УДК 004.422.635.33

А. В. Аксенов – аспирант кафедры вычислительных систем и сетей
И. С. Колосова – студент кафедры вычислительных систем и сетей

АНАЛИЗ ЭФФЕКТИВНОСТИ БАЛАНСИРУЕМЫХ СТРУКТУР ДАННЫХ С ПРИОРИТЕТАМИ ЭЛЕМЕНТОВ

В ряде задач, связанных со статистической обработкой дискретных данных, возникает потребность в структуре данных, обеспечивающей хранение элементов некоторого заранее заданного множества вкупе с некоторой информацией о них. При этом зачастую играет важную роль эффективность выполнения операций с данной структурой (добавление, поиск). Очевидно, что если объем данных достаточно велик, а мощность множества возможных значений величины невелико, наиболее частой операцией, совершаемой со структурой данных, будет поиск некоторого элемента по его ключу. При этом, если предположить, что элементы неравнозначны по востребованности, то возникает необходимость ускорить доступ именно к тем элементам, которые обладают наивысшим приоритетом.

Поставленной задачей является разработка структуры данных, которая позволяет хранить элементы со связанными с ними приоритетами, где приоритет отражает востребованность данного элемента, и минимизация средневзвешенного времени доступа к элементу, где вес – приоритет элемента. При этом приоритет узлов может динамически меняться, и вычисляется как количество раз, когда был осуществлен его поиск.

Фактически, задача сводится к реализации интерфейса ассоциативного массива, обладающего дополнительным свойством ускорения доступа к наиболее востребованным элементам.

К возможным вариантам физической реализации структуры можно отнести массив, хеш-таблицу, связный список, двоичное дерево.

Из перечисленных структур данных только дерево (при условии его сбалансированности) может гарантировать логарифмическую сложность операций поиска и добавления элемента. При этом хеш-таблицы не удовлетворяют другим ограничениям задачи, рассмотрение которых выходит за рамки данной статьи.

Для решения данной задачи необходимо применять балансировку дерева по весу (приоритету), поскольку балансировка по высоте может не дать необходимых результатов. [1]

Распространенной древовидной структурой данных, оперирующей с приоритетами элементов, является декартово дерево. [2]

Декартово дерево хранит пары (X,Y) в виде бинарного дерева таким образом, что она является деревом поиска по X и кучей по Y. Если положить, что Y — приоритет узла, то узлы с большим приоритетом окажутся выше к корню. В данном случае очень часто встречается равенство приоритетов у узлов, поэтому приоритеты узлов-потомков могут быть равны приоритету родителя, несмотря на то, что это нарушает строгое определение кучи.

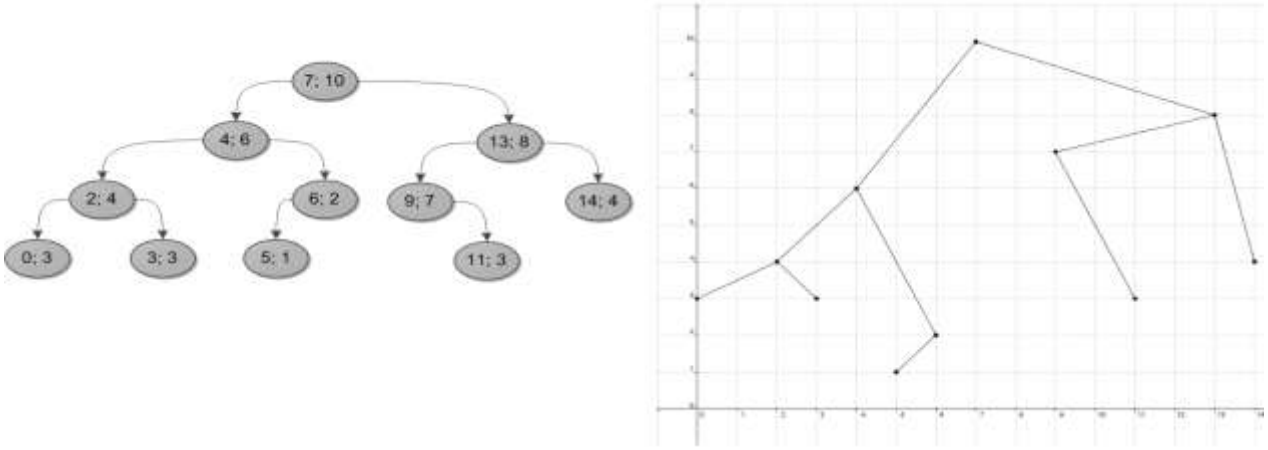

Рис. 1 Представление декартова дерева на декартовой плоскости

Строго говоря, декартово дерево не является балансируемой структурой данных, так как расположение узлов в зависимости от их приоритетов обусловлено свойством кучи.

Реализация декартова дерева может быть основана на малых вращениях или операциях Split и Merge.

При поиске элемент a, если он был найден, то его приоритет увеличивается на 1. После чего происходит проверка условия кучи. Если условие не выполняется, то дерево вращается относительно элемента-родителя. Так повторяется на каждом уровне.

Альтернативная реализация декартова дерева предполагает 2 операции, с помощью которых осуществляются все действия со структурой. Split (T, X) - разделяет дерево T на два дерева T1 и T2 таким образом, что T1 содержит все элементы, меньшие по ключу X, а T2 содержит все элементы, большие X.

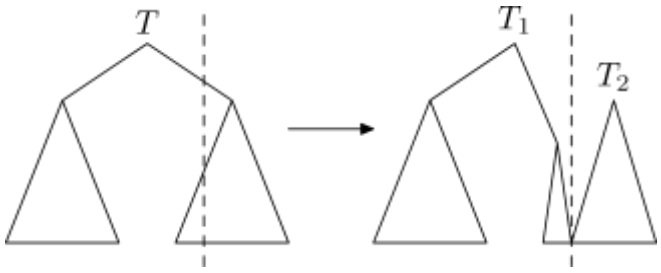

Рис. 2 Операция декартова дерева Split

Merge (T_1, T_2) - объединяет два поддерева T_1 и T_2. Она работает в предположении, что T_1 и T_2 обладают соответствующим порядком (все значения X в первом меньше значений X во втором).

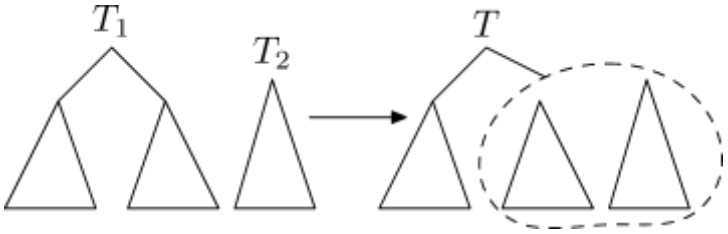

Рис. 3 Операция декартова дерева Merge

При добавлении элемента X, сначала выполняется операция Split(T,X) относительно этого элемента, после чего две последовательные операции Merge (T_1, X) и Merge (T, T_2) объединяют этот элемент с двумя получившимися поддеревьями.

Разрабатываемая структура данных (в дальнейшем – TKOL-дерево) является альтернативой декартовому дереву и осуществляет балансировку по весу.

При выполнении операций с TKOL при каждом проходе вниз по дереву в каждом узле проверяется его сбалансированность. Сбалансированность в данном случае — это модуль разности суммы приоритетов узлов левого поддерева текущего узла и суммы приоритетов узлов правого поддерева. Сравнивается сбалансированность дерева в его текущем состоянии и при его повороте. Если разность этих значений будет положительной, то производится малое вращение. [1] Таким образом, получаем дерево, сбалансированное на каждом уровне. Сама операция вращения не требует много ресурсов, так как только происходит изменение нескольких ссылок.

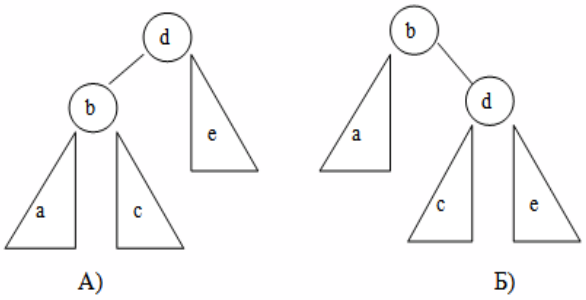

Рис. 4 Малое вращение
А) Структура дерева до вращения
Б) Структура дерева после вращения

Критерий вращения:

где P_i – вес i-го узла или поддерева.

Было произведено теоретическое и практическое сравнение производительности операций с двумя видами рассматриваемых структур данных.

Ниже приводятся результаты теоретического анализа эффективности операций добавления элемента.

Таблица 2

Операция добавления элемента	**Сложность O(n)**
Реализация декартова дерева через операции Split и Merge	6 * log(n)
Реализация декартова дерева через малые вращения	2 * log(n)
TKOL	2 * log(n)

Из таблицы 2 видно, что реализация декартова дерева через малые вращения и TKOL равны по времени выполнения, но оно не учитывает самой операции вращения. Декартово дерево выполняет вращения в разы чаще, чем TKOL, что оказывает негативное влияние на время выполнения операций.

Количество переходов по дереву, которое в среднем придется совершить, чтобы добраться до произвольного узла определяется формулой:

$$T = \frac{P_{left}}{P}(\log N_{left} + 1) + (1 - \frac{P_{left}}{P})(\log(N - N_{left} - 1) + 1),$$

где:

P_{left} - вес левого поддерева;

P - вес всего дерева;

N_{left} - количество узлов в левом поддереве;

N - суммарное количество узлов дерева.

Для оценки эффективности балансировки для каждого распределения весов возьмем худшее время выполнения операций среди всех разбиений по количеству узлов в левом и правом поддеревьях. Проведенные исследования этой зависимости показали, что $T(P_{left})$ минимально, когда процент веса в левом поддереве равен 50%, то есть поддеревья равны по весу. TKOL использует это свойство как критерий, минимизируя таким образом количество переходов до случайного элемента.

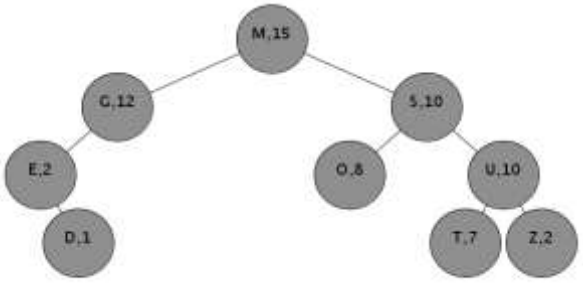

Рис. 5 Пример неэффективного декартова дерева

На рисунке 5 приведено декартово дерево, сгенерированное последовательностью символов «mgesoutzdmgtsgugugmsgutugmsguuggmtsuumsgumotstozmosomosotmoesmmmmm». Из рисунка видно, что, несмотря на выполнение свойства кучи, время доступа к случайному элементу будет не оптимальным. Средневзвешенный путь к элементу в таком дереве AWD_{Treap}=1,9, а если построить на этих же элементах TKOL, то AWD_{TKOL}=1,4.

Анализ эффективности показал ряд преимуществ TKOL перед декартовым деревом, что было подтверждено экспериментально.

Библиографический список

1. Аксенов А.В. Балансировка деревьев статистической модели энтропийного кодировщика, Сборник докладов Научной сессии ГУАП, посвященной Всемирному дню космонавтики, СПб, ГУАП, 2011 г.

2. http://habrahabr.ru/post/101818/ - Полозов А. Декартово дерево: Часть 1. Описание, операции, применения.