АЛГОРИТМЫ ЛОГАРИФМИЧЕСКОЙ ВРЕМЕННОЙ СЛОЖНОСТИ ДЛЯ ПОИСКА В ДЕКАРТОВОМ ДЕРЕВЕ СПЕЦИАЛЬНОГО ВИДА

УО «Белорусский государственный университет информатики и радиоэлектроники»
г. Минск, Республика Беларусь

Гомон С. В., Лапицкий А. В.

Ивашенко В. П., ассистент кафедры ИИТ

При описании обрабатываемых данных зачастую принято использовать абстракцию типа данных. В настоящее время существует значительное количество накопленных типов данных, начиная от простых и заканчивая сложными структурами. Структуры некоторых типов удобно использовать для представления множеств однотипных элементов или элементов различных типов. Как известно множества [1] могут быть ориентированными (вектора, кортежи) и неориентированными. Для широкого представления неориентированных множеств используется любые типы данных, которые удовлетворяют спецификации абстрактного типа «множество», иные типы – в более частных случаях; для ориентированных множеств – используются массивы, списки, двоичные деревья поиска и прочие типы данных. Для структуры данных, представляющей множество, необходимым требованием является наличие определения двух процедур – порождающей (перечисляющей) и разрешающей. Первая позволяет перебрать элементы множества (выбрать следующий элемент), вторая – установить: является ли указанный аргумент элементом множества, либо – нет. Эти процедуры, как и другие, могут различаться пространственной и временной сложностью [2]. Например, для списков [2] существует простая и эффективная реализация порождающей процедуры. Однако для них же разрешающая процедура будет работать за линейное время, что зачастую неприемлемо при обработке множеств большого размера. Более эффективным в этом плане будет применение структуры такого типа как сбалансированное двоичное дерево поиска [3], однако оно потребует больше пространства в памяти для своего хранения. Существуют типы данных, представляющие множества специальных видов – соответствия, функции, биекции, отношения: списки рёбер, ассоциативные массивы и другие.

Рассмотрим реализацию модификации такой структуры данных как декартово дерево. Назовём тип таких структур «сбалансированным декартовым деревом». Сбалансированное декартово дерево позволяет явно представить некоторую функцию [1]. Её область определения состоит из ключей, а область значений – из приоритетов. Отличие от известного декартова дерева в том, что балансировка дерева осуществляется по ключам, а в качестве приоритетов вершин сбалансированного декартова дерева сохраняются величины, вычисленные на основании заданных приоритетов ключей. В качестве принципа балансирования используется принцип дихотомии. В этом случае предполагается, что максимальное и минимальное значение из всех значений типа ключа известно (0 и S). Тогда на множестве всевозможных ключей сбалансированное декартово дерево строится следующим образом: поддеревья каждой вершины хранят ключи из своей половины множества значений ключей дерева, корнем которого является эта вершина, она же хранит ключ, значение которого ближе всего к центральному значению из множества значений ключей этого же дерева. Таким образом, высота дерева ограничена логарифмом по основанию два мощности множества значений ключей. Приоритет каждой вершины вычисляется как максимум из всех приоритетов ключей в дереве, корнем которого является эта вершина. Следует отметить, что сбалансированное декартово дерево может быть сбалансировано и на иных принципах. Основная задача при формировании сбалансированного декартова дерева – произвести перевычисление приоритетов для некоторых его вершин. С деревом возможны следующие действия: добавить пару, содержащую ключ и приоритет; удалить пару по заданному ключу; найти пару по заданному ключу; найти минимальный ключ, приоритет которого не меньше заданного приоритета и другие. В результате были получены соответствующие алгоритмы (см. рисунок 1), которые имеют логарифмическую временную сложность. Реализованные в виде C++ шаблонов [4] алгоритмы были протестированы и применены в системе динамического распределения памяти, использующей стратегию «первый подходящий» [2]. Размер исходных кодов составил 90 КиБ. Времена выполнения по результатам тестирования работы алгоритмов в системе динамического распределения памяти представлены на рисунке 2.

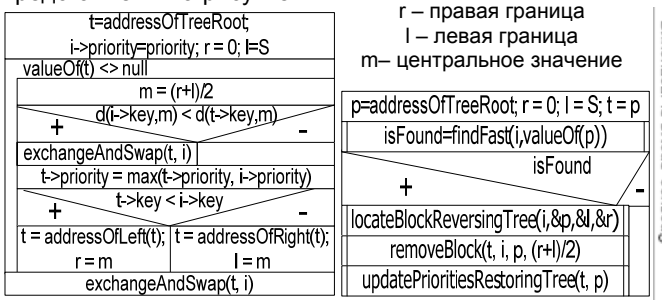

Рисунок 1. Алгоритм добавления и удаления узла i

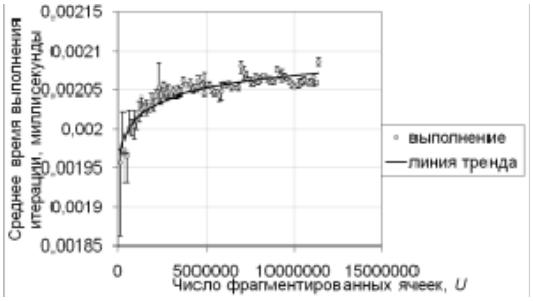

Рисунок 2. Зависимость времени от числа ячеек

Список источников:
1. Новиков Ф. А. Дискретная математика для программистов / Ф. А. Новиков СПб: Питер, 2000. – 304 с.
2. Кнут, Д. Искусство программирования / Д. Кнут. – Пер с англ.:М.: Издательский дом «Вильямс», 2000. – 682 с. – Т. 1
3. Шилд, Г. Полный справочник по C++ / Г. Шилд. – Пер. с англ. — М.: Издательский дом «Вильямс», 2006. – 800 с.