Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 14

ПОРОЖДЕННОЕ ДЕРЕВО РЕКУРСИИ И ЕГО ХАРАКТЕРИСТИКИ

МЕТОД АНАЛИЗА ТРУДОЕМКОСТИ РЕКУРСИВНЫХ АЛГОРИТМОВ

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

1. Дерево рекурсии

Если мы при схематичном изображении рекурсивных вызовов и возвратов хотим показать только соподчинение рекурсивных вызовов, то схема вызовов для факториала (см. лекция 13) может быть преобразована в унарное дерево, в вершинах которого мы указываем аргумент, с которым порожден следующий рекурсивный вызов.

Мы получаем графическую интерпретацию последовательности рекурсивных вызовов, порожденную данным алгоритмом. В дальнейшем будем понимать под деревом рекурсии граф, имеющий древовидную структуру, вершины которого отражают рекурсивную функцию или процедуру с данным аргументом, а дуги (ребра) — непосредственно рекурсивные вызовы и возвраты.

Дерево рекурсии — это граф, отражающий соподчинение рекурсивных вызовов, порожденных данным алгоритмом при данном входе.

2. Анализ трудоемкости вызова рекурсивной функции

Поскольку передаваемые в функцию фактические параметры, так и возвращаемые из них значения помещаются в программный стек специальными командами процессора, то мы считаем эти операции со стеком базовыми в нашей модели вычислений.

При подсчете трудоемкости вызова учитываем, что при вызове процедуры или функции в стек помещается адрес возврата, состояние необходимых регистров процессора, состояние *локальных ячеек* вызывающей функции или процедуры, адреса возвращаемых значений и передаваемые параметры. После этого выполняется переход по адресу на вызываемую процедуру, которая извлекает переданные фактические параметры. При завершении работы вызываемая процедура восстанавливает регистры, локальные ячейки, выталкивает из стека адрес возврата и осуществляет переход по этому адресу.

Для анализа трудоемкости механизма вызова-возврата для рекурсивной процедуры введем следующие обозначения

p — количество передаваемых фактических параметров,

r — количество сохраняемых в стеке регистров,

k — количество возвращаемых по адресной ссылке значений,

l — количество локальных ячеек процедуры, сохранение которых необходимо для обеспечения реентерабельности.

Поскольку каждый объект в некоторый момент помещается в стек, и в какой-то момент выталкивается из него, то трудоемкость рекурсивной процедуры на один вызов-возврат — $f_R(1)$ в базовых операциях составит:

$$f_R(1) = 2 \cdot (p + k + r + l + 1).$$

Дополнительная единица в формуле учитывает операции с адресом возврата.

Для рекурсивной функции мы должны дополнительно учесть еще одну ячейку стека, через которую передается значение функции. Мы обозначим этот параметр через f, считая, что его значение равно единице, тогда трудоемкость на один вызов/возврат рекурсивной функции может быть определена следующим образом:

$$f_R(1) = 2 \cdot (p + k + r + f + l + 1),\ f = 1.$$

Анализ трудоемкости рекурсивных алгоритмов в части совокупной трудоемкости самого рекурсивного вызова-возврата можно выполнять разными способами в зависимости от того, как формируется итоговая сумма базовых операций — либо отдельно по цепочкам рекурсивных вызовов и возвратов, либо совокупно по вершинам дерева рекурсии.

В любом случае необходима информация о структуре дерева рекурсии.

3. Анализ дерева рекурсии

При теоретическом построении ресурсных функций рекурсивного алгоритма необходимо учесть ряд ресурсных затрат и особенностей рекурсивной реализации, а именно:

— ресурсные затраты на обслуживание рекурсивных вызовов-возвратов, передачу параметров и возврат значений рекурсивных функций (ресурсные затраты на обслуживание рекурсии);

— специфику фрагмента останова рекурсии, приводящую к необходимости отдельного учета ресурсных затрат в листьях порожденного дерева рекурсии.

Учет этих особенностей при теоретическом анализе рекурсивных алгоритмов приводит к необходимости получить функциональные зависимости общего количества вершин дерева рекурсии и количества его внутренних вершин и листьев, от характеристик множества входных данных.

Первым и основным этапом метода является исследование порожденного дерева рекурсии. В предположении, что D — конкретный вход алгоритма, введем следующие обозначения для характеристик дерева рекурсии, порожденного рекурсивным алгоритмом:

$R(D)$ — общее количество вершин дерева рекурсии для входа D;

$R_V(D)$ — количество внутренних вершин дерева для входа D;

$R_L(D)$ — количество листьев дерева рекурсии для входа D;

$H_R(D)$ — максимальная глубина дерева рекурсии (максимальное по всем листьям дерева количество вершин в пути от корня дерева до листа), тогда очевидно, что справедливы соотношения

$$R(D) = R_V(D) + R_L(D), H_R(D) \leq R_V(D) + 1.$$

Основной интерес представляет получение таких характеристик в *аналитическом виде*, т.е. как функций от каких-либо характеристик входа D.

4. Метод анализа рекурсивных алгоритмов

Трудоемкость рекурсивного алгоритма A на конкретном входе D определяется трудоемкостью обслуживания дерева рекурсии, зависящей от общего количества его вершин, и трудоемкостью продуктивных вычислений, выполненных во всех вершинах дерева рекурсии. Пусть:

$f_R(D)$ — трудоемкость порождения и обслуживания дерева рекурсии,

$f_C(D)$ — трудоемкость продуктивных вычислений алгоритма, тогда

$$f_A(D) = f_R(D) + f_C(D).$$

Если функция $R(D)$ известна, на обслуживание одного рекурсивного вызова затрачивается фиксированное количество базовых операций $f_R(1)$, то

$$f_R(D) = R(D) \cdot f_R(1).$$

При подсчете $f_C(D)$ необходимо учесть, что для листьев рекурсивного дерева алгоритм будет выполнять непосредственное вычисление значений.

Трудоемкость во внутренних вершинах отлична от трудоемкости в листьях.

Пусть:

$f_{CV}(D)$ — трудоемкость вычислений во внутренних вершинах,

$f_{CL}(D)$ — трудоемкость вычислений в листьях дерева рекурсии

$$f_C(D) = f_{CV}(D) + f_{CL}(D).$$

Обозначим через $f_{CL}(1)$ трудоемкость алгоритма при останове рекурсии, заметим, что, как правило, значение $f_{CL}(1)$ может быть достаточно легко получено, т.к. останов рекурсии выполняется на малых значениях аргумента.

Зная количество листьев рекурсивного дерева, можно определить $f_{CL}(D)$

$$f_{CL}(D) = R_L(D) \cdot f_{CL}(1).$$

Во внутренних вершинах дерева рекурсии выполняются некоторые действия, связанные с подготовкой параметров следующих рекурсивных вызовов и обработкой возвращаемых результатов. Трудоемкость такой обработки может зависеть как от обрабатываемых в этой вершине данных, так и от положения вершины в дереве рекурсии. С целью учета этой зависимости введем нумерацию внутренних вершин, начиная с корня, по уровням дерева. Заметим, что число уровней внутренних вершин в дереве на единицу меньше глубины рекурсии $H_R(D)$. Пусть m есть номер уровня $m = \overline{1, H_R(D) - 1}$, а k — номер вершины на уровне $k = \overline{1, K(m)}$, где $K(m)$ — количество внутренних вершин на уровне m, заметим, что неполное дерево на уровне k может содержать как внутренние вершины, так и листья. С учетом такой нумерации обозначим вершины дерева через

$$v_{mk},\ m = \overline{1, H_R(D) - 1};\ k = \overline{1,\ K(m)},$$

при этом очевидно, что

$$\sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(m)} 1 = R_V(D).$$

Обозначим трудоемкость продуктивных вычислений в вершине v_{mk} через $f_{CV}(v_{mk})$, тогда формула для трудоемкости продуктивных вычислений во внутренних вершинах дерева рекурсии имеет вид

$$f_{CV}(D) = \sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(m)} f_{CV}(v_{mk}).$$

Заметим, что в случае, когда значения функции $f_{CV}(v_{mk})$ не зависят от номера вершины дерева рекурсии, т. е. трудоемкость продуктивных вычислений в вершинах не зависит от данных, то, обозначая трудоемкость продуктивных вычислений для любой внутренней вершины дерева через $f_{CV}(v)$, имеем

$$f_{CV}(D) = R_V(D) \cdot f_{CV}(v).$$

Подставляя полученные результаты, окончательно получаем формулу для определения трудоемкости рекурсивного алгоритма на основе метода подсчета вершин дерева рекурсии в общем случае

$$f_A(D) = R(D) \cdot f_R(1) + R_L(D) \cdot f_{CL}(1) + \sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(m)} f_{CV}(v_{mk}),$$

и в частном случае, когда трудоемкость продуктивных вычислений для любой внутренней вершины дерева рекурсии одинакова

$$f_A(D) = R(D) \cdot f_R(1) + R_L(D) \cdot f_{CL}(1) + R_V(D) \cdot f_{CV}(v).$$

Можно рассмотреть дополнительную характеристику трудоемкости рекурсивного алгоритма — долю операций обслуживания дерева рекурсии. Обозначая ее через $F_R(D)$ имеем

$$F_R(D) = \frac{f_R(D)}{f_A(D)}, \quad 0 < F_R(D) < 1.$$

Значение $F_R(D)$ показывает, насколько трудоемкость обслуживания дерева рекурсии значима в общей трудоемкости рекурсивного алгоритма.

Для тех алгоритмов, для которых значение $F_R(D)$ велико (близко к единице) применение рекурсии, очевидно, не является целесообразным. В ряде случаев это отношение убывает с увеличением длины входа, тогда рекурсивная реализация становится приемлемой для б*о*льших длин.

4. Пример дерева рекурсии

Это реальный пример — попробуйте получить аналитическое решение для общего количества вершин дерева рекурсии $R(n, 2) =?$ А если подумать, то можно получить и большее, а именно аналитическое решение для $R(n, k) =?$