Двоичные деревья поиска

Двоичные деревья поиска позволяют выполнять следующие операции с динамическими множествами: ПОИСК, МИНИМУМ, МАКСИМУМ, ПРЕДЫДУЩИЙ, СЛЕДУЮЩИЙ, ВСТАВИТЬ и УДАЛИТЬ. Таким образом, дерево поиска может быть использовано и как словарь, и как очередь с приоритетами.

Время выполнения словарных операций пропорционально высоте дерева. Если двоичное дерево «плотно заполнено» (все его уровни имеют максимально возможное число вершин), то его высота (и тем самым время выполнения операций) пропорционально логарифму числа вершин. Напротив, если дерево представляет собой линейную цепочку из n вершин, это время вырастает до O(n). Высота случайного двоичного дерева поиска есть O(log n), так что в этом случае время выполнения основных операций O(log n).

В двоичном дереве поиска каждая вершина может иметь (или не иметь) левого и правого сына; каждая вершина кроме корня имеет родителя. При представлении с использованием указателей мы храним для каждой вершины дерева, помимо значения ключа и дополнительных данных, также и указатели ЛЕВЫЙ_СЫН, ПРАВЫЙ_СЫН и РОДИТЕЛЬ. Если ребёнка нет (или родителя - для корня), соответствующее поле содержит ПУСТО.

Ключи в двоичном дереве поиска хранятся с соблюдением свойства упорядоченности. Свойство упорядоченности позволяет напечатать все ключи в неубывающем порядке с помощью простого рекурсивного алгоритма. Существует три способа обхода деревьев: прямой (префиксный), внутренний (инфиксный), обратный (постфиксный).

Вызов ИНФИКСНЫЙ_ДВОИЧНОЕ_ДЕРЕВО(КОРЕНЬ[T]) печатает (в указанном порядке(все ключи, входящие в дерево T с корнем КОРЕНЬ[T].

ИНФИКСНЫЙ_ДВОИЧНОЕ_ДЕРЕВО(x)

если x ≠ ПУСТО

тогда ИНФИКСНЫЙ_ДВОИЧНОЕ_ДЕРЕВО(ЛЕВЫЙ_СЫН[x])

напечатать КЛЮЧ[x]

ИНФИКСНЫЙ_ДВОИЧНОЕ_ДЕРЕВО(ПРАВЫЙ_СЫН[x])

Поиск в двоичном дереве

Поиск

Процедура поиска получает на вход искомый ключ к и указатель x на корень дерева, в котором производится поиск. Она возвращает указатель на вершину к (если такая есть) или специальное значение ПУСТО (если такой вершины нет).

ПОИСК_В_ДЕРЕВЕ(x, к)

если x = ПУСТО или к = КЛЮЧ[x]

тогда вернуть x

если к<КЛЮЧ[x]

тогда вернуть ПОИСК_В_ДЕРЕВЕ(ЛЕВЫЙ_СЫН[x], к)

иначе вернуть ПОИСК_В_ДЕРЕВЕ(ПРАВЫЙ_СЫН[x], к)

Вот итеративная версия той же процедуры (которая, как правило, более эффективна):

ИТЕРАТИВНЫЙ_ПОИСК_В_ДЕРЕВЕ(x, к)

пока x ≠ ПУСТО и к ≠ КЛЮЧ[x]

делать если к < КЛЮЧ[x]

тогда x ← ЛЕВЫЙ_СЫН[x]

иначе x ← ПРАВЫЙ_СЫН[x]

вернуть x

Минимум и максимум

Минимальный ключ в дереве поиска можно найти, пройдя по указателям ЛЕВЫЙ_СЫН от корня (пока не упрёмся в ПУСТО). Процедура возвращает указатель на минимальный элемент поддерева с корнем x.

МИНИМУМ_ДЕРЕВА (x)

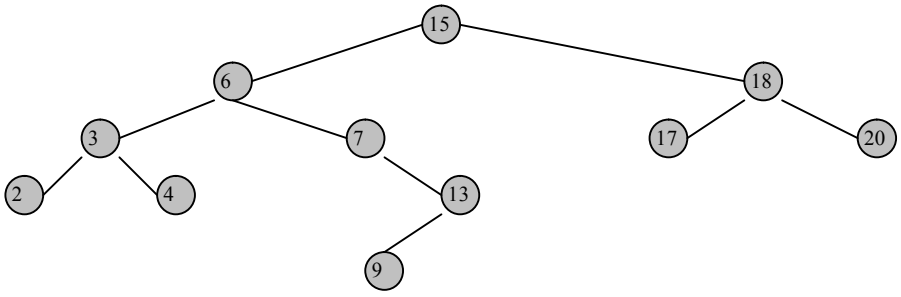
пока ЛЕВЫЙ_СЫН[x] ≠ ПУСТО

делать x ← ЛЕВЫЙ_СЫН[x]

вернуть x

Свойство упорядоченности гарантирует правильность процедуры МИНИМУМ_ДЕРЕВА.

Алгоритм МАКСИМУМ_ДЕРЕВА симметричен:



МАКСИМУМ_ДЕРЕВА (x)

пока ПРАВВЫЙ_СЫН[x] ≠ ПУСТО

делать x ← ПРАВЫЙ_СЫН[x]

вернуть x

Оба алгоритма требуют времени O(h), где h – высота дерева (поскольку двигаются по дереву только вниз).

Следующий и предыдущий элементы

Свойство упорядоченности позволяет определять следующего и предыдущего, двигаясь по дереву. Вот процедура, которая возвращает указатель на следующий элемент (если все ключи различны, он содержит следующий по величине ключ) или ПУСТО, если элемент x – последний в дереве:

СЛЕДУЮЩИЙ_В_ДЕРЕВЕ(x)
если ПРАВЫЙ_СЫН[x] ≠ ПУСТО
тогда вернуть МИНИМУМ_ДЕРЕВА(ПРАВЫЙ_СЫН[x])
y ← p[x]
пока y ≠ ПУСТО и x = ПРАВЫЙ_СЫН[y]
делать x ← y
y ← p[y]
вернуть y

Процедура СЛЕДУЮЩИЙ_В_ДЕРЕВЕ отдельно рассматривает два случая. Если правое поддерево вершины x непусто, то следующий за x элемент – минимальный элемент в этом поддереве и равен МИНИМУМ_ДЕРЕВА(ПРАВЫЙ_СЫН[x]).

Пусть теперь правое поддерево вершины x пусто. Тогда мы идём от x вверх, пока не найдём вершину, являющуюся левым сыном своего родителя. Этот родитель (если он есть) и будет искомым элементом.

Время работы этой процедуры на дереве высоты h есть O(h), так как мы двигаемся либо только вверх, либо только вниз.

Процедура ПРЕДЫДУЩИЙ_В_ДЕРЕВЕ симметрична.

Добавление и удаление элемента

Эти операции меняют дерево, сохраняя свойство упорядоченности.

Добавление

Процедура ДОБАВИТЬ_В_ДЕРЕВО добавляет элемент в подходящее место дерева T (сохраняя свойство упорядоченности). Параметром процедуры является указатель z на новую вершину, в которую помещены значения КЛЮЧ[z] (добавляемое значение ключа), ЛЕВЫЙ_СЫН[z]=ПУСТО и ПРАВЫЙ_СЫН[z]=ПУСТО. В ходе работы процедура меняет дерево T и (возможно) некоторые поля вершины z, после чего новая вершина с данным значением ключа оказывается вставленной в подходящее место дерева.

ДОБАВИТЬ_В_ДЕРЕВО(T, z)
y ← ПУСТО
x ← КОРЕНЬ[T]
пока x ≠ ПУСТО
    делать y ← x
        если КЛЮЧ[z] < КЛЮЧ[y]
            тогда x ← ЛЕВЫЙ_СЫН[x]
            иначе ПРАВЫЙ_СЫН[x]
p[z] ← y

если y = ПУСТО

тогда КОРЕНЬ[T] ← z

иначе если КЛЮЧ[z] < КЛЮЧ[y]

тогда ЛЕВЫЙ_СЫН[y] ← z

иначе ПРАВЫЙ_СЫН[y] ← z

Как и остальные операции, добавление требует времени O(h), для дерева высоты h.

Удаление

Параметром процедуры удаления является указатель на удаляемую вершину. При удалении возможны три случая:

- если у вершины нет детей
- если у вершины один ребёнок
- если у вершины два ребёнка

В первом случае всё очень просто. Если же у z есть один сын, можно «вырезать» z, соединив его родителя непосредственно с его ребёнком. Если же детей двое, требуются некоторые приготовления: мы находим следующий (в смысле порядка на ключах) за z элемент y; у которого нет левого сына. Теперь можно скопировать ключ и дополнительные данные из вершины y в вершину z, а саму вершину y удалить описанным выше способом.

Примерно так и действует процедура УДАЛИТЬ_ИЗ_ДЕРЕВА.

УДАЛИТЬ_ИЗ_ДЕРЕВА(T, z)

если ЛЕВЫЙ_СЫН[z] = ПУСТО или ПРАВЫЙ_СЫН[z] = ПУСТО

тогда y ← z

иначе y ← СЛЕДУЮЩИЙ_В_ДЕРЕВЕ(z)

если ЛЕВЫЙ_СЫН[y] ≠ ПУСТО

тогда x ← ЛЕВЫЙ_СЫН[y]

иначе x ← ПРАВЫЙ_СЫН[y]

если x ≠ ПУСТО

тогда p[x] ← p[y]

если p[y] = пУСТО

тогда КОРЕНЬ[T] ← x

иначе если y = ЛЕВЫЙ_СЫН[p[y]]

тогда ЛЕВЫЙ_СЫН[p[y]] ← x

иначе ПРАВЫЙ_СЫН[p[y]] ← x

если y ≠ z

тогда КЛЮЧ[z] ← КЛЮЧ[y]

►Копируем дополнительные данные, связанные с y.

вернуть y

Время выполнения есть O(h) на дереве высоты h.