**39. Оценка рекурсивных алгоритмов. Метод деревьев рекурсии.**

Рекурсивные алгоритмы относятся к классу алгоритмов с высокой ресурсоемкостью, так как при большом количестве самовызовов рекурсивных функций происходит быстрое заполнение стековой области. Кроме того, организация хранения и закрытия очередного слоя рекурсивного стека являются дополнительными операциями, требующими временных затрат. На трудоемкость рекурсивных алгоритмов влияет и количество передаваемых функцией параметров.

Рассмотрим один из методов анализа трудоемкости рекурсивного алгоритма, который строится на основе подсчета вершин рекурсивного дерева. Для оценки трудоемкости рекурсивных алгоритмов строится **полное дерево рекурсии**. Оно представляет собой *граф*, вершинами которого являются наборы *фактических параметров* при всех вызовах функции, начиная с первого обращения к ней, а *ребрами* – пары таких наборов, соответствующих взаимным вызовам. При этом вершины дерева рекурсии соответствуют фактическим вызовам рекурсивных функций. Следует заметить, что одни и те же наборы параметров могут соответствовать разным вершинам дерева. *Корень полного дерева рекурсивных вызовов* – это *вершина* полного дерева рекурсии, соответствующая начальному обращению к функции.

Важной характеристикой рекурсивного алгоритма является **глубина рекурсивных вызовов** – наибольшее одновременное количество рекурсивных обращений функции, определяющее максимальное количество слоев рекурсивного стека, в котором осуществляется хранение отложенных вычислений. Количество элементов полных рекурсивных обращений всегда не меньше глубины рекурсивных вызовов. При разработке рекурсивных программ необходимо учитывать, что глубина рекурсивных вызовов не должна превосходить максимального размера стека используемой вычислительной среды.

При этом **объем рекурсии** – это одна из характеристик сложности *рекурсивных вычислений* для конкретного набора параметров, представляющая собой количество вершин полного рекурсивного дерева без единицы.

Будем использовать следующие обозначения для конкретного входного параметра D:

R(D) – общее число вершин дерева рекурсии,

RV(D) – объем рекурсии без листьев (*внутренние вершины*),

RL(D) – количество *листьев дерева* рекурсии,

HR(D) – глубина рекурсии.

Будем использовать следующие обозначения для конкретного входного параметра D:

R(D) – общее число вершин дерева рекурсии,

RV(D) – объем рекурсии без листьев (*внутренние вершины*),

RL(D) – количество *листьев дерева* рекурсии,

HR(D) – глубина рекурсии.

Например, для вычисления n -го члена последовательности Фибоначчи разработана следующая *рекурсивная функция*:

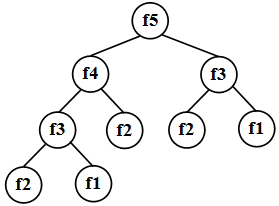
intFib(int n){ //n – номер члена последовательности

if(n<3) return 1; //база рекурсии

returnFib(n-1)+Fib(n-2); //декомпозиция

}

Тогда полное *дерево* рекурсии для вычисления пятого члена последовательности Фибоначчи будет иметь вид ([рис. 34.1](https://www.intuit.ru/studies/courses/648/504/lecture/11462?page=1#image.34.1)):



**Рис. 34.1.**Полное дерево рекурсии для пятого члена последовательности Фибоначчи

Характеристиками рассматриваемого метода оценки алгоритма будут следующие величины.

|  |  |
| --- | --- |
| **D = 5** | **D = n** |
| R(D)=9 | R(D)=2fn-1 |
| RV(D)=4 | RV(D)=fn-1 |
| RL(D)=5 | RL(D)=fn |
| HR(D)=4 | HR(D)=n-1 |
|  |  |

суть метода

- строим дерево выполнения подзадач

- считаем кол-во операций на каждом уровне

-считаем сумму операций по всем уровням

Особенности

-позволяет легко подтвердить догадку

-подходит для методов разделяй и властвуй (например, быстрая сортировка)