Анализ рекурсивных алгоритмов. Стек вызовов функций. Виды рекурсии. Решение рекуррентных уравнений. Основная теорема (master method). Анализ эффективности алгоритма сортировки слиянием.

Рекурсивная функция (*recursive function*) — функция, в теле которой присутствует вызов самой себя. Алгоритм, основанный на таких функциях, называется **рекурсивным алгоритмом** (*recursive algorithm*).

Рекурсивный алгоритм можно охарактеризовать деревом рекурсивных вызовов (*recursiontree*). В таком дереве каждый узел соответствует вызову функции. Время выполнения рекурсивного алгоритма определяется числом узлов в его дереве рекурсивных вызовов. Точнее говоря, суммарным временем выполнения всех узлов– временем реализации всех рекурсивных вызовов.

Анализ эффективности рекурсивных алгоритмов сложнее анализа итеративных алгоритмов. Трудности здесь связаны с необходимостью оценивания высоты дерева рекурсивных вызовов.

Системный стек:

Системный стек (*stack*) — память, предназначенная для хранения адресов возврата из функций, локальных переменных и передачи аргументов в функции. Рекурсивные функции могут занимать значительную часть стековой памяти для хранения адресов возврата. Стек имеет конечный размер.

```
$ ulimit -s
-8192 Mb (по умолчанию)
```

Виды рекурсии

- **Линейная рекурсия** (*linear recursion*) – в функции присутствует единственный рекурсивный вызов самой себя.
- **Древовидная рекурсия** (*нелинейная, non-linear recursion*) – в функции присутствует несколько рекурсивных вызовов.

Решение рекуррентных уравнений. Анализ эффективности алгоритма сортировки слиянием.

- Время T(n) работы алгоритма включает время сортировки левого подмассива длины $\lceil n / 2 \rceil$ и правого – с числом элементов $\lfloor n / 2 \rfloor$, а также время $\Theta(n)$ слияния подмассивов после их рекурсивного упорядочивания

$$T(n) = T(\lceil n / 2 \rceil) + T(\lfloor n / 2 \rfloor) + \Theta(n)$$

- Необходимо решить это рекуррентное уравнение – получить выражение для T(n) без рекуррентности

Для удобства считаем n степенью двойки. Тогда уравнение принимает вид:

T(n)=2T(n/2)+Θ(n).T(n)=2T(n/2)+Θ(n).

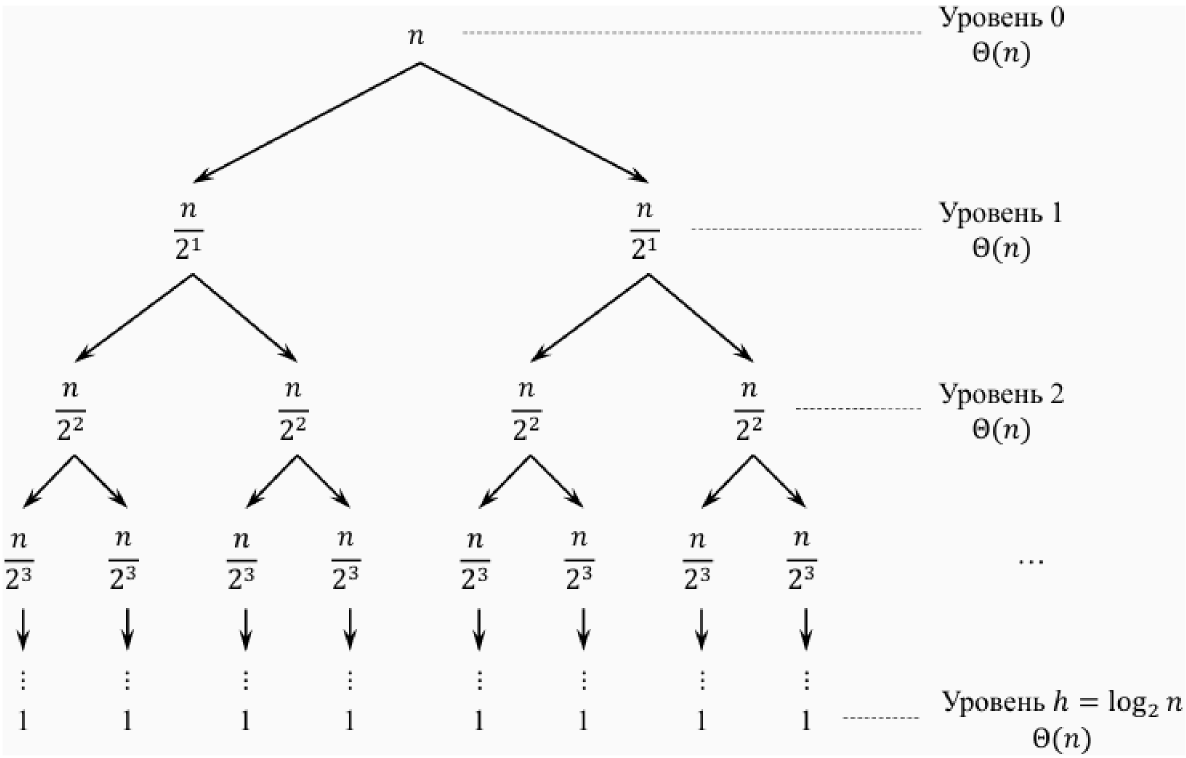

Рис. 2.4. Дерево рекурсивных вызовов алгоритма сортировки слиянием массива из n элементов (n равно степени двойки).

Высота дерева:

На каждом уровне массив делится пополам. Разбиение заканчивается при длине подмассива 1:

$$\frac{n}{2^h} = 1 \quad \Rightarrow \quad h = \log_2 n.$$

Таким образом, высота дерева: Θ(logn).

В общем случае на каждом уровне $i \in \{0, 1, \ldots, \log_2 n\}$ находится 2^i узлов. Каждый узел требует выполнения $n/2^i$ операций. Вычислим сумму операций всех узлов на всех уровнях

$$T(n) = \sum_{i=0}^{h} 2^i \frac{n}{2^i} = \sum_{i=0}^{h} n = (h+1)n = n \log_2 n + n = \Theta(n \log n).$$

Вычислительная сложность сортировки слиянием в худшем случае равна **Θ(nlogn).** Сложность по памяти алгоритма есть **Θ(n)**, так как слияние требует создания копии сортируемого массива.

Основной метод и теорема

- Рассмотрим решение рекуррентных уравнений, когда исходную задачу размера n можно разделить на $a \geq 1$ подзадач размера n / b
- Будем считать, что для решения задачи размера 1 требуется время O(1)
- Декомпозиция задачи размера n и комбинирование (слияние) решений подзадач требует $f(n)$ единиц времени
- Тогда время $T(n)$ решения задачи размера n можно записать как

$$T(n) = aT(n / b) + f(n),$$

где $a \geq 1$, $b > 1$

- Записанное уравнение называется **обобщенным рекуррентным уравнением декомпозиции** (*general divide-and-conquer recurrence*)
- Решением этого уравнения является порядок роста функции $T(n)$, который определяется из следующей **основной теоремы** (*master theorem*)

27

- **Теорема.** Если в обобщенном рекуррентном уравнении декомпозиции

$$f(n) = \Theta(n^d), \text{ где } d \geq 0, \text{ то}$$

$$T(n) = \begin{cases} \Theta(n^d), & \text{если } a < b^d, \\ \Theta(n^d \log n), & \text{если } a = b^d, \\ \Theta(n^{\log_b a}), & \text{если } a > b^d. \end{cases}$$

- **Пример 1.** В рекуррентном уравнении алгоритма сортировки слиянием

$$T(n) = 2T(n / 2) + \Theta(n)$$

- $a = 2$, $b = 2$, $f(n) = \Theta(n)$ и $d = 1$
- Следовательно, имеем случай $a = b^d$
- Тогда, следуя теореме, вычислительная сложность сортировки слиянием в худшем случае равна

$$T(n) = \Theta(n^d \log n) = \Theta(n \log n)$$

28
