Веса деревьев в косом бинарном представлении ($2^{i+1}-1$): 1, 3, 7, 15, 31, …

Косая система счисления избыточна, однако, можно вернуть уникальность представления, если ввести дополнительное требование, что лишь самая младшая не нулевая цифра может быть двойкой. Будем говорить, что такое число записано в *каноническом виде* (canonical form). Каноничность и единственность представления определяется следующими ограничениями:

1. Не более одной цифры 2 в представлении числа
2. Если цифра 2 присутствует в представлении, то все цифры справа от нее должны быть равны 0

Алгоритм построения биномиальных деревьев в канонической косой системе счисления представлен на рис. 9.

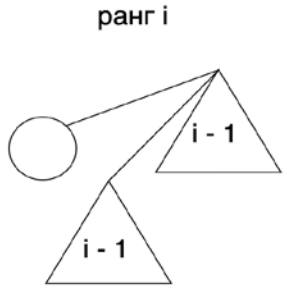

Рис. 9. Косое дерево ранга i

Пример косой биномиальной пирамиды приведен на рис. 10.

Операция Insert выполняется за время O(1). Возможны два случая:

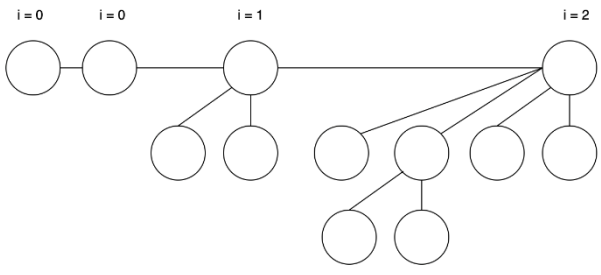

Рис.10. Каноническая косая биномиальная пирамида, соответствующая числу 12_{10}= 112_{skew}

Случай 1: ранги двух наименьших деревьев совпадают. В этом случае производится косое связывание (рис. 11):

1) дерево с большим корнем становится левым потомком дерева с меньшим корнем;
2) К корню получившегося дерева добавляется новый элемент.

Случай 2: ранги двух наименьших деревьев не совпадают. В этом случае создается новое одноэлементное дерево и добавляется к началу корневого списка.

Операция DeleteMin выполняется по следующему алгоритму:

1. Удаляется минимальный элемент из корневого списка, и список потомков объединяется с корневым;
2. Если существует хотя бы одно дерево ранга 0, то проходя по корневому списку справа налево необходимо слить 2 наибольших дерева одинакового ранга, если они существуют, и дерево ранга 0 в одно дерево;
3. Повторить шаг 2 до тех пор, пока это возможно.

К достоинствам можно отнести:

- Операция Insert выполняется за время O(1);
- Высота дерева максимального ранга меньше на 1 в сравнении с не косыми системами, в которых значения цифр $w_i = 2^i$.

К недостаткам следует отнести сложные в реализации алгоритмы операции Merge и операций, где Merge используется (DeleteMin и Delete); из-за нестандартных значений цифр w_i в представлении.

Магическая косая система счисления

Магическая косая система счисления [8] по своей сути является избыточной, т.к. содержит пять цифр $d_i \in \{0, 1, 2, 3, 4\}$, значения которых $w_i = (2^{i+1} - 1)$. Однако операционное определение магической косой системы и способ выполнения операций гарантируют уникальное представление для любого целого числа.

Экстремальные цифры: di ∈ {0, 1, 3, 4}

Нижние цифры: di ∈ {0, 1}

Верхние(большие) цифры: di ∈ {3, 4}

Операция Insert выполняется в худшем случае за O(1) по следующему алгоритму:

1. Добавить элемент в корневой список и проходя по нему слева направо необходимо найти первые три или четыре дерева одинакового произвольного ранга j.
2. Если такие имеются – необходимо слить два последних из них и дерево ранга 0 в одно дерево, ранг которого будет равен j+1.
3. Если ранг j+1 полученного дерева не равен 1, то последнее из деревьев ранга j необходимо представить как список двух деревьев ранга j-1 и дерева ранга 0.

Операции FindMin и Insert выполняется за время O(1). Операция delete имеет сложность O(lgn), количество сравнений элементов при этом 6 lgn + O(1) в худшем случае. Алгоритмы для этих и остальных операций представлены и подробно описаны в [10].

К достоинствам можно отнести:

- Insert выполняется за время O(1)
- Высота дерева максимального ранга меньше на 1 в сравнение с системами, в которых значения цифр $w_i = 2^i$.

К недостаткам следует отнести сложность в реализации алгоритмов операций Merge, DeleteMin и Delete из-за пяти возможных цифр d_i в представлении и их значений w_i

Регулярная косая система счисления

система, в которой комбинируются свойства строго-регулярной и канонической косой систем: соблюдается условие регулярности и значения $w_i = (2^{i+1} - 1)$ для цифр $d_i \in \{0, 1, 2\}$ (между двумя цифрами 2 в записи числа в регулярном косом представлении должен быть хотя бы один ноль) [10].

Используя регулярную косую систему для облегчения вставки и объединения пирамид на основе указателей, сохраняем следующие ограничения:

Операции FindMin и Insert имеют сложность O(1) в худшем случае. Операция delete имеет сложность O(lgn), количество сравнений элементов при этом 5 lgn + O(1) в худшем случае. Операция merge имеет сложность $O(\lg^2 m)$, где m размер меньшей сливаемой пирамиды.

Имеются 2 следствия, определяющих количество цифр в записи числа и сумму цифр в записи числа:

Следствие 1: количество цифр в записи числа n, представленной в регулярной косой системе, составляет не более $log_2(n) - 1$. Это значит, что дерево максимальной высоты будет иметь ранг, не превышающий
$log_2(n) - 2$. Это дает нам максимальную высоту дерева $log_2(n) - 2$

Следствие 2: сумма цифр в записи числа, представленной в регулярной косой системе, не превышает $log_2(n) + 1$, что дает нам длину корневого списка в худшем случае $log_2(n) + 1$

Операция Insert и Merge выполняются за время O(1) и O($log_2^2(n)$) в худшем случае соответственно. Также данная система счисления поддерживает операции FindMin за время O($log_2(n) + 1$) и Decrease за O($log_2(n) - 1$).

Более подробно вышеперечисленные операции описаны в [10]. К достоинствам можно отнести:

- Операция Insert выполняется за время O(1)
- Высота дерева максимального ранга меньше на 1 в сравнение с системами, в которых значения цифр $w_i = d_i * 2^i$.

К недостаткам следует отнести - сложные в практическом плане алгоритмы для операций Insert, Merge, DeleteMin и Delete из-за необходимости в соблюдении условия регулярности и нестандартных значений w_i в представлении.

Структуры данных, которые можно описать как числовые представления, встречаются па удивление часто, но явным образом связь с каким-либо вариантом системы счисления упоминают лишь изредка [15, 18, 5]. Косые биномиальные пирамиды описаны в [4].

Заключение

Подводя итог исследованию многочисленных публикаций по этому направлению, можно говорить об актуальности и эффективности этих работ.

Стандартным биномиальным пирамидам, как впрочем и некоторым другим структурам, свойственны каскадные вычисления при добавлении или удалении элементов, что существенно замедляет такие базовые операции как вставка и слияние и зависящие от них другие операции. Если в обычной биномиальной пирамиде вставка выполняется со сложностью O(lgn) в худшем случае, то в избыточной бинарной системе счисления вставка выполняется со сложностью O(1) в худшем случае. Платой за это является время на преобразование структуры из бинарной системы в избыточную бинарную систему счисления.

Чтобы выполнить списочные операции head (получить голову списка) и tail (получить хвост списка) со сложностью O(1) используется безнулевая бинарная система счисления.

Возможности избыточной и безнулевой систем счисления объединяются в избыточную, безнулевую систему счисления.

Наличие регулярных непрерывных одинаковых цифр позволяет их объединить в блоки. Последовательное использование избыточных и регулярных систем счисления позволяет выполнять операции слияния и уменьшения за постоянное время в худшем случае.

Регулярные системы счисления могут также использоваться для поддержки приращения в произвольной цифре строки. Для поддержки как увеличения так и уменьшения со сложностью O(1) в худшем случае используется расширенная регулярная система счисления.

Безнулевая регулярная система счисления позволяет получить верхний предел количества сравнений при удалении 2lgn+O(1) вместо 7lgn+O(1) в обычной биномиальной пирамиде.

Использование строго регулярной бинарной системы счисления применительно к биномиальным пирамидам позволяет выполнять операции поиска минимума, вставки, слияния со сложностью O(1) в худшем случае. Количество сравнений при удалении составляет 2lgn+O(1).

К недостаткам всех регулярных систем следует отнести необходимость соблюдения условий регулярности.

Косое бинарное исчисление на практике приводит к более простым и быстрым программам. Любая косая система избыточна. Чтобы вернуть уникальность путём ограничений создаётся каноническая косая система счисления. При определённых её достоинствах, операция слияния здесь более сложная.

Альтернативой является магическая косая система счисления за счёт увеличения количества деревьев одного ранга.

Регулярная косая система счисления комбинирует свойства строго регулярной и канонической косой систем.

Как видно, эффективность применения различных систем счисления зависит от конкретных алгоритмов, точнее от выполняемой последовательности операций в этих алгоритмах.

Структуры данных, которые можно описать как числовые представления, встречаются па удивление часто, но явным образом связь с каким-либо вариантом системы счисления упоминают лишь изредка [15, 18, 5]. Косые биномиальные пирамиды описаны в [4].

Этот обзор интересен и тем, что даёт возможность задуматься о дальнейшей работе в этом направлении. Использование различных систем счисления на примере биномиальных пирамид не исключает их применения на других видах пирамидальных структур (фибоначчиевы пирамиды, парные пирамиды и другие).

Библиография:

1. Гулаков В. К., Гулаков К. В. Монопирамидальные структуры данных. – М.: Горячая линия –Телеком, 2019. –148 с.: ил.
2. Brodal, G. Fast Meldable Priority Queues. WADS 1995: 282-290
3. Brodal, G. S. Worst-case efficient priority queues, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM/SIAM (1996), 52-58.
4. Brodal, G. S. Okasaki, C. Optimal purely functional priority queues, Journal of Functional Programming 6, 6 (1996), 839-857.
5. Carlsson, S. Munro, J. I. Poblete, P. V. An implicit binomial queue with constant insertion time, Proceedings of the 1st Scandinavian Workshop on Algorithm Theory,Lecture Notes in Computer Science 318, Springer-Verlag (1988), 1-13.
6. Clancy, M. J. A programming and problem-solving seminar / M. J. Clancy, D. E. Knuth // Technical Report STAN-CS-77-606, Department of Computer Science, Stanford University, APRIL 1977
7. Elmasry, A.. On the power of structural violations in priority queues. / A. Elmasry, C. Jensen, J. Katajainen. // In Proc. 13th Computing: The Australasian Theory Symposium., volume 65 of CRPIT, pages 45–53. Australian Computer Society, 2007.
8. Elmasry, A. The Magic of a Number System / Amr Elmasry, Claus Jensen, and Jyrki Katajainen from book Fun with Algorithms: 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings (pp.156-165)

9. Elmasry, A. Strictly-regular number system and data structures / A. Elmasry, C. Jensen, J. Katajainen // 12th Scandinavian Symposium and Workshops on Algorithm Theory (2010), Bergen, Norway, LNCS 6139, 26–37.
10. Elmasry, A. Two skew-binary numeral systems and one application/ Amr Elmasry, Claus Jensen, Jyrki Katajainen // Published in Theory of Computing Systems 2011
11. Elmasry, A., Jensen, C., Katajainen, J.: Two-tier relaxed heaps. Acta Informatica 45(3), 193–210 (2008)
12. Elmasry, A. Katajainen, J. Worst-case optimal priority queues via extended regular counters, 7th International Computer Science Symposium in Russia (2012), Nizhny Novgorod, Russia, LNCS 7353, 130–142.
13. Elmasry, A. Improving the efficiency of priority-queue structures / Amr Elmasry, Claus Jensen, Jyrki Katajainen // University of Copenhagen, Denmark, 23 October 2012
14. Elmasry and J. Katajainen, In-place binary counters, 38th Inter-national Symposium on Mathematical Foundations of Computer Science (2013), Klosterneuburg, Austria, LNCS 8087, 349–360
15. Elmasry, A. Number Systems and Data Structures, Alexandria University https://ru.scribd.com/document/175364781/Numeral-Systems-and-Data-Structures
16. Guibas, L. J. Edward M. McCreight, Michael F. Plass, Janet R. Roberts: A New Representation for Linear Lists. STOC 1977: 49-60
17. Haim Kaplan, Robert Endre Tarjan: Persistent lists with catenation via recursive slow-down. STOC 1995: 93-102
18. Kaplan, H., Tarjan, R.E.: Purely functional representations of catenable sorted lists. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 202–211. ACM, New York (1996)
19. Kaplan, H. Nira Shafrir, Robert Endre Tarjan: Meldable heaps and boolean union-find. STOC 2002: 573-582
20. Eugene W. Myers. An applicative random-access stack. *Information Processing Letters*, 17(5):241–248, December 1983. (pp. 76, 82, 84)
21. Okasaki, C. Purely Functional Data Structures /Dissertation for the degree of Doctor of Philosophy. CMU-CS-96-177, September 1996
22. Vuillemin J. A unifying look at data structures // Communications of the ACM. 1980. 23. P. 229–239.

Сведения об авторах:

Гулаков Василий Константинович
кандидат технических наук
Брянский государственный технический университет
профессор кафедры Информатика и программное обеспечение, (БГТУ)
Эл. почта: gvk10@yandex.ru
Телефон: +79103328612

Гулаков Константин Васильевич
кандидат технических наук
Брянский государственный технический университет
доцент кафедры Информатика и программное обеспечение, (БГТУ)
Эл. почта: gulakov32@yandex.ru
Телефон: +79103386234

Overview of binomial pyramids based on different number systems

Vasiliy K. Gulakov, Konstantin V. Gulakov

***Abstract* - The article provides an overview of binomial data structures based on various number systems. This approach improves the performance of operations on binomial pyramids, such as increasing and decreasing the number of elements, inserting elements, merging pyramids. Various number systems are considered: binary, redundant, nonzero, regular, oblique, and their combinations. The structure of binomial pyramids is well described by a binary number equal to the number of elements in the pyramid. By working with these numbers, you can efficiently perform various operations on pyramidal structures. The problem with representing a binary number is that increasing or decreasing the number by one can cause many of the digits of the number to cascade, which affects the result in the worst case. In order to get away from this situation, other number systems are used. The article briefly discusses various number systems, their combinations and their capabilities.**

***Keywords*: trees, binomial pyramid, redundant number systems, algorithm complexity**

REFERENCES

1. Gulakov V. K., Gulakov K. V. Monopiramidal'nye struktury dannyh. – M.: Gorjachaja linija –Telekom, 2019. –148 s.: il.
2. Brodal, G. Fast Meldable Priority Queues. WADS 1995: 282-290
3. Brodal, G. S. Worst-case efficient priority queues, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM/SIAM (1996), 52-58.
4. Brodal, G. S. Okasaki, C. Optimal purely functional priority queues, Journal of Functional Programming 6, 6 (1996), 839-857.
5. Carlsson, S. Munro, J. I. Poblete, P. V. An implicit binomial queue with constant insertion time, Proceedings of the 1st Scandinavian Workshop on Algorithm Theory,Lecture Notes in Computer Science 318, Springer-Verlag (1988), 1-13.
6. Clancy, M. J. A programming and problem-solving seminar / M. J. Clancy, D. E. Knuth // Technical Report STAN-CS-77-606, Department of Computer Science, Stanford University, APRIL 1977
7. Elmasry, A.. On the power of structural violations in priority queues. / A. Elmasry, C. Jensen, J. Katajainen. // In Proc. 13th Computing: The Australasian Theory Symposium., volume 65 of CRPIT, pages 45–53. Australian Computer Society, 2007.
8. Elmasry, A. The Magic of a Number System / Amr Elmasry, Claus Jensen, and Jyrki Katajainen from book Fun with Algorithms: 5th International Conference, FUN 2010, Ischia, Italy, June 2-4, 2010. Proceedings (pp.156-165)
9. Elmasry, A. Strictly-regular number system and data structures / A. Elmasry, C. Jensen, J. Katajainen // 12th Scandinavian Symposium and Workshops on Algorithm Theory (2010), Bergen, Norway, LNCS 6139, 26–37.
10. Elmasry, A. Two skew-binary numeral systems and one application/ Amr Elmasry, Claus Jensen, Jyrki Katajainen // Published in Theory of Computing Systems 2011
11. Elmasry, A., Jensen, C., Katajainen, J.: Two-tier relaxed heaps. Acta Informatica 45(3), 193–210 (2008)
12. Elmasry, A. Katajainen, J. Worst-case optimal priority queues via extended regular counters, 7th International Computer Science Symposium in Russia (2012), Nizhny Novgorod, Russia, LNCS 7353, 130–142.
13. Elmasry, A. Improving the efficiency of priority-queue structures / Amr Elmasry, Claus Jensen, Jyrki Katajainen // University of Copenhagen, Denmark, 23 October 2012
14. Elmasry and J. Katajainen, In-place binary counters, 38th Inter-national Symposium on Mathematical Foundations of Computer Science (2013), Klosterneuburg, Austria, LNCS 8087, 349–360
15. Elmasry, A. Number Systems and Data Structures, Alexandria University https://ru.scribd.com/document/175364781/Numeral-Systems-and-Data-Structures
16. Guibas, L. J. Edward M. McCreight, Michael F. Plass, Janet R. Roberts: A New Representation for Linear Lists. STOC 1977: 49-60
17. Haim Kaplan, Robert Endre Tarjan: Persistent lists with catenation via recursive slow-down. STOC 1995: 93-102
18. Kaplan, H., Tarjan, R.E.: Purely functional representations of catenable sorted lists. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 202–211. ACM, New York (1996)
19. Kaplan, H. Nira Shafrir, Robert Endre Tarjan: Meldable heaps and boolean union-find. STOC 2002: 573-582
20. Eugene W. Myers. An applicative random-access stack. Information Processing Letters, 17(5):241–248, December 1983. (pp. 76, 82, 84)
21. Okasaki, C. Purely Functional Data Structures /Dissertation for the degree of Doctor of Philosophy. CMU-CS-96-177, September 1996
22. Vuillemin J. A unifying look at data structures // Communications of the ACM. 1980. 23. P. 229–239.