Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних інформаційних систем

Алгоритми та складність

Лабораторна робота №2

“Реалізація персистентної множини дійсних чисел на основі червоно-чорного

дерева”

Виконав студент 2-го курсу

Групи ІПС-22

Чорнокур Іван Вадимович

2024

Завдання:

Реалізація персистентної множини дійсних чисел на основі червоно-чорного дерева. Час роботи вставки і видалення в найгіршому випадку і об‘єм необхідної пам’яті мають бути О(log n). Тип даних - комплексні числа з цілочисельними компонентами

Теорія:

Реалізація персистентної множини дійсних чисел на основі червоно-чорного дерева передбачає створення структури даних, яка підтримує кілька версій дерева, де кожна нова операція створює нову версію, зберігаючи попередні. Червоний-Чорний Дерево (ЧЧД) — це самобалансуюче бінарне дерево пошуку, що підтримує баланс шляхом використання кольору кожного вузла (червоний або чорний). Персистентність означає здатність структури даних зберігати історію змін, підтримуючи доступ до попередніх версій даних.

Алгоритм:

1. Оголошення класів і структур:

• Визначити перерахування Color з двома значеннями: RED і BLACK.

• Визначити структуру Node, що представляє вузол дерева. Вузол містить значення типу T, колір, а також вказівники на лівого, правого нащадків і батька.

• Визначити клас PersistentRBTree, який міститиме корінь дерева та методи для вставки, видалення, пошуку та друку дерева.

2. Конструктори та допоміжні функції:

• Конструктор класу PersistentRBTree ініціалізує корінь дерева як nullptr.

• Функція createNode створює новий вузол із заданими значенням, кольором і батьківським вузлом.

• Функція copyNode створює глибоку копію вузла та його нащадків, повертаючи новий вузол з оновленими вказівниками на батька.

3. Операції обертання:

• Функції leftRotate і rightRotate виконують ліве та праве обертання вузла відповідно для підтримки властивостей червоно-чорного дерева.

4. Фіксація вставки:

• Функція insertFixup виправляє можливі порушення властивостей червоно-чорного дерева після вставки нового вузла.

5. Вставка вузла:

• Функція insertNode вставляє новий вузол у дерево, зберігаючи властивості бінарного дерева пошуку.

• Метод insert створює копію поточного дерева, вставляє новий вузол і викликає insertFixup.

6. Трансплантація вузлів:

• Функція transplant замінює один вузол іншим, оновлюючи вказівники.

7. Видалення вузла:

• Функція minimum знаходить вузол з мінімальним значенням у піддереві.

• Функція deleteFixup виправляє можливі порушення властивостей червоно-чорного дерева після видалення вузла.

• Функція deleteNode видаляє вузол із заданим значенням із дерева.

• Метод remove створює копію поточного дерева і викликає deleteNode для видалення вузла.

8. Пошук вузла:

• Метод search знаходить вузол із заданим значенням у дереві.

9. Обхід і друк дерева:

• Функція inOrderTraversal виконує обхід дерева в порядку in-order і друкує значення вузлів.

• Метод printTree викликає inOrderTraversal для друку всього дерева.

10. Основна функція:

• Створюється об’єкт дерева.

• Виконуються операції вставки, пошуку, видалення і друку дерева.

Псевдокод для ключових функцій:

Вставка вузла:

insert(T value):

newRoot = copyNode(root, nullptr)

node = createNode(value, RED)

root = insertNode(newRoot, node)

insertFixup(node)

Видалення вузла:

remove(T value):

newRoot = copyNode(root, nullptr)

root = deleteNode(newRoot, value)

Пошук вузла:

search(T value):

current = root

while current != null:

if current.value == value:

return true

else if value < current.value:

current = current.left

else:

current = current.right

return false

Обертання вліво:

leftRotate(NodePtr x):

y = createNode(\*x.right, x.parent)

x.right = y.left

if y.left != null:

y.left.parent = x

y.parent = x.parent

if x.parent == null:

root = y

else if x == x.parent.left:

x.parent.left = y

else:

x.parent.right = y

y.left = x

x.parent = y

Обертання вправо:

rightRotate(NodePtr x):

y = createNode(\*x.left, x.parent)

x.left = y.right

if y.right != null:

y.right.parent = x

y.parent = x.parent

if x.parent == null:

root = y

else if x == x.parent.right:

x.parent.right = y

else:

x.parent.left = y

y.right = x

x.parent = y

Виправлення після вставки:

insertFixup(NodePtr z):

while z.parent != null and z.parent.color == RED:

if z.parent == z.parent.parent.left:

y = z.parent.parent.right

if y != null and y.color == RED:

z.parent.color = BLACK

y.color = BLACK

z.parent.parent.color = RED

z = z.parent.parent

else:

if z == z.parent.right:

z = z.parent

leftRotate(z)

z.parent.color = BLACK

z.parent.parent.color = RED

rightRotate(z.parent.parent)

else:

y = z.parent.parent.left

if y != null and y.color == RED:

z.parent.color = BLACK

y.color = BLACK

z.parent.parent.color = RED

z = z.parent.parent

else:

if z == z.parent.left:

z = з.parent

rightRotate(z)

z.parent.color = BLACK

z.parent.parent.color = RED

leftRotate(z.parent.parent)

root.color = BLACK

Обхід і друк дерева:

inOrderTraversal(NodePtr node):

if node != null:

inOrderTraversal(node.left)

print(node.value)

inOrderTraversal(node.right)

Основна функція:

main():

tree = PersistentRBTree<complex<int>>()

tree.insert({1, 2})

tree.insert({3, 4})

tree.insert({5, 6})

tree.insert({7, 8})

print("Tree after insertions:")

tree.printTree()

print("Search for (3, 4):", tree.search({3, 4}) ? "Found" : "Not Found")

print("Search for (9, 10):", tree.search({9, 10}) ? "Found" : "Not Found")

tree.remove({3, 4})

print("Tree after removing (3, 4):")

tree.printTree()

Модулі програми:

Модуль 1: Визначення кольору та структури вузла

Визначає кольори вузлів (червоний або чорний) та структуру вузла червоно-чорного дерева, включаючи значення, колір і вказівники на нащадків і батьківський вузол.

Модуль 2: Визначення класу PersistentRBTree

Визначає клас червоно-чорного дерева, який містить корінь дерева і методи для вставки, видалення, пошуку та друку.

Модуль 3: Конструктори і допоміжні функції

Містить конструктор класу, який ініціалізує корінь дерева, а також допоміжні функції для створення та копіювання вузлів.

Модуль 4: Операції обертання

Включає методи для виконання лівого та правого обертання вузлів для підтримки балансу дерева.

Модуль 5: Фіксація вставки

Містить методи для виправлення можливих порушень властивостей червоно-чорного дерева після вставки нового вузла.

Модуль 6: Вставка вузла

Включає методи для вставки нового вузла в дерево, зберігаючи властивості бінарного дерева пошуку і червоно-чорного дерева.

Модуль 7: Трансплантація вузлів

Містить методи для заміни одного вузла іншим, оновлюючи відповідні вказівники.

Модуль 8: Видалення вузла

Включає методи для видалення вузла із заданим значенням із дерева та виправлення можливих порушень після видалення.

Модуль 9: Пошук вузла

Містить методи для пошуку вузла із заданим значенням у дереві.

Модуль 10: Обхід і друк дерева

Включає методи для обходу дерева в порядку in-order та друку значень вузлів.

Модуль 11: Основна функція

Включає приклад використання дерева, виконуючи операції вставки, пошуку, видалення та друку дерева.

Висновок:

Персистентна множина дійсних чисел на основі червоно-чорного дерева, дозволяє ефективно виконувати операції вставки, видалення та пошуку за час O(log n). Вона також забезпечує персистентність, тобто можливість зберігати попередні версії дерева після виконання операцій модифікації. Це досягається шляхом копіювання вузлів, що змінюються, замість зміни існуючих вузлів безпосередньо. Основні модулі програми включають: