

Описание модификации R-дерева для работы с перемещающимися объектами

А.Г. Гогиян

(Университет ИТМО, Санкт-Петербург)

Научный руководитель - Д.Н. Шинкарук

(Университет ИТМО, Санкт-Петербург)

Введение. Стандартная реализация R-дерева (англ. R-tree) используется для организации доступа к пространственным, многомерным данным. В случае перемещения объектов в пространстве их положение меняется. В стандартной реализации дерева реализовать перемещение объектов возможно лишь удалением объекта и вставкой его с обновленными координатами. Таким образом, перемещение множества объектов на по одному вектору возможно реализовать за $O(N)$ (удаление и повторная вставка каждого элемента). Описываемое улучшение позволяет реализовать перемещение множества элементов на единый вектор за $O(\log(N))$ (сложность, пропорциональную высоте дерева).

Описание модификации. Предлагается перевести каждый узел R-дерева из абсолютной системы координат в относительную от своего отца. Координаты корня дерева всегда являются абсолютными. Для этого при вставке, удалении или поиске элементов на каждом переходе от предка к потомку необходимо смещать рабочую пространственную область на вектор, заданный относительными координатами потомка. Таким образом, при переходе от корня дерева к листу необходимо совершить h смещений рабочей области, где h – высота R-дерева.

Также в интерфейс R-дерева добавляется функция `move`, позволяющая сместить все хранящиеся в указанной области объекты на указанный вектор. Функция вызывается на корне дерева и обрабатывает его прямых потомков следующим образом:

- если область потомка не пересекается со смещаемой областью, потомок пропускается
- если область потомка является подобластью смещаемой области, относительные координаты потомка смещаются на вектор смещения.
- если область потомка пересекается с областью смещения, функция рекурсивно вызывается на этом потомке.

Описанная выше функция позволяет смещать множество узлов путем смещения их общего предка за константное время.

При построении R-дерева для ускорения операций с данными необходимо расположить объекты так, чтобы пересекающиеся друг с другом области узлов были минимальны. Стоит заметить, что описанные выше смещения узлов могут нарушить структуру дерева, увеличивая пересекающиеся друг с другом области. Для исправления подобных ситуаций необходимо после вызова функции `move` выполнить следующие действия.

- 1) После смещения узла необходимо проверить наличие пересечений областей между смещенным узлом и всеми его братьями. В случае, если пересечения обнаружены, необходимо разбить все пересекающиеся узлы заново с помощью функции `split` (существуют разные способы разбиения узлов на группы, рассмотрение их реализаций выходит за рамки данного доклада).

2) В случае если смещение узла вызвало изменение области его родителя, выполнить алгоритм с шага 1 на родителе.

Сложность описанной балансировки дерева зависит от реализации функции разбиения (сложность которой варьируется от $O(N)$ до $O(N^2)$, где N – количество разбиваемых узлов). К примеру, для линейного алгоритма разбиения, сложность которого составляет $O(N)$, сложность балансировки дерева составит $O(N \log(N))$.

Примером области применения описанного выше алгоритма могут быть электронные таблицы, в большинстве реализаций которых есть возможность сдвинуть указанный диапазон на заданное число ячеек по вертикали либо горизонтали. Внедрение описанного алгоритма позволит существенно ускорить подобные смещения и сохранить данные в сбалансированном виде, без существенного ущерба для других операций.

Автор:

Гогиян А.Г.

Научный руководитель:

Шинкарук Д.Н.