**Тема 4. Методы поиска**

## *4.2. Логарифмический поиск в динамических таблицах*

### *4.2.1. Деревья бинарного поиска*

Последовательно распределенные таблицы позволяют обеспечить достаточно быстрое нахождение имен применением бинарного поиска. Однако существенным недостатком такой организации таблиц является низкая эффективность выполнения операций включения и исключения имен относительно времени поиска, что существенно снижает общую эффективность работы с динамическими таблицами. Временная сложность операций включения и исключения составляет $O(n)$, а для бинарного поиска – $O(\log n)$.

Связная организация динамических таблиц в виде связных списков позволяет достичь высокой эффективности выполнения операций включения и исключения (время их выполнения не зависит от размера таблицы). Однако к таким таблицам нельзя применить бинарный поиск (поскольку не обеспечивается прямой доступ к именам таблицы), а возможен только последовательный поиск с временной сложностью $O(n)$.

Для динамических таблиц необходима такая структура данных, которая была бы приспособлена как к методу бинарного поиска, так и к эффективному выполнению операций включения и исключения. Такой структурой данных является дерево бинарного поиска.

*Деревом бинарного поиска* (ДБП) над именами $x_1, x_2, ..., x_n$ называется расширенное бинарное дерево, все внутренние вершины которого помечены именами из таблицы $T = \{x_1, x_2, ..., x_n\}$ таким образом, что симметричный порядок прохождения вершин совпадает с естественным порядком имен. Каждая из $n + 1$ внешних вершин (листьев) соответствует промежутку в таблице. Поскольку симметричный порядок прохождения вершин является естественным порядком, для каждой вершины $x_i$ все имена в левом поддереве с корнем $x_i$ предшествуют $x_i$ в естественном порядке и все имена в правом поддереве с корнем $x_i$ следуют за $x_i$ в естественном порядке. На рис. 4.3 представлено ДБП для таблицы $T = \{x_1, x_2, \ldots, x_9\}$, $x_1 \le x_2 \le \ldots \le x_9$.

Каждая внутренняя вершина ДБП может быть представлена узлом, состоящим из полей *left*, *info* и *right*, где *left* и *right* содержат указатели на левого и правого сыновей соответственно, а поле *info* содержит имя, хранящееся в узле (узловое представление дерева). Доступ к дереву обеспечивается с помощью внешнего указателя *root* на его корень. Очевидно, что для пустого дерева *root* = Λ. Если указатель (*left* или *right*) имеет значение Λ, это означает, что у узла нет соответствующего сына, т. е. он указывает на пустое поддерево.

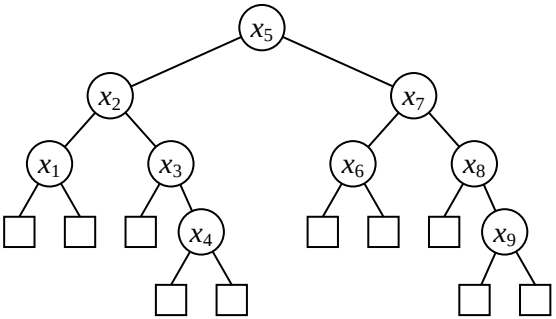


Рис. 4.3. Дерево бинарного поиска

Понятие внешних вершин носит формальный характер, поэтому соответствующие им узлы можно не создавать. При разработке некоторых алгоритмов, чтобы упростить обработку граничных условий, можно явно хранить и использовать единственный фиктивный внешний узел, соответствующий всем внешним узлам дерева, т. е. фиктивный узел будет выполнять функции сторожа. В этом случае значение указателя на этот фиктивный узел будет играть роль значения $\Lambda$.

При необходимости в структуру узла можно добавить поле *father*, которое указывает на отца данного узла, для облегчения движения от потомков к предкам. Очевидно, что для корня дерева значение этого поля равно $\Lambda$.

**Поиск *z***. Поиск имени *z* в ДБП осуществляется сравнением *z* с именем, стоящим в корне. Если дерево пустое, то *z* в таблице отсутствует и поиск завершается безуспешно. Если *z* совпадает с именем в корне, поиск завершается успешно. Если *z* предшествует имени в корне, поиск продолжается рекурсивно в левом поддереве. Если *z* следует за именем в корне, поиск продолжается рекурсивно в правом поддереве корня. Очевидно, что успешный поиск завершается во внутреннем узле ДБП, а безуспешный – во внешнем узле. Процедуру поиска имени *z* в таблице, организованной в виде ДБП, можно представить алгоритмом 4.6.

$p \leftarrow root$
**while** $p \neq \Lambda$ **do**
$$\textbf{case}\begin{cases} z = p.info : \textbf{return}(p) & \text{// найдено: } p \text{ указывает на } z \\ z < p.info : p \leftarrow p.left \\ z > p.info : p \leftarrow p.right \end{cases}$$
**return**$(\Lambda)$ // не найдено

Алгоритм 4.6. Поиск в дереве бинарного поиска

**Распечатка**. Очевидно, что операцию распечатки, предполагающую печать имен из таблицы в естественном порядке, легко выполнить использованием симметричного прохождения бинарных деревьев.

**Включение z**. Включение имени *z* в таблицу выполняется в том случае, если поиск *z* в таблице завершается безуспешно. Для таблиц, организованных в виде ДБП, включение имени *z* соответствует добавлению в дерево нового узла. Новый узел всегда добавляется как лист вместо внешнего узла, соответствующего промежутку, где могло бы находиться *z*, если бы оно входило в таблицу. При этом добавляемый узел должен быть связан с последним узлом, пройденным во время безуспешного поиска *z*.

Для реализации включения необходимо обеспечить сохранение адреса отца добавляемого узла, поскольку в алгоритме поиска указатель *p* после выхода из цикла **while** имеет значение Λ. Для этого достаточно добавить внешний указатель *q*, в котором запоминается предыдущее значение указателя *p*. Кроме того, необходимо предусмотреть возможность добавления нового узла в первоначально пустое дерево, поскольку установка связей между узлами в этом случае отличается от остальных. В результате получается процедура включения, представленная алгоритмом 4.7.

$p \leftarrow root$
$q \leftarrow \Lambda$ // $q$ – указатель отца добавляемого узла

$$\textbf{while } p \neq \Lambda \textbf{ do case} \begin{cases} z = p.info: \text{ // } z \text{ уже в таблице} \\ z < p.info: \begin{cases} q \leftarrow p \\ p \leftarrow p.left \end{cases} \\ z > p.info: \begin{cases} q \leftarrow p \\ p \leftarrow p.right \end{cases} \end{cases}$$

// $z$ нет в таблице, добавить узел
$new(p)$
$p.info \leftarrow z$
$p.left \leftarrow p.right \leftarrow \Lambda$
**if** $root = \Lambda$
  **then** $root \leftarrow p$ // узел добавляется в пустое ДБП
  **else if** $z < q.info$
    **then** $q.left \leftarrow p$ // левый сын
    **else** $q.right \leftarrow p$ // правый сын

Алгоритм 4.7. Включение нового узла в дерево бинарного поиска

Возможна и рекурсивная реализация операции включения имени $z$ в ДБП, представленная алгоритмом 4.8. Рекурсивная функция *INSERT*($z$, $t$) выдает в качестве значения указатель на дерево, в которое добавляется $z$. Таким образом, для добавления $z$ в дерево с указателем корня *root* достаточно выполнить операцию *root* ← *INSERT*(*z, root*). Следует отметить, что функция автоматически обрабатывает ситуацию, когда до включения нового узла ДБП было пустым. Очевидно, что рекурсивная реализация включения с точки зрения времени работы уступает нерекурсивному алгоритму.

**function** $INSERT(z, t)$
  $p \leftarrow t$
  **if** $p = \Lambda$
    **then** $\begin{cases} new(p) \\ p.info \leftarrow z \\ p.left \leftarrow p.right \leftarrow \Lambda \end{cases}$
    **else case** $\begin{cases} z = p.info: \text{ // } z \text{ уже в таблице} \\ z < p.info: p.left \leftarrow INSERT(z, p.left) \\ z > p.info: p.right \leftarrow INSERT(z, p.right) \end{cases}$
  $INSERT \leftarrow p$
**return**

Алгоритм 4.8. Рекурсивная функция включения нового узла в ДБП

**Исключение *z***. Исключение сложнее включения, поскольку в ДБП исключается внутренний узел, который может быть листом, иметь одного или двух сыновей. Если исключаемый узел с именем *z* является листом или имеет только одного сына, удаление выполняется достаточно просто – при исключении узла *z* его сын (если он есть) становится сыном отца узла *z* (рис. 4.4, *а*). Если же исключаемый узел с именем *z* имеет двух сыновей, его прямо удалить нельзя. В этом случае в таблице необходимо найти имя $y_1$, непосредственно предшествующее имени *z*, или имя $y_2$, непосредственно следующее за именем *z* в естественном порядке. Очевидно, что оба имени принадлежат узлам, имеющим не более одного сына. Далее имя *z* исключается заменой его либо именем $y_1$, либо именем $y_2$, а затем удалением узла, который содержал $y_1$ или $y_2$ соответственно (рис. 4.4, *б*).

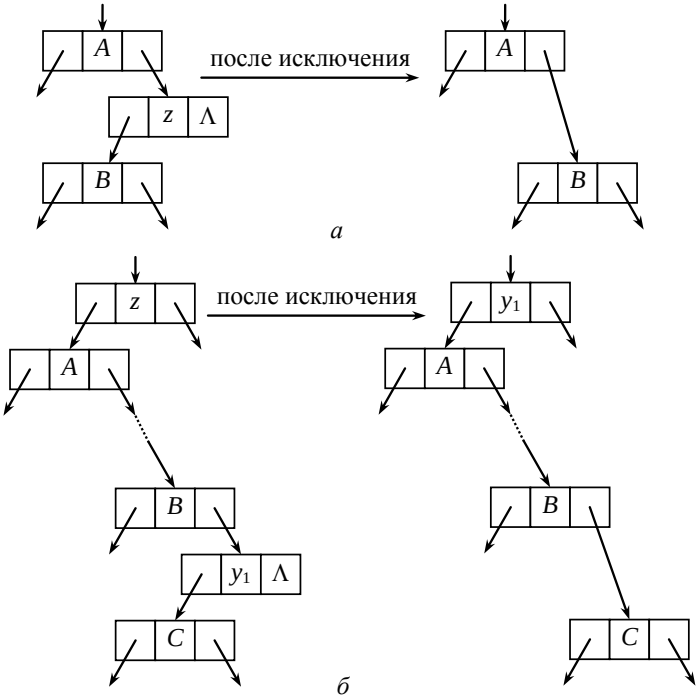


Рис. 4.4. Исключение узла из дерева бинарного поиска:
*а* – исключаемый узел имеет не более одного сына;
*б* – исключаемый узел имеет двух сыновей