Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 15

МЕТОД ДЕКОМПОЗИЦИИ КАК МЕТОД РАЗРАБОТКИ РЕКУРСИВНЫХ АЛГОРИТМОВ

Лектор: Михаил Васильевич Ульянов,

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

1. Замечания о термине «метод разработки алгоритмов»

Методы, о которых пойдет речь, не являются методами в строго математическом понимании этого термина. Термин «метод» по отношению к разработке алгоритмов — это скорее дань исторической традиции, а не констатация гарантии быстрой и эффективной разработки. Под этим термином понимаются *приемы или способы*, пользуясь которыми может быть получится построить алгоритм решения задачи при условиях:

— во-первых, сам метод теоретически применим к этой задаче;

— и, во-вторых, при условии, что Вам удастся адаптировать этот метод к ее особенностям.

Речь скорее идет о некоторых общих подходах, не содержащих рецептов конкретного применения, и мы снова констатируем интеллектуальный аспект в процессе разработки алгоритмов.

2. Идея метода декомпозиции

Чтобы перенести сотню кирпичей с одного места на другое, мы, скорее всего, будем переносить за один раз несколько кирпичей, чем всю сотню сразу.

Таким образом, вместо того чтобы решать задачу сразу, мы несколько раз решаем более простую задачу с понижением размерности — собственно говоря, эта интуитивно здравая идея и лежит в основе метода декомпозиции. Другое название этого метода (американское) — метод «разделяй и властвуй».

Таким образом, мы пытаемся каким-то образом разделить задачу на все более и более простые задачи (задачи меньшей размерности), пока не дойдем до такой размерности (длины входа), при которой задача решается элементарно. Следующий шаг — объединение полученных решений в решение для исходной размерности. Общая идея или схема метода может быть описана как последовательность следующих шагов:

2. Этапы метода декомпозиции

1. *Способ разделения задачи*. На этом этапе выбирается способ разделения задачи на некоторое количество подзадач меньшей размерности.

2. *Этап рекурсии для полученных подзадач*. Это рекурсивный шаг — мы рассматриваем каждую полученную подзадачу как задачу определенной размерности, и разделяем ее снова на собственные подзадачи, выполняя этап 1 (рекурсия), до тех пор, пока мы не получим такую размерность, при которой решение может быть получено непосредственно.

3. *Этап останова рекурсии*. На этом шаге выполняется непосредственное решение полученных подзадач для малых размерностей.

4. *Этап объединения решений*. На этом шаге полученные решения подзадач меньшей размерности при возврате в точку рекурсивного вызова объединяются в решение задачи текущей размерности.

3. Замечания о применимости метода декомпозиции

Во-первых, идея метода ничего не говорит о том, как эти шаги должны быть реализованы в конкретной задаче — нужна адаптация метода к задаче.

Во-вторых, условием применения метода является свойство *аддитивности решений*, полученных для подзадач меньшей размерности. Необходимо доказать, что задача обладает тем свойством, что объединение решений допустимо. Для задачи коммивояжера простое объединение двух оптимальных путей для графов половинной размерности вообще не является решением исходной задачи.

В-третьих, для одной и той же задачи могут быть предложены различные идеи ее разделения, и здесь необходимо оценить к каким затратам приводит каждая из этих идей на этапе объединения решений, который, как правило, и является наиболее сложным.

4. Преимущества метода декомпозиции

Основное преимущество метода декомпозиции состоит в том, что он позволяет получить алгоритмы с хорошими асимптотическими оценками трудоемкости. А.А. Карацуба (1962 г.) — алгоритм умножения длинных целых чисел с оценкой, лучше, чем n^2, W. Strassen (1969) — алгоритм умножения квадратных матриц с оценкой, лучше, чем n^3. Соответствующее обоснование асимптотических оценок трудоемкости алгоритмов, разработанных методом декомпозиции — в следующей лекции.

Приведем примеры применения метода декомпозиции. Исторически одним из первых применений метода считается алгоритм сортировки слиянием, приписываемый Дж. фон Нейману. Рассмотрим задачу сортировки массива чисел из n элементов, и обсудим шаги метода декомпозиции, применительно к этой задаче.

5. Сортировка слиянием

1. Шаг разделения задачи. Мы организуем деление на две подзадачи — массив разделяется на два массива, содержащих $\lfloor n/2 \rfloor$ и $\lceil n/2 \rceil$ элементов.

2. Шаг решения полученных подзадач. Если условие останова рекурсии не выполнено, то полученные массивы вновь разделяются на два массива — это шаг порождения дерева рекурсии.

3. Шаг останова рекурсии. В классическом изложении останов рекурсии происходит при длине массива равном единице — массив из одного числа очевидно отсортирован.

4. Шаг объединения решений. На цепочке рекурсивных возвратов мы получаем два отсортированных массива, которые должны быть объединены в один. Этот шаг выполняется специальным алгоритмом слияния сортированных массивов, по которому и получил название сам алгоритм сортировки слиянием.

Эти шаги порождают асимптотически оптимальный алгоритм сортировки для задачи сортировки сравнениями со сложностью $\Theta(n \cdot ln\, n)$. Из-за большого коэффициента при главном порядке область его рациональной применимости начинается с длин массива порядка 100 и более точно определяется особенностями реализации, в основном — алгоритмом слияния двух сортированных массивов.

6. Умножение матриц

Второй пример — задача умножения квадратных матриц. Для изложения идеи мы рассмотрим случай, когда линейный размер квадратной матрицы — n является степенью двойки. Шаги метода декомпозиции могут быть реализованы следующим образом:

1. Шаг разделения задачи. Мы организуем деление пополам линейного размера матрицы. Поскольку значение n является степенью двойки, то на любом шаге деления значение $n/2$ является целым числом. Таким образом, каждая из перемножаемых матриц делится на четыре подматрицы

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} r & s \\ t & u \end{pmatrix},$$

где подматрицы результата задаются уравнениями

$$r = a \times e + b \times g;$$

$$s = a \times f + b \times h;$$

$$t = c \times e + d \times g;$$

$$u = c \times f + d \times h. \qquad (1)$$

Эти уравнения приводят к необходимости решения *восьми* подзадач размерностью $n/2$. Операции умножения в (1) есть операции умножения матриц.

2. Шаг решения полученных подзадач. Если после разделения полученные матрицы размерностью $n/2 \times n/2$ не являются матрицами размерности два, то полученные матрицы вновь разделяются на четыре подматрицы — это шаг порождения дерева рекурсии.

3. Шаг останова рекурсии. Если на текущем вызове алгоритма матрицы имеют размер 2×2, то результат может быть вычислен непосредственно по формуле (1), обозначения в которой в данном случае интерпретируются как обозначения элементов матриц.

4. Шаг объединения решений. По цепочке рекурсивных возвратов мы получаем восемь результирующих матриц, имеющих размер $n/2 \times n/2$ относительно размера текущей матрицы. В соответствии с (1) нам необходимо выполнить сложение матриц и заполнение соответствующих позиций матрицы результата.

Элементарный анализ этих шагов позволяет получить рекуррентное соотношение, задающее суммарное количество выполненных алгоритмом операций умножения и сложения над числами — элементами матриц

$$S(n) = 8 \cdot S(n/2) + \Theta(n^2).$$

Асимптотическая оценка решения этого рекуррентного соотношения $S(n) = \Theta(n^3)$, так что в таком подходе алгоритм, разработанный методом декомпозиции, оказывается по порядку таким же, как и обычный алгоритм умножения матриц. Тем не мене, именно этот метод позволил W. Strassen'у получить алгоритм с асимптотически лучшей оценкой — $\Theta\left(n^{log_2 7}\right)$. Если Вы придумаете, как вместо восьми умножений матриц размером $n/2 \times n/2$ можно выполнить только семь, то Вы получите результат, аналогичный алгоритму Штрассена.

7. Выпуклый охватывающий контур

Рассмотрим задачу о построении выпуклого охватывающего контура на n точках плоскости. Придумайте:

1. Как разбить задачу на подзадачи?
2. При какой размерности решение элементарно?
3. Как объединять решения?

В целом с использованием метода декомпозиции в настоящее время получены эффективные алгоритмы решения целого ряда задач. Основная их особенность — «плата» за асимптотически лучшую функцию в оценке сложности б***о***льшим коэффициентом при главном порядке. Тем самым эти алгоритмы становятся рационально применимыми, начиная с относительно больших размерностей.

4. Литература

1. Баррон Д. Рекурсивные методы в программировании. — М.: Мир, 1974. – 79 с.

2. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-ое издание : Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1296 с.

3. Хаггарти Р. Дискретная математика для программистов. — М.: Техносфера, 2005. — 400с.