Алгоритмы и структуры данных - бинарные деревья

опубликована **10/03/2013 17:49:10** от **Ivan**
изменена **08/04/2013 01:51:36** от **Ivan**

Введение

В этой статье представлю деревья. Это еще одна структура данных, которая очень важна в компьютерных науках и может быть использована для моделирования различных связей между объектами из реального мира. Также деревья могут быть полезны как структура данных для различных алгоритмов или как основа некоторых других структур данных, как хэш таблици и множества, например.

Бинарные деревья

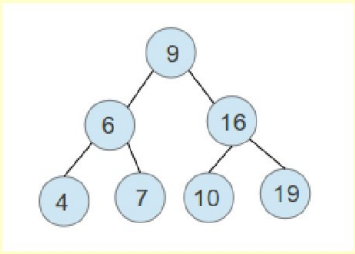
В общем, дерево, это структура данных, которая имеет корень и определенное количество детей, которые тоже могут иметь своих детей и т.д. Существует несколько терминов, которые необходимо дефинировать перед тем, как продолжить вперед:

1. Родитель – это прямой предшественник узла дерева.
2. Ребенок – прямой наследник узла дерева.
3. Предшественник
4. Наследник

Представлю бинарные деревья, потому что они имеют некоторые интересные характеристики. Но алгоритмы, показанные тут, могут также использоваться и в других деревьях с некоторыми изменениями. Бинарные деревья имеют следующие характеристики:

- Каждый узел имеет по два ребенка.
- Стоимость левого ребенка меньше стоимости родителя.
- Стоимость правого ребенка больше стоимости родителя.

Следующая фигура показывает бинарное дерево:



Дерево на этой фигуре сбалансировано, но это не является жестким требованием для бинарных деревьев. Хотя это желательная характеристика, потому что позволяет алгоритмам, базированным на деревьях, показывать оптимальную скорость.

Следующий фрагмент кода показывает структуру бинарного дерева в C/C++:

```
typedef int item_type;
typedef struct tree {
    item_type item;
    struct tree *parent;//optional
    struct tree *left;
    struct tree *right;
} tree;
```

Вот то же самое в Java:

Поиск на сайте

Про автора

Иван Николов

- ASP.NET
- C/C++/C#
- Java
- MapReduce (Hadoop)
- Python
- Silverlight
- WPF
- ADO.NET Entity Framework
- WCF Web Services
- Фотография
- Спорт

```java
/**
 * Represents a tree node.
 *
 * @author Ivan Nikolov
 *
 */
public class Node {
    private int data;
    private Node left;
    private Node right;
    private Node parent;

    public int getData() {
        return this.data;
    }

    public void setData(int data) {
        this.data = data;
    }

    public Node getLeft() {
        return this.left;
    }

    public void setLeft(Node left) {
        this.left = left;
    }

    public Node getRight() {
        return this.right;
    }

    public void setRight(Node right) {
        this.right = right;
    }

    public Node getParent() {
        return this.parent;
    }

    public void setParent(Node parent) {
        this.parent = parent;
    }
}
```

```java
/**
 * Represents a tree.
 * @author Ivan Nikolov
 *
 */
public class Tree {
    private Node root;

    public Tree() {
        this.root = null;
    }
}
```

Операции над бинарными деревьями

Опять же есть нескольно основных операций над бинарными деревьями.

1. Добавление
2. Удаление
3. Поиск
4. Обход – это дополнительная операция к трем основным, которые мы рассматривали в структурах данных.

Добавление

Добавление элемента в бинарное дерево – операция, которая имеет **логарифмическую сложность** по отношению размеру дерева (**logn**). Это очень интересная операция, потому что каждый элемент в бинарном дереве имеет определенную позицию. Поэтому сначала надо найти, куда его поставить, а затем и осуществить это.

Следует код на C/C++:

```c
void insert_tree(tree **t, item_type item, tree *parent) {
    tree *p;
    if (*t == NULL) {
        p = (tree *)malloc(sizeof(tree));
        p->left = p->right = NULL;
        p->parent = parent;
        p->item = item;
        *t = p;
        return;
    }

    if (item < (*t)->item) {
        insert_tree(&((*t)->left), item, *t);
    } else {
        insert_tree(&((*t)->right), item, *t);
    }
}
```

То же самое на Java выглядит так:

```java
public void insert(int data) {
    root = insert(root, data, null);
}

private Node insert(Node current, int data, Node parent) {
    if (current == null) {
        current = new Node();
        current.setData(data);
        current.setLeft(null);
        current.setRight(null);
        current.setParent(parent);
    } else if (data < current.getData()) {
        current.setLeft(insert(current.getLeft(), data, current));
    } else {
        current.setRight(insert(current.getRight(), data, current));
    }
    return current;
}
```

Сейчас уже вероятнее всего можете почувствовать разницу между C/C++ и Java!

Поиск

Сейчас пришло время представить операцию поиска в бинарном дереве. Она выглядит как добавление и занимает **логарифмическое время** (**logn**). Это относится к хорошо сбалансированному дереву, конечно!

Код в C/C++:

```c
tree *search_tree(const tree *t, item_type i) {
    if (t == NULL) return NULL;
    if (t->item == i) return t;
    if (i < t->item) {
        return search_tree(t->left, i);
    } else {
        return search_tree(t->right, i);
    }
}
```

Сейчас и Java:

```java
public Node find(int data) {
    return find(root, data);
}

private Node find(Node current, int data) {
    if (current == null)
        return null;
    if (current.getData() == data)
        return current;
    return find(
            data < current.getData() ? current.getLeft()
                    : current.getRight(), data);
}
```

Бинарное дерево имеет несколько интересных характеристик, позволяющих нам найти специальные элементы очень легко. Например минимальный и максимальный элемент. Самая маленькая стоимость та, которая находится в самом левом крае, а максимальная – в самом правом.

Вот код поиска минимума в C/C++:

```c
tree *find_min(const tree *t) {
    if (t == NULL) return NULL;
    tree *min = t;
    while (min->left != NULL) {
        min = min->left;
    }
    return min;
}
```

И в Java:

```java
public Node findMin() {
    Node min = root;
    if (min == null) return null;
    while (min.getLeft() != null) {
        min = min.getLeft();
    }
    return min;
}
```

Удаление

Удаление элемента из дерева самая трудная операция. Тем не менее она также отнимает **logn** время. Есть несколько специальных ситуаций, которые должны быть решены:

1. Удаление элемента без детей – просто освобождаем память.
2. Удаление элемента с одним ребенком – смена указателя родителя указывать директно к ребенку удаляемого элемента и освобождение памяти.
3. Удаление элемента с только одним ребенком и это КОРЕНЬ – перемещение ребенка на место корня и освобождение памяти.
4. Удаление элемента с двумя детьми – это самая сложная операция. Самый подходящий способ исполнения это разменять стоимости удаляемого элемента и максимальную стоимость левого поддерева или минимальную правого поддерева (потому что это сохранит характеристики дерева) и тогда удаляем элемент без или с одним ребенком.

Вот код в C/C++:

```c
void delete_element(tree **t, item_type item) {
    if (t == NULL) return;
    tree *element = search_tree(*t, item);
    if (element == NULL) return;
    int hasParent = element->parent != NULL;
    int isLeft = hasParent && element->item < element->parent->item;
    if (element->left == NULL && element->right == NULL) {//no childrer
        if (hasParent) {
            if (isLeft) {
                element->parent->left = NULL;
            } else {
                element->parent->right = NULL;
            }
        }
        free(element);
    } else if (element->left != NULL && element->right == NULL) {//only
        element->left->parent = element->parent;//even if it does not h
        if (hasParent) {
            if (isLeft) {
                element->parent->left = element->left;
            } else {
                element->parent->right = element->left;
            }
        } else {
            *t = element->left;
        }
        free(element);
    } else if (element->left == NULL && element->right != NULL) {//only
        element->right->parent = element->parent;
        if (hasParent) {
            if (isLeft) {
                element->parent->left = element->right;
            } else {
                element->parent->right = element->right;
            }
        } else {
            *t = element->right;
        }
        free(element);
    } else {//it has two children...
        //find the minimum of the right subtree and relabel the current
        tree *right_min = find_min(element->right);
        element->item = right_min->item;//relabel
        //delete the new minimum
        delete_element(&right_min, right_min->item);
    }
}
```

И сейчас имплементация на Java:

```java
public void delete(int data) {
    root = delete(root, data);
}

private Node delete(Node startNode, int data) {
    Node element = find(startNode, data);
    if (element == null) return startNode;
    boolean hasParent = element.getParent() != null;
    boolean isLeft = hasParent && element.getData() < element.getParent
    if (element.getLeft() == null && element.getRight() == null) {
        if (hasParent) {
            if (isLeft) {
                element.getParent().setLeft(null);
            } else {
                element.getParent().setRight(null);
            }
        }
    } else if (element.getLeft() != null && element.getRight() == null)
        if (hasParent) {
            if (isLeft) {
                element.getParent().setLeft(element.getLeft());
            } else {
                element.getParent().setRight(element.getLeft());
            }
        } else {
            startNode = element.getLeft();
        }
    } else if (element.getLeft() == null && element.getRight() != null)
        if (hasParent) {
            if (isLeft) {
                element.getParent().setLeft(element.getRight());
            } else {
                element.getParent().setRight(element.getRight());
            }
        } else {
            startNode = element.getRight();
        }
    } else {
        Node rightMin = findMin(element.getRight());
        element.setData(rightMin.getData());
        return delete(rightMin, rightMin.getData());
    }
    element = null;
    return startNode;
}
```

Опять же можете увидеть как C/C++ выглядит более оптимально, потому что не требуется препоставлять референции и лично для меня это выглядит более интуитивно.

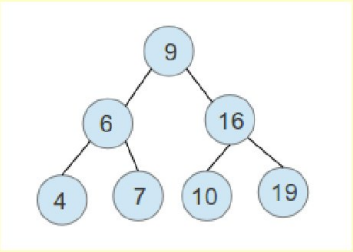
Обход дерева

Обход дерева, это процесс обхождения всех узлов и их обрабатывании. Существует три типа обхода

бинарного дерева:

1. Inorder – посещение левого поддерева, корня и правого поддерева.
2. Preorder – посещение корня, левого поддерева и правого поддерева.
3. Postorder – посещение левого поддерева, правого поддерева и корня.

Вот как выглядит выход для каждой из этих операций на дереве, показанном на следующей фигуре.



1. Inorder - 4, 6, 7, 9, 10, 16, 19. Как видите, стоимости сортированы. Это очень удобная характеристика бинарного дерева.
2. Preorder - 9, 6, 4, 7, 16, 10, 19.
3. Postorder - 4, 7, 6, 10, 19, 16, 9.

Вот и код, который имплементирует обход в C/C++:

```
typedef enum traversal_type {
    INORDER,
    PREORDER,
    POSTORDER
} traversal_type;

void traverse_tree(const tree *t, traversal_type trav_type) {
    if (t == NULL) return;
    switch (trav_type) {
        case INORDER:
            traverse_tree(t->left, trav_type);
            process_node(t->item);
            traverse_tree(t->right, trav_type);
        break;
        case PREORDER:
            process_node(t->item);
            traverse_tree(t->left, trav_type);
            traverse_tree(t->right, trav_type);
        break;
        case POSTORDER:
            traverse_tree(t->left, trav_type);
            traverse_tree(t->right, trav_type);
            process_node(t->item);
        break;
    }
}
```

И то же самое в Java:

```
/**
 * Represents traverse types available for a binary tree.
 * @author Ivan Nikolov
 *
 */
public enum TraverseType {
    INORDER,
    PREORDER,
    POSTORDER;
}
```

```
public void traverseTree(TraverseType traverseType) {
    traverseTree(root, traverseType);
    System.out.println();
}

private void traverseTree(Node current, TraverseType traverseType) {
    if (current == null)
        return;
    switch (traverseType) {
    case INORDER:
        traverseTree(current.getLeft(), traverseType);
        processNode(current);
        traverseTree(current.getRight(), traverseType);
        break;
    case PREORDER:
        processNode(current);
        traverseTree(current.getLeft(), traverseType);
        traverseTree(current.getRight(), traverseType);
        break;
    case POSTORDER:
        traverseTree(current.getLeft(), traverseType);
        traverseTree(current.getRight(), traverseType);
        processNode(current);
        break;
    }
}
```

Заключение

Эта статья представляет бинарные деревья с некоторыми операциями над ними и некоторые интересные характеристики. Множество подходов, использованных тут, могут быть легко перенесены в любой другой тип деревьев.

Веселитесь, открывая деревьи и их силу, когда создаете разные программы и алгоритмы!

Коментарии

Mark сказал/а:

метод processNode в Java не определен

12/05/2013 в 11:27:33

Ivan сказал/а:

Так же и в C++. Я его оставил, потому-что все зависит от того какая у тебя крайная цель. Самое легкое решение просто сделать например: System.out.println(node.getData());

21/05/2013 в 00:59:57

Jarik сказал/а:

Спасибо, отличная статья.

05/08/2014 в 18:23:19

Ivona сказал/а:

Здравейте :) Много ми беше полезна информацията, добре структурирана и разбрана. Но ми трябва и нещо за графи - създаване и прости операции. Ще съм благодарна, ако споделите нещо по тази тема!

05/11/2014 в 08:08:50

Al сказал/а:

Много кода мало объяснений.

25/01/2015 в 04:45:38

RoyBug сказал/а:

Зачем путать людей у Вас код не C/C++, а чистый C, а то, что он соберется и будет работать под C++ не делает его кодом C++. Если бы Вы написали код на ООП плюсах, то он был бы очень схож на Java.

24/06/2015 в 08:23:29

max сказал/а:

Здравствуйте. почему в методе insert я обязательно должен root = insert(root, data, null);. почему нельзя просто передать значение insert(root, data, null);.

29/06/2015 в 16:26:19

max сказал/а:

и почему вы не придерживаетесь "В поисковом дереве нет двух вершин с одинаковыми ключевыми значениями"? спасибо за статью. очень помогла!

29/06/2015 в 16:30:05

Anton сказал/а:

в поиск минимуму в коде Java нужно добваить ещё функцию public Node find(int data) { return find(root, data); } так как в удалении вы её вызываете с аргументом Node rightMin = findMin(element.getRight());

15/02/2016 в 01:44:36

Ivan сказал/а:

RoyBug - я согласен. Просто я видел много людей, которые использовали термин C/C++ и я решил так-же сделать.

24/09/2016 в 20:38:43

Ivan сказал/а:

max - когда я изучал бинарные деревья, я такое правило не слышал. Это просто сделать если поменять условия в if-ах. То, что я не применил такое правило, не делает код неправильным. В конце концов все зависит от конкретных требований. Ну, а если его применить, у тебя будет имплементация Set. :)

24/09/2016 в 20:38:59

Ivan сказал/а:

Anton, посмотри еще раз - код на месте. :)

24/09/2016 в 20:39:08

Ivan D. сказал/а:

Страхотна работа. Благодаря от мен.

29/01/2017 в 15:17:29

WalterTut сказал/а:

АЛКОВЕРИН АКТИВИРУЕТ РЕЖИМ АЛКОГОЛЬНОГО ОТТОРЖЕНИЯ С ALCOVIRIN выпить ПРОСТО НЕ УДАСТСЯ! Это первый биогенный растительный комплекс, способствующий выработке непереносимости алкоголя при совместном приеме капель и спиртных напитков, вызывая тошноту и его полное отторжение организмом! Кроме того, он оказывает мощное оздоровительное действие, устраняя алкогольную интоксикацию и способствуя восстановлению правильной работы органов и систем. Официальный сайт: http://alcovirin.bxox.info

15/05/2017 в 18:35:33

ShawnHef сказал/а:

Perfect instrumental background music for romantic and sentimental films, presenting your business, new products or your company in general with an optimistic and motivational touch. Visit site: https://audiojungle.net/user/commercial_music https://audiojungle.net/user/corporate_sound https://audiojungle.net/user/elijah_studio https://audiojungle.net/user/ie_sound https://audiojungle.net/user/momentumofmelody ==========

16/05/2017 в 03:20:33

Мебель каталог сказал/а:

Визитной карточкой любой квартиры является гостиная, она наиболее полно отражает вкусы и характер владельца. Необходимым элементом интерьера давным-давно стали гостиные стенки. В советские времена стенка была показателем достатка и престижности, импортной мебелью гордились и хвастались перед друзьями. Эти времена прошли, но стенки для гостиной остались важным элементом обстановки. За несколько десятилетий мебель преобразилась, улучшился дизайн, при производстве используются новейшие технологии, новые материалы. Современные стенки для гостиной обрели широкие функциональные возможности: они сочетают в себе свойства тумбы под аудио- и видеотехнику, шкафа для одежды, бара, книжного шкафа. При этом некоторые модели при достаточной вместительности обладают небольшими размерами, идеально вписываясь в интерьер небольших комнат. Применение новейших материалов и технологий позволяет дизайнерам создавать стенки не только классической конфигурации, но и оригинальной формы, например, стенки-горки. Качественная фурнитура делает легким и бесшумным открывание дверей на шкафах, ящики выдвигаются без особых усилий. Эксплуатация современных стенок приносит удовольствие их владельцам. Официальный сайт: http://bigwork.info/

04/06/2017 в 18:44:35

tinedol.1stbest.info сказал/а:

Tinedol – эффективное средство от грибка стопы, неприятного запаха и зуда. Перейти на сайт: http://tinedol.1stbest.info/

09/06/2017 в 19:06:37

GeorgeFus сказал/а:

Мазь Tinedol отзывы специалистов. кожа увлажнена, приятная наощупь. Португалия - 39 EUR. Передозировка, превышающая рекомендуемую дозу более чем в 2 раза, сопровождалась у больных острым снижением артериального давления и тромбоцитопенией со смертельным исходом. Откуда у меня появился грибок. 2016 Все права защищены. Производитель гарантирует стойкий результат, при условии регулярности применения Тинедола и соблюдения гигиены Кстати, применять мазь просто в домашних условиях и не требует наблюдения у специалистов Натуральные растительные ингредиенты не принесут вреда здоровью и коже, противопоказания к применению Tinedol отсутствуют. Он устраняет нежелательный запах и чувство зуда. Цена и где купить. Посылку получила неделю назад, с тех пор применяю мазь каждый день, как по инструкции Прошел пока только зуд, но это уже о многом говорит Буду применять до победного. Официальный сайт: http://tinedol.1stbest.info/

09/06/2017 в 23:59:40

Добавление комментария

Войдите или **зарегистрируйтесь** или добавьте дополнительные обязательные данные.

Имя (обязательно):

E-mail (обязателен, но не будет показан):

Сайт:

Докажите, что вы человек:

Добавить коментарь

Добавить коментарь