Формулировка задачи. Дано двоичное дерево. Необходимо организовать полный его проход с минимальными затратами дополнительной памяти.

Для полного обхода любого дерева можно легко и просто составить рекурсивную процедуру. Но рекурсия для своей реализации требует достаточно много дополнительной памяти, что не укладывается в требование минимальности затрат. Экономное решение очевидно должно иметь нерекурсивную природу.

Идея алгоритма. Обозначим ветви входящие в текущий узел, как ИСТОЧНИК, ЛЕВАЯ и ПРАВАЯ (ветвь это указатель на узел, соответственно ЛЕВАЯ указывает на дочерний узел слева, и ПРАВАЯ на дочерний узел справа). Ветвь ИСТОЧНИК соединяет данный узел с узлом предком, а ветви ЛЕВАЯ и ПРАВАЯ это соответственно ветви идущие вглубь дерева. Проблема организации обхода очевидно заключается в моментах возврата. Вернувшись в узел, вследствие невозможности продолжать движение вглубь мы можем встретиться с двумя принципиально разными ситуациями. Во-первых, может оказаться, что из данного возможен путь вглубь по другой ветке и во-вторых, может оказаться, что путь вглубь и по ветви ЛЕВАЯ и по ветви ПРАВАЯ исчерпан и необходимо выполнить процедуру возврата. То есть, процедура возврата может либо породить еще одну процедуру возврата либо процедуру движения вглубь.

Для того, чтобы не ошибиться в принятии решения нужна дополнительная информация, о том, что было сделано в момент предыдущего вхождения в узел и именно запоминанием такой информации и занимается механизм рекурсии.

Отправной точкой алгоритм являются три очевидных утверждения. Во-первых, каждый узел хранит информацию о трёх указанных ветвях и во-вторых, возврат в каждый узел двоичного дерева возможен только дважды и в третьих, по каждой ветви дерева движение будет выполняться только дважды: при движении вглубь и в момент возврата. А теперь собственно идея алгоритма Дойча.

Введем для каждого узла одно дополнительное битовое поле (назовем его ФЛАГ) инициализированное нулем. При первом входе (движение вглубь) в узел мы обнаруживаем ФЛАГ равный нулю. Это является сигналом, что возможно движение по ветви ЛЕВАЯ. Мы уходим по ветви ЛЕВАЯ и так как движение по ней из данного узла уже невозможно, то используем ветвь ЛЕВАЯ для запоминания ветви ИСТОЧНИК.

Если мы обнаружили ФЛАГ = 0 выполняя возврат, то это означает, что ветвь ЛЕВАЯ уже пройдена и сейчас она фактически хранит информацию о ветви ИСТОЧНИК. Тогда мы уходим вглубь по ветви ПРАВАЯ, значение ФЛАГА меняем на 1. И сейчас ветвь ИСТОЧНИК можно запомнить в ветви ПРАВАЯ.

Если же выполняя возврат, мы обнаруживаем значение ФЛАГА = 1 это означает, что обе ветви ведущие вглубь уже пройдены и необходимо выполнить процедуру возврата еще раз, а информация для этого возврата хранится в ветви ПРАВАЯ.

Описанная идея содержит один существенный недостаток. Запоминая в ЛЕВОЙ (ПРАВОЙ) ветви информацию о ветви ИСТОЧНИК, информацию о ЛЕВОЙ (ПРАВОЙ) ветви мы теряем. Это приводит к тому, что с обходом дерева оно практически уничтожается.

Указанный недостаток легко исправить, если заметить, что ЛЕВАЯ (ПРАВАЯ) ветвь очередного узла, есть ИСТОЧНИК его дочернего, то есть того, в который будет осуществлен переход из текущего. Иначе говоря информация о ЛЕВОЙ (ПРАВОЙ) ветви сохраняется в дочернем узле и следовательно её вполне можно восстановить. Подобности смотрите в алгоритме, существо же алгоритма в оперировании четырьмя указателями: двумя глобальными по отношению к дереву, это ИСТОЧНИК и ТЕКУЩИЙ УЗЕЛ (указывающий на текущую точку обхода дерева) и двумя локальными, хранимыми в структуре дерева, это ЛЕВАЯ ветвь и ПРАВАЯ ветвь.

Алгоритм

ТЕКУЩИЙ УЗЕЛ = КОРНЕВОМУ узлу
ИСТОЧНИК = КОРНЕВОМУ узлу
Цикл, действие которого завершается тогда, когда ТЕКУЩИЙ УЗЕЛ это корневой узел и его ФЛАГ = 1

Если Флаг = 0 то

Если есть дочерний узел то выполняется процедура «Переход на дочерний узел»

Иначе

выполняется процедура «Возврат»

Если ФЛАГ = 0 то

ФЛАГ = 1

выполняется процедура «Переход на дочерний узел»

Иначе

выполняется процедура «Возврат»

Если ФЛАГ = 0 то

ФЛАГ =1

выполняется процедура «Переход на дочерний узел»

Процедура «Переход на дочерний узел»

Если ФЛАГ =0 то

ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ = ЛЕВАЯ ветвь
ЛЕВАЯ ветвь = ИСТОЧНИК
ТЕКУЩИЙ УЗЕЛ = ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ

Иначе

ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ = ПРАВАЯ ветвь
ПРАВАЯ ветвь = ИСТОЧНИК
ТЕКУЩИЙ УЗЕЛ = ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ

Процедура «Возврат»

Если возврат осуществляется на корень

То

ЛЕВАЯ ветвь = ТЕКУЩИЙ УЗЕЛ
ТЕКУЩИЙ УЗЕЛ = КОРНЕВОМУ узлу

Иначе

ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ = ТЕКУЩИЙ УЗЕЛ
ТЕКУЩИЙ УЗЕЛ = ИСТОЧНИК
Если ФЛАГ =0 то

ИСТОЧНИК = ЛЕВАЯ ветвь
ЛЕВАЯ ветвь = ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ

Иначе

ИСТОЧНИК = ПРАВАЯ ветвь
ПРАВАЯ ветвь = ДОПОЛНИТЕЛЬНЫЙ УКАЗАТЕЛЬ