Неупорядоченные множества всегда могут обрабатываться, как упорядоченные: путем искусственного введения порядка над их элементами (например, присвоением «имен» элементам и использованием алфавитного порядка). Типовая структура данных в этой ситуации такова:

Таблица II

Структура данных	Допустимые операции
Сливаемая пирамида	ВСТАВИТЬ, УДАЛИТЬ, НАЙТИ, ОБЪЕДИНИТЬ, (MIN)

Каждую из вышеуказанных операций можно выполнить за время $O(\log N)$, где N — размер множества, запасенного в этой структуре данных путем использования, как обычно, дерева, сбалансированного по высоте. Если элементы рассматриваемого множества представлены целыми числами от 1 до N, то более сложная реализация этой структуры данных позволяет выполнить N операций над множеством размером N за время $O(N \cdot A(N))$, где $A(N)$ — чрезвычайно медленно растущая функция, связанная с обратной функцией Аккермана (например, для $N \leqslant 2^{2^{16}}$, или $\sim 10^{20\,000}$, $A(N) \leqslant 5$).

Стандартные структуры данных, рассмотренные выше, широко используются в алгоритмах вычислительной геометрии. Однако природа геометрических задач привела к созданию специальных, необычных структур данных, две из которых оказались настолько общезначимыми, что целесообразно представить их в данной вводной главе. Эти структуры — дерево отрезков и реберный список с двойными связями.

1.2.3.1. Дерево отрезков

Дерево отрезков, впервые введенное Дж. Бентли [Bentley (1977)], это структура данных, созданная для работы с такими интервалами на числовой оси, концы которых принадлежат *фиксированному* множеству из N абсцисс. Поскольку множество абсцисс фиксировано, то дерево отрезков представляет собой *статическую* структуру по отношению к этим абсциссам, т. е. структуру, на которой не разрешены вставки и удаления абсцисс; кроме того, эти абсциссы можно нормализовать, заменяя каждую из них ее порядковым номером при обходе их слева направо. Не теряя общности, можно считать эти абсциссы целыми числами в интервале $[1, N]$.

Дерево отрезков — это двоичное дерево с корнем. Для заданных целых чисел l и r таких, что $l < r$, дерево отрезков $T(l, r)$ строится рекурсивно следующим образом: оно состоит из корня v с параметрами $B[v] = l$ и $E[v] = r$ (B и E мнемонически соответствуют словам «beginning» (начало) и «end» (конец)), а если $r - l > 1$, то оно состоит из левого поддерева $T(l, \lfloor (B[v] + E[v])/2 \rfloor)$ и правого поддерева $T(\lfloor (B[v] + E[v])/2 \rfloor, r)$. (Корни этих поддеревьев естественно обозначить через ЛСЫН$[v]$ и ПСЫН$[v]$ соответственно.) Параметры $B[v]$

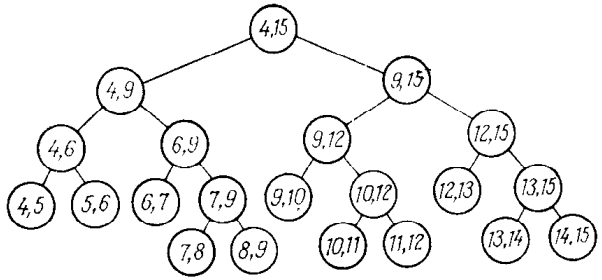

Рис. 1.1. Дерево отрезков $T(4, 15)$.

и $E[v]$ обозначают *интервал* $[B[v], E[v]] \subseteq [l, r]$, связанный с узлом v. Пример дерева отрезков приведен на рис. 1.1. Интервалы, принадлежащие множеству $\{[B[v], E[v]]: v$ — узел $T(l, r)\}$, называются *стандартными интервалами* дерева $T(l, r)$. Стандартные интервалы, принадлежащие листьям $T(l, r)$, называются *элементарными интервалами*[1]). Можно непосредственно убедиться, что $T(l, r)$ сбалансировано (все его листья принадлежат двум смежным уровням) и имеет глубину $\lceil \log_2 (r - l) \rceil$.

Дерево отрезков $T(l, r)$ предназначено для *динамического* запоминания тех интервалов, чьи концы принадлежат множеству $\{l, l+1, \ldots, r\}$ (т. е. допустимы операции вставки и удаления). В частности, при $r - l > 3$ произвольный интервал $[b, e]$ с целыми $b < e$ будет разбит в набор из не более чем $\lceil \log_2 (r - l) \rceil + \lfloor \log_2 (r - l) \rfloor - 2$ стандартных интервалов дерева $T(l, r)$. Фрагментация интервала $[b, e]$ полностью определяется операцией, которая заносит (вставляет) $[b, e]$ в дерево отрезков T, и, значит, обращением ВСТАВИТЬ$(b, e$; корень $(T))$ к следующей процедуре:

[1]) Строго говоря, интервал, связанный с v, это полуоткрытый интервал $[B[v], E[v])$, за исключением узлов самого правого пути в $T(l, r)$, чьи интервалы замкнуты.

```
        procedure ВСТАВИТЬ(b,e;v)
  begin if (b ⩽ B[v]) and (E[v] ⩽ e) then назначить [b, e] узлу v
        else begin if (b < ⌊(B[v] + E[v])/2⌋) then
                       ВСТАВИТЬ(b,e;ЛСЫН[v]);
                   if (⌊(B[v] + E[v])/2⌋ < e) then
                       ВСТАВИТЬ(b,e;ПСЫН[v])
             end
  end.
```

Действие ВСТАВИТЬ$(b, e;$ корень $(T))$ соответствует «маршруту» в T, имеющему следующую общую структуру (рис. 1.2): (возможно, пустой) начальный путь, именуемый $P_{\text{нач}}$, от корня до узла v^*, называемого *развилкой*, из которого выходят два

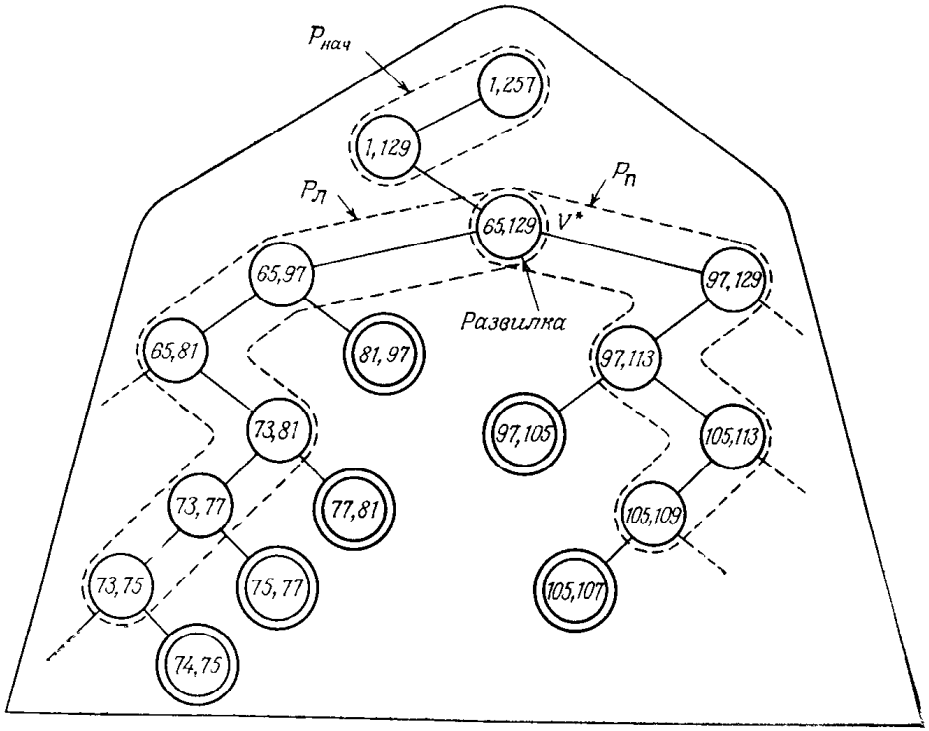

Рис. 1.2. Вставка интервала [74, 107] в $T(1, 257)$. Узлы отнесения окружены дважды.

(возможно, пустых) пути — $P_л$ и $P_п$. Вставляемый интервал относится или полностью к развилке (в этом случае $P_л$ и $P_п$ оба пусты), или ко всем правым сыновьям пути $P_л$, которые сами не на $P_л$, наряду со всеми левыми сыновьями пути $P_п$, которые сами не на $P_п$: при этом определяется фрагментация $[b, e]$ (*узлы отнесения*).

Отнесение интервала к узлу v из T может принимать разные формы в зависимости от потребностей конкретного приложения. Часто все, что надо знать, — это мощность множества интервалов, отнесенных к любому заданному узлу v; ее можно запомнить в единственном неотрицательном целом параметре $C[v]$, обозначающем эту мощность. Тогда отнесение $[b, e]$ к узлу v — это всего лишь

$$C[v] := C[v] + 1.$$

В других приложениях надо сохранить сведения об интервалах, отнесенных к узлу v. Тогда к каждому узлу дерева T добавляется вторичная структура — связный список $\mathscr{L}[v]$, чьи записи являются идентификаторами этих интервалов.

Совершенно симметрична операции ВСТАВИТЬ операция УДАЛИТЬ, выражаемая следующей процедурой (здесь предполагается, что нас интересует только состояние параметра $C[v]$):

```
        procedure УДАЛИТЬ(b,e;v)
  begin if (b ⩽ B[v]) and (E[v] ⩽ e) then C[v] := C[v] − 1
        else begin if (b < ⌊(B[v] + E[v])/2⌋) then
                       УДАЛИТЬ(b,e;ЛСЫН[v]);
                   if (⌊(B[v] + E[v])/2⌋ < e) then
                       УДАЛИТЬ(b,e;ПСЫН[v])
             end
  end.
```

(Заметим, что удаление только ранее вставленных интервалов гарантирует корректность.)

Дерево отрезков — чрезвычайно гибкая структура данных, в чем мы еще сможем убедиться в связи с многочисленными приложениями (гл. 2 и 8). Отметим только, что если надо знать число интервалов, содержащих данную точку x, то простой двоичный поиск в T (т. е. прохождение пути от корня к листу) полностью решает эту задачу.

1.2.3.2. Реберный список с двойными связями

Реберный список с двойными связями (РСДС) особенно удобен для представления планарного графа, уложенного на плоскости[1]) [Muller, Preparata (1978)]. Плоская укладка планарного графа $G = (V, E)$ — это отображение каждой вершины из V в точку на плоскости, а каждого ребра из E

[1]) Планарный граф, уложенный на плоскости, принято называть *плоским*. — *Прим. перев.*