Алгоритмы и структуры данных

ЛЕКЦИЯ 2

Анализ рекурсивных алгоритмов

Валенда Н.А.

Кафедра Программной инженерии,
факультет КН, ХНУРЕ

Рекурсия

Рекурсивный алгоритм – это алгоритм, в описании которого содержится обращение к самому себе.

Рекурсивная функция – функция, в теле которой присутствует вызов самой себя.

Рекурсия – фундаментальное математическое понятие. Она предполагает пошаговую организацию вычислительного процесса, где вычисления производятся по одному и тому же алгоритму, но каждый раз с меньшим объемом данных. Предыдущий этап не может завершиться, т.к. его результат вычисляется через результат этого же алгоритма, но с меньшим объемом данных.

Такая организация вычислительного процесса предполагает получение результата на последнем шаге. Затем производится возврат к предыдущим шагам до получения окончательного результата.

Пример рекурсии. Вычисление факториала

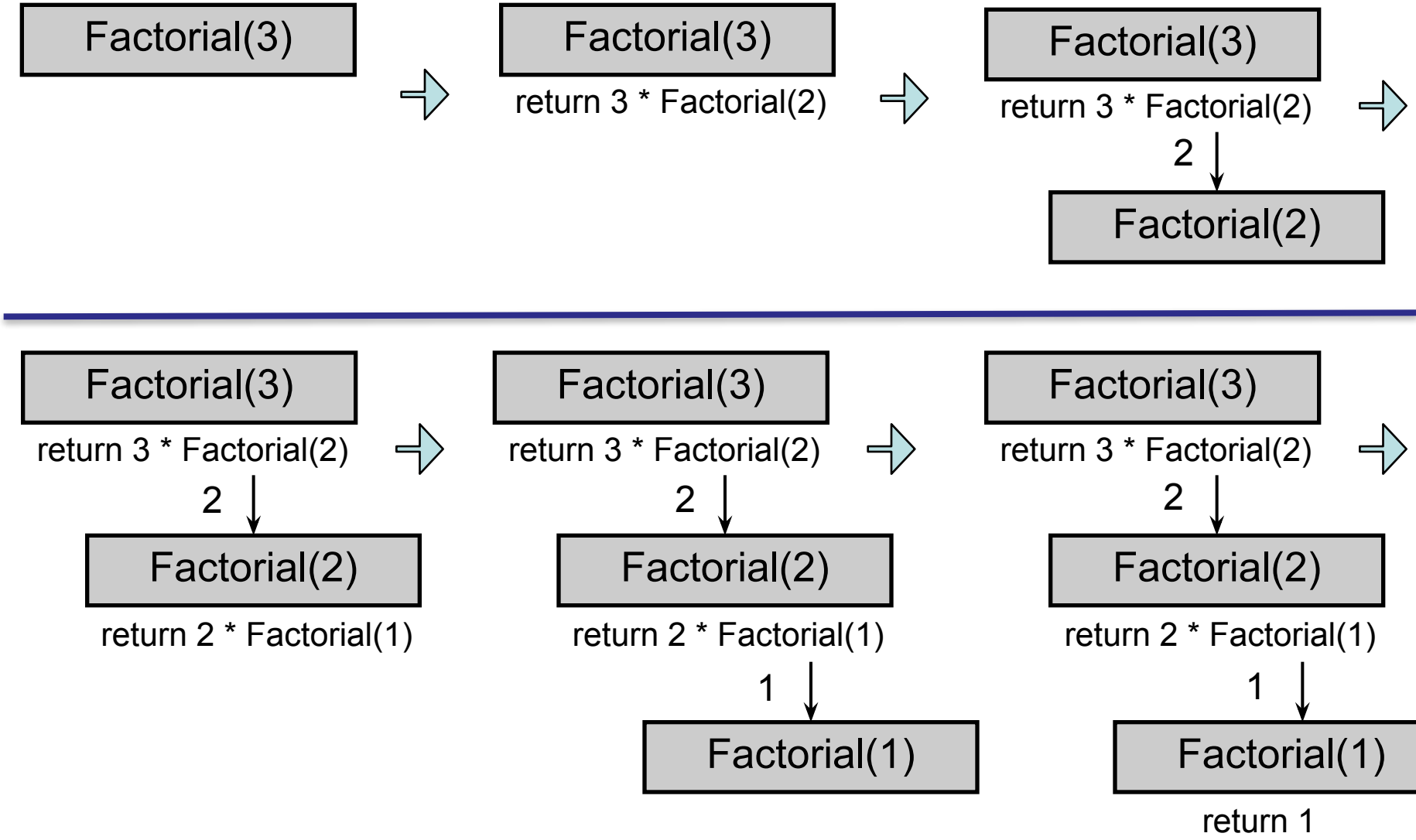
***n*!** - факториал натурального числа *n* определяется как произведение всех натуральных чисел от 1 до *n* включительно.

Формула: ***n*!= *n**(*n-1)*!**

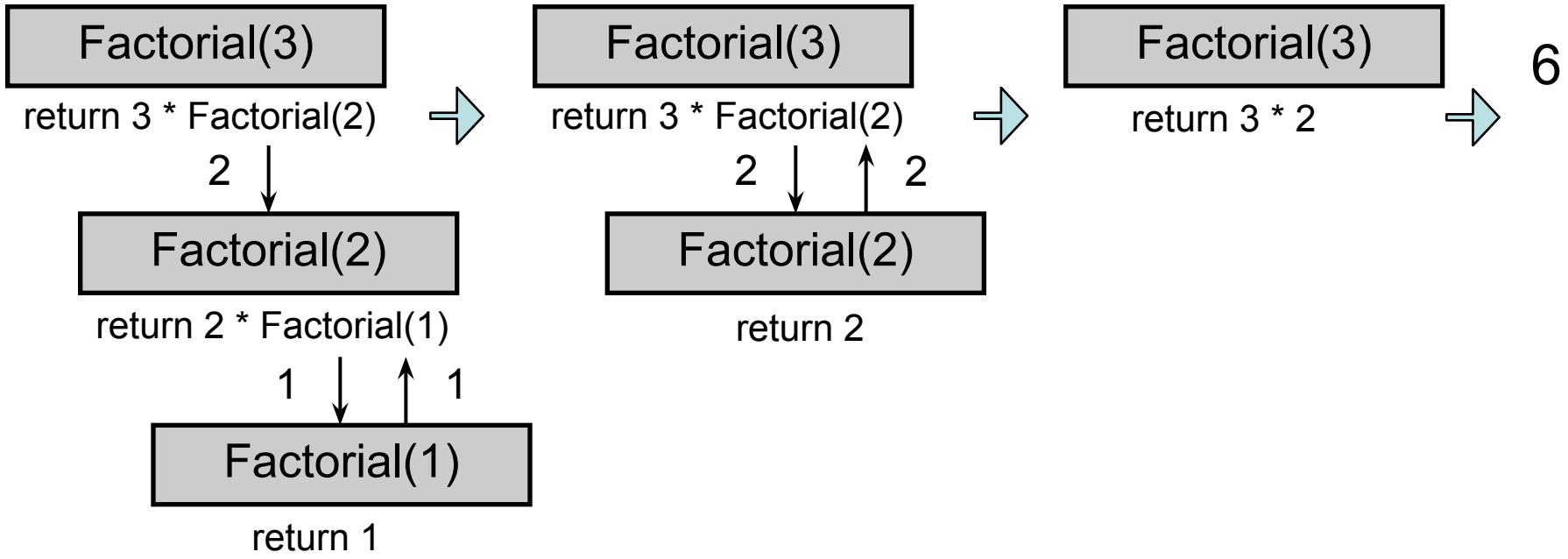
***0*!=1**

```
static int Factorial(int x)
{
    if (x == 1)
    {
        return 1;
    }
    else
    {
        return x * Factorial(x - 1);
    }
}
```

Рекурсивные вызовы



Рекурсивные вызовы



Построение оценки для рекурсивного алгоритма

```
static int Factorial(int x)                T(n)
{
    if (x == 1)                            1
    {
        return 1;                          1
    }
    else                                   1
    {
        return x * Factorial(x - 1);       T(n-1)
    }
}
```

$T(n)=T(n-1)+\Theta(1)$

Методы решения рекуррентных отношений

- Метод итераций
- Дерево рекурсии
- Основная теорема

Метод итераций

- На основании формулы T(n) записываем формулу для меньшего элемента, находящегося в правой части формулы, например T(n-1).
- Подставляем правую часть полученной формулы в исходную формулу
- Выполняем первые два шага, пока не развернем формулу в ряд без функции T(n)
- Оценим полученный ряд на основании арифметической или геометрической прогрессии

Суммы прогрессий

Сумма n - первых членов арифметической прогрессии $S_n=a_1+a_2+\ldots+a_n$ может быть найдена по формуле $$S_n = \frac{a_1 + a_n}{2} \cdot n$$

где a_1 — первый член прогрессии, a_n — член с номером n , n — количество суммируемых членов.

Числовая последовательность, каждый член которой, начиная со второго, равен предыдущему, умноженному на постоянное для этой последовательности число q , называется *геометрической прогрессией*. Число q называется *знаменателем прогрессии*. Любой член геометрической прогрессии вычисляется по формуле: $b_n = b_1 \ q^{n-1}$.

Сумма n первых членов геометрической прогрессии вычисляется по формуле:

$$S_n = \frac{b_1(1-q^n)}{1-q}.$$

Бесконечно убывающая геометрическая прогрессия. Это геометрическая прогрессия, у которой $|q| < 1$. *Сумма членов бесконечно убывающей геометрической прогрессии* вычисляется по формуле:

$$S = \frac{b_1}{1-q}.$$

Метод итераций. Вычисление Factorial

$T(n)=T(n-1)+1$,
$T(1)=1$

$T(n)=\theta(g(n))$, $g(n)=?$

$T(n-1)=T(n-2)+1$
$T(n-2)=T(n-3)+1$

$T(n)= T(n-1)+1=$
$T(n-2)+1+1=$
$T(n-3)+1+1+1=\ldots$
$T(1) +n-1= n$

$T(n)=\theta(n)$

Асимптотическая оценка алгоритма вычисление факториала

Factorial(n) = θ(n)

Линейная зависимость скорости работы алгоритма от объема входных данных

Анализ рекурсивных алгоритмов. Сортировка слиянием

Сортировка слиянием (merge sort) – асимптотически оптимальный алгоритм сортировки сравнением, основанный на методе декомпозиции («разделяй и властвуй», decomposition)

Требуется упорядочить заданный массив A[1..n] по неубыванию (non-decreasing order) так, чтобы

$$A[1] \le A[2] \le \cdots \le A[n]$$

Алгоритм включает два этапа

1) Разделение (partition) – рекурсивное разбиение массива на меньшие подмассивы, их сортировка
2) Слияние (merge) – объединение упорядоченных массивов в один

Сортировка слиянием

1. Разбиваем массив на две половины меньшего размера, пока размер массива больше 1

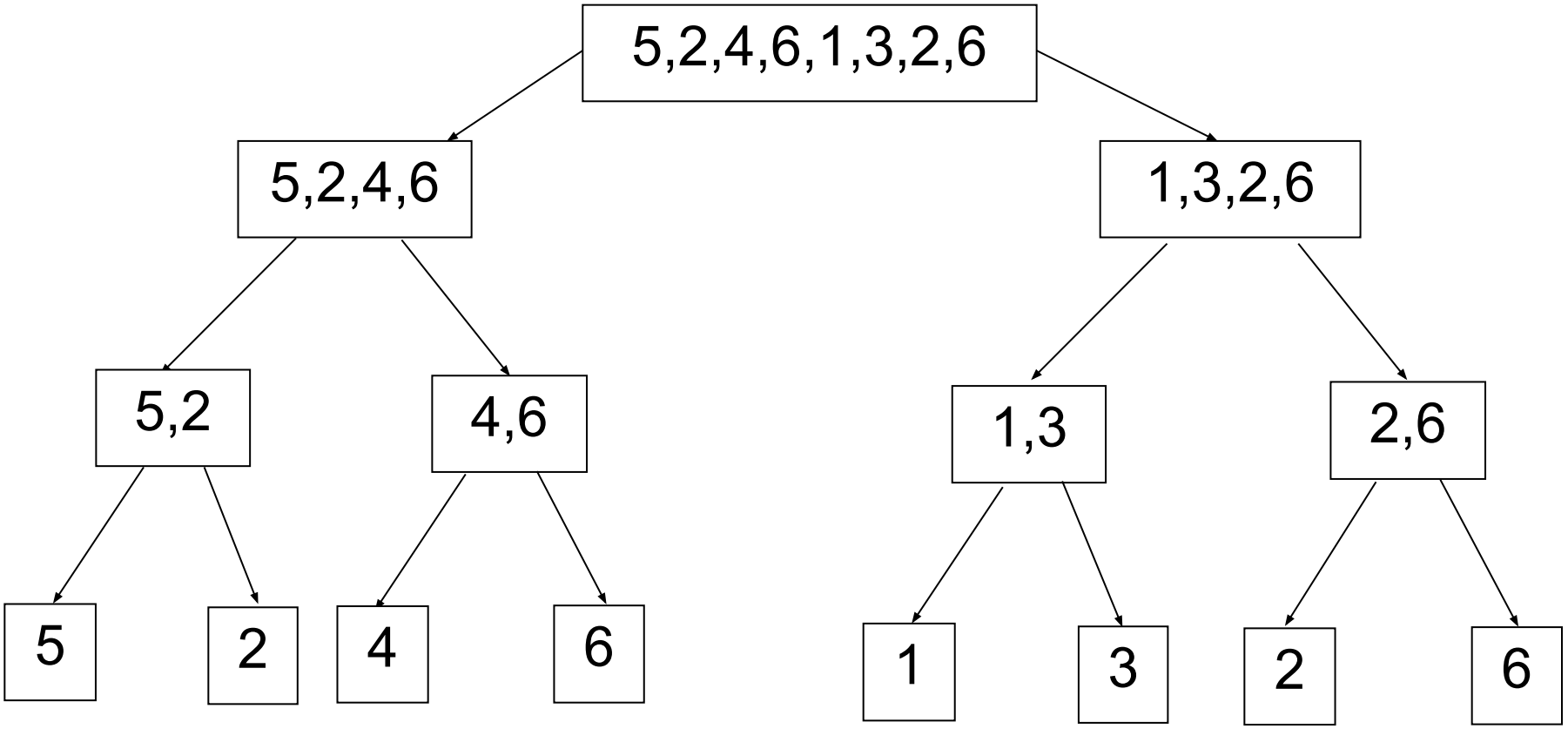
2. Рекурсивно сортируем каждую половину

3. Сливаем два отсортированных массива

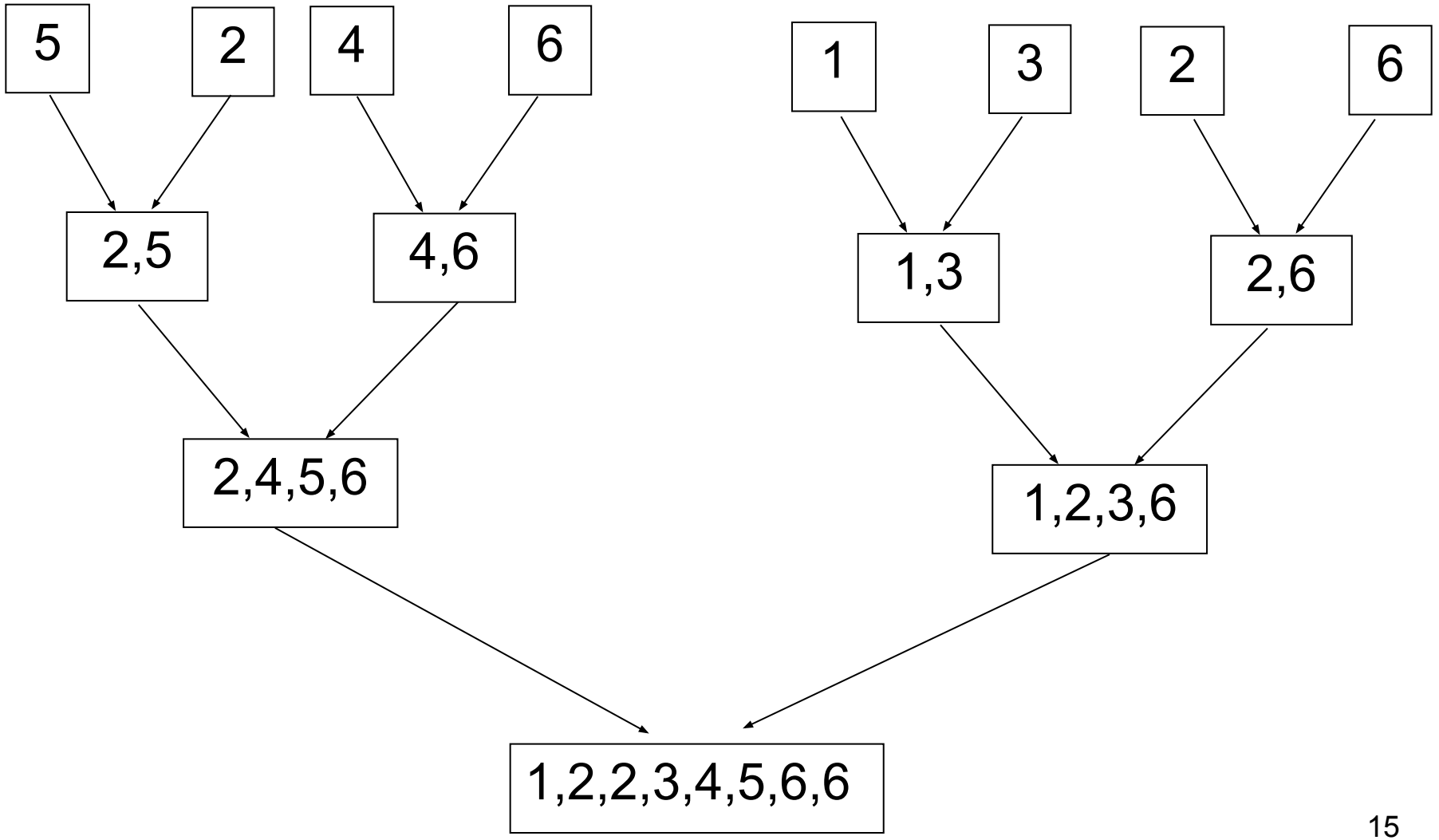
Сортировка слиянием

```
void mergeSort (int a[], int l, int r) {
    int mid;
    if (l < r) {
        mid = (l + r)/2;
        mergeSort(a, l, mid);
        mergeSort(a, mid +1, r);
        merge(a, l, mid, r);
    }
}
```

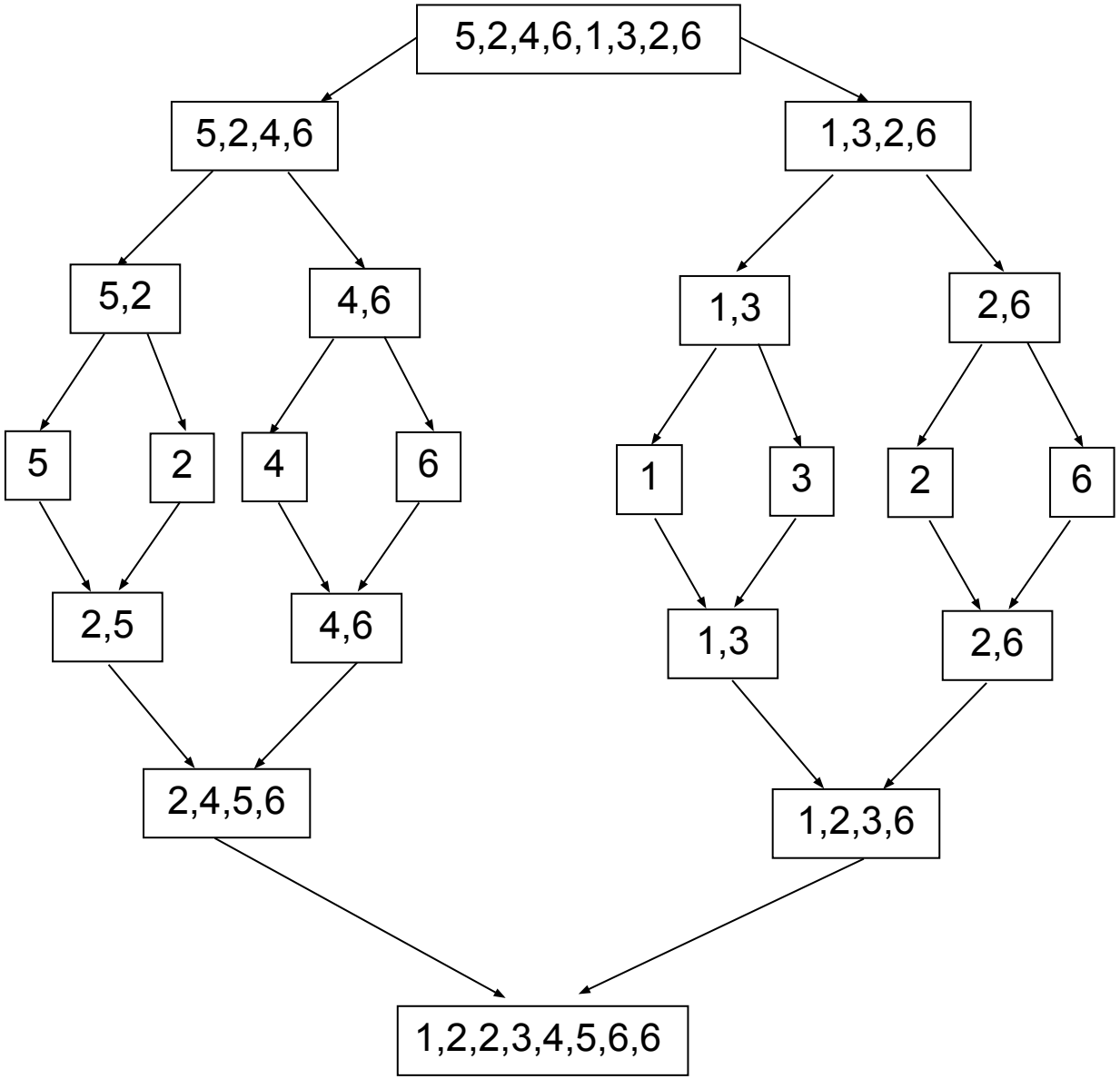
Этап разделения



Этап слияния



Сортировка слиянием



A1: 3, 4, 6, 11 A2: 2, 5, 7, 8

A: 2 2<3

A1: 3, 4, 6, 11 A2: 5, 7, 8

A: 2, 3 3<5

A1: 4, 6, 11 A2: 5, 7, 8

A: 2, 3, 4 4<5

A1: 6, 11 A2: 5, 7, 8

A: 2, 3, 4, 5 5<6

A1: 6, 11 A2: 7, 8

A: 2, 3, 4, 5, 6 6<7

A1: 11 A2: 7, 8

A: 2, 3, 4, 5, 6, 7 7<11

A1: 11 A2: 8

A: 2, 3, 4, 5, 6, 7, 8 8<11

A: 2, 3, 4, 5, 6, 7, 8, 11

Слияние упорядоченных массивов

Анализ рекурсивных алгоритмов. Сортировка слиянием

```
void mergeSort (int a[], int l, int r) {          T(n)
   int mid;                                 1
   if (l < r) {                             1
     mid = (l + r)/2;                1
     mergeSort(a, l, mid);                  T(n/2)
     mergeSort(a, mid +1, r);               T(n/2)
     merge(a, l, mid, r);                   n
   }
}
```

Рекуррентное соотношение

Рекуррентное соотношение - это соотношение содержащее функцию T(n) в левой и правой части выражения

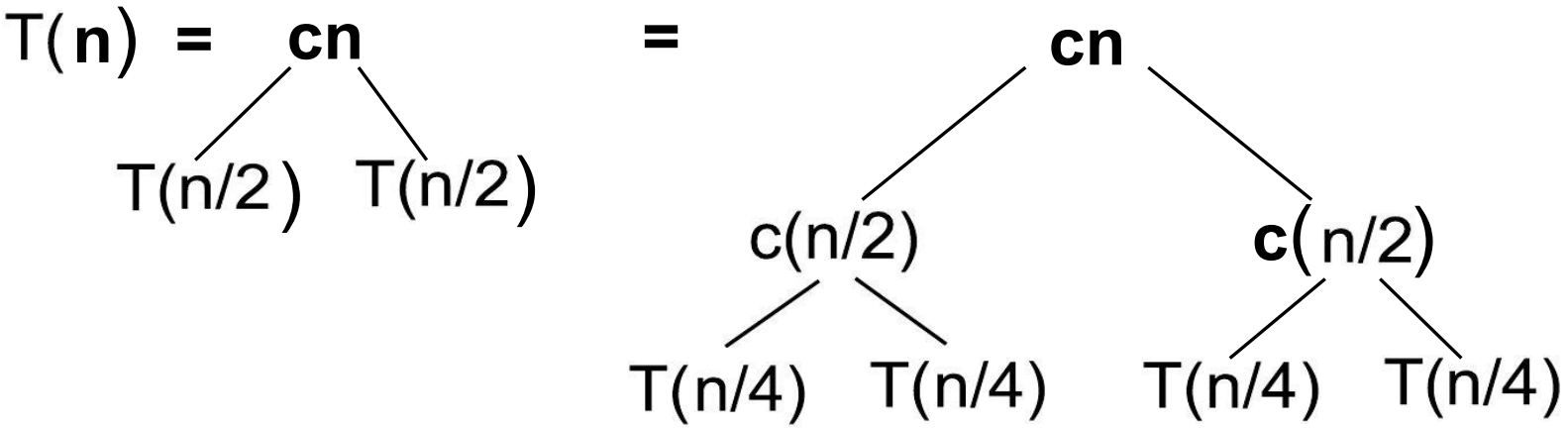
$$T(n) = \begin{cases} \Theta(1) & n = 1 \\ 2 * T(n/2) + \Theta(n) & n > 1 \end{cases}$$

Дерево рекурсии: $T(n) = 2*T(n/2) + cn$, c – const, $c>0$

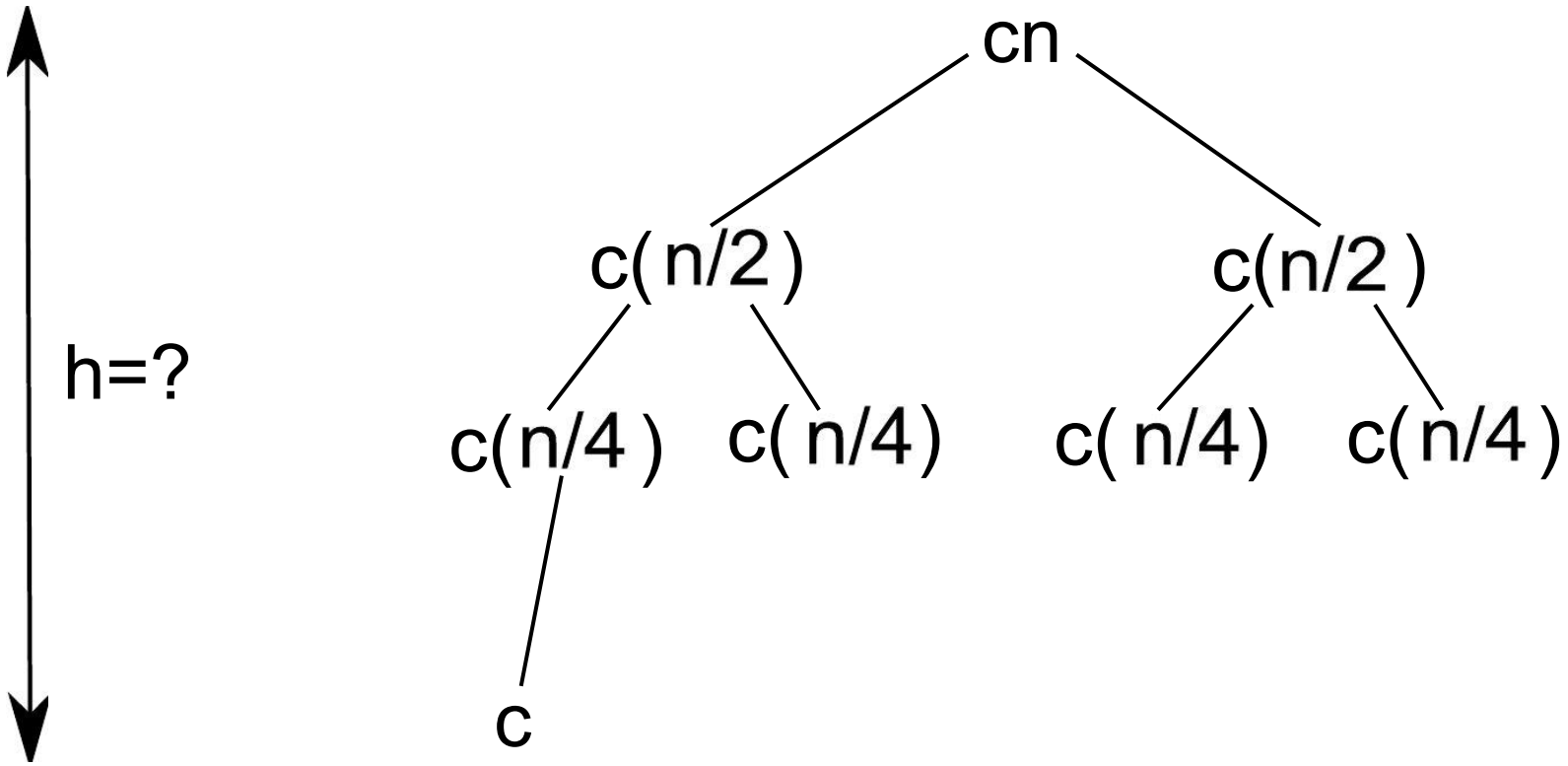
Дерево рекурсии

Дерево рекурсии для

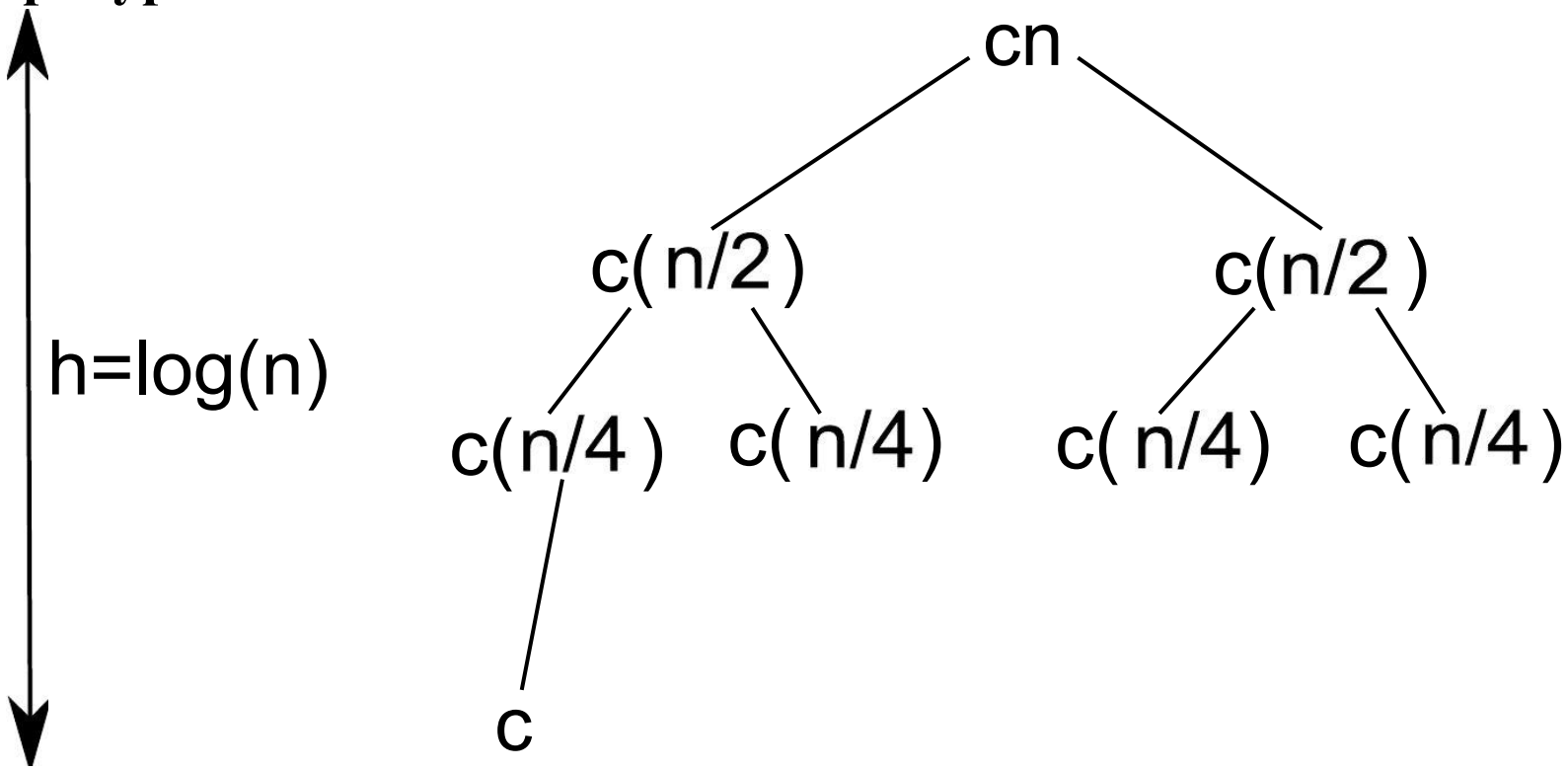
$T(n) = 2*T(n/2) + cn$



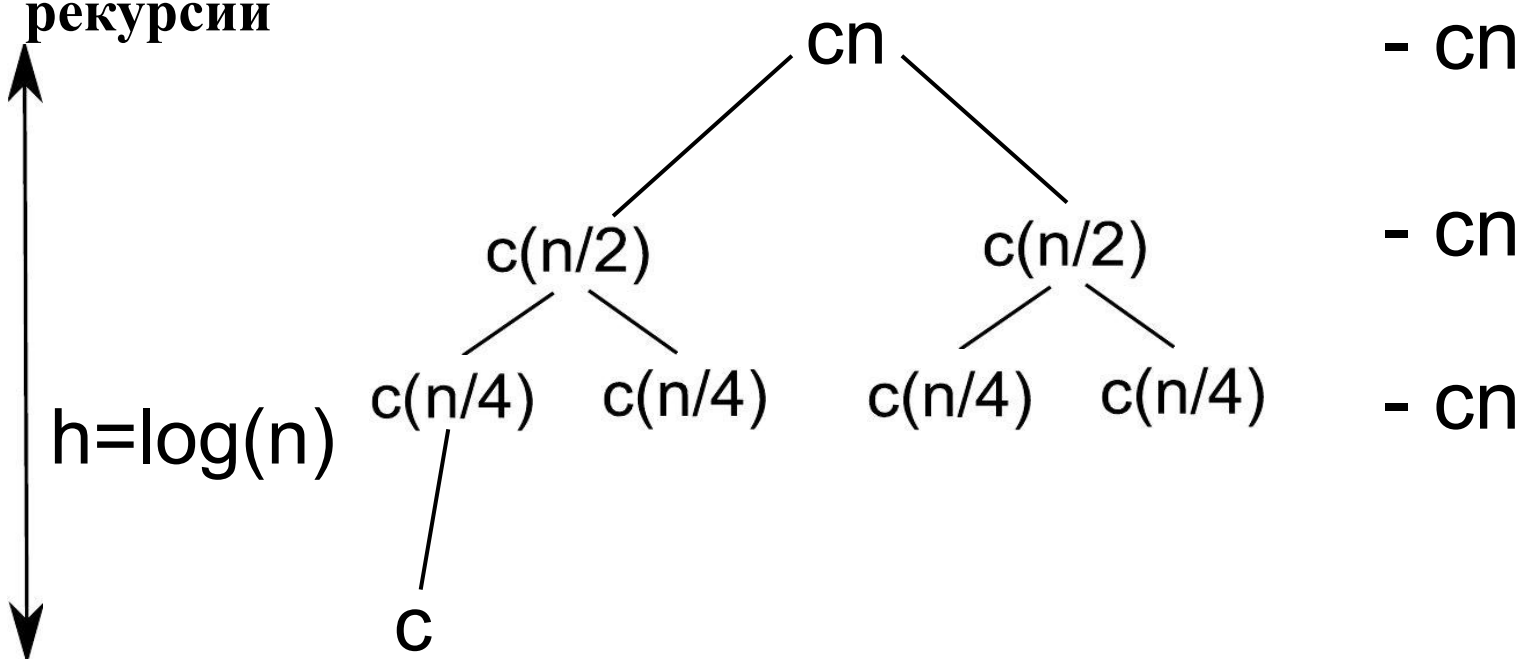
Дерево рекурсии



Дерево рекурсии



Дерево рекурсии



$$T(n) = \Theta(n \cdot \log_2(n))$$

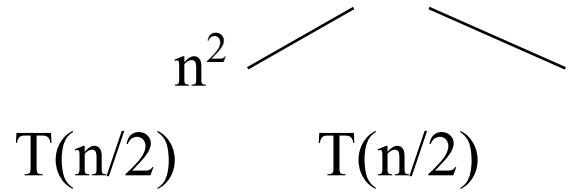
Сравнение алгоритмов сортировки

- Сортировка вставками – $\Theta(n^2)$
- Сортировка слиянием - $\Theta(n * log_2(n))$

Дерево рекурсии

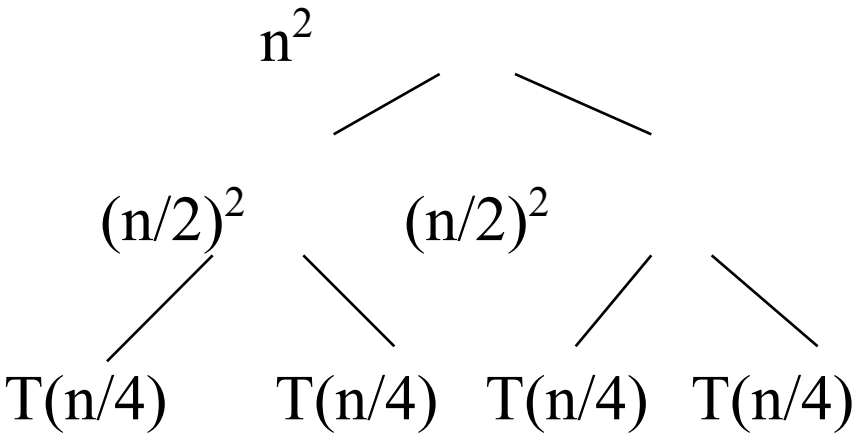
Дерево рекурсии - графический метод, который отображает разворачивание рекуррентного соотношения в систему рекурсивных вызовов

$T(n)=2T(n/2)+n^2$



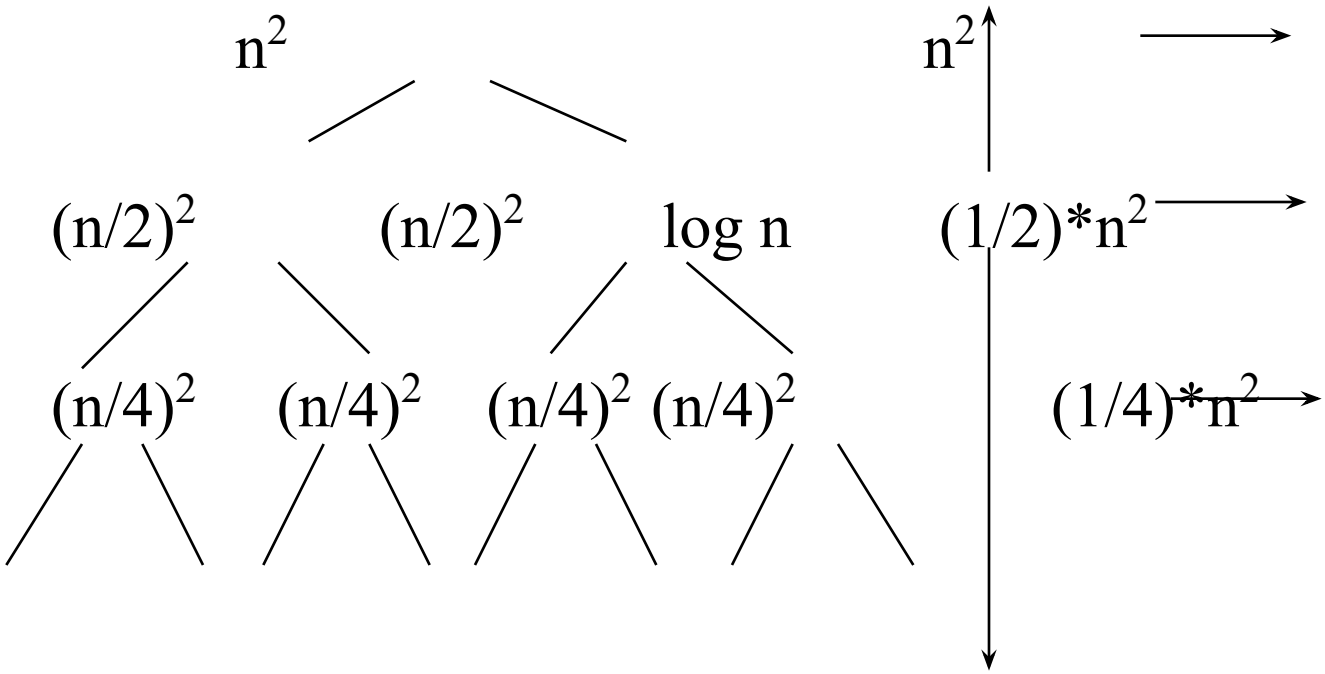
Дерево рекурсии

$T(n)=2T(n/2)+n^2$



Дерево рекурсии

$T(n)=2T(n/2)+n^2$



$T(n)=\Theta(n^2)$

Подсчет оценки

Сумма: по уровням

$(1 + 1/2 + 1/4 + \ldots + 1/2^k + \ldots)n^2 = ?$

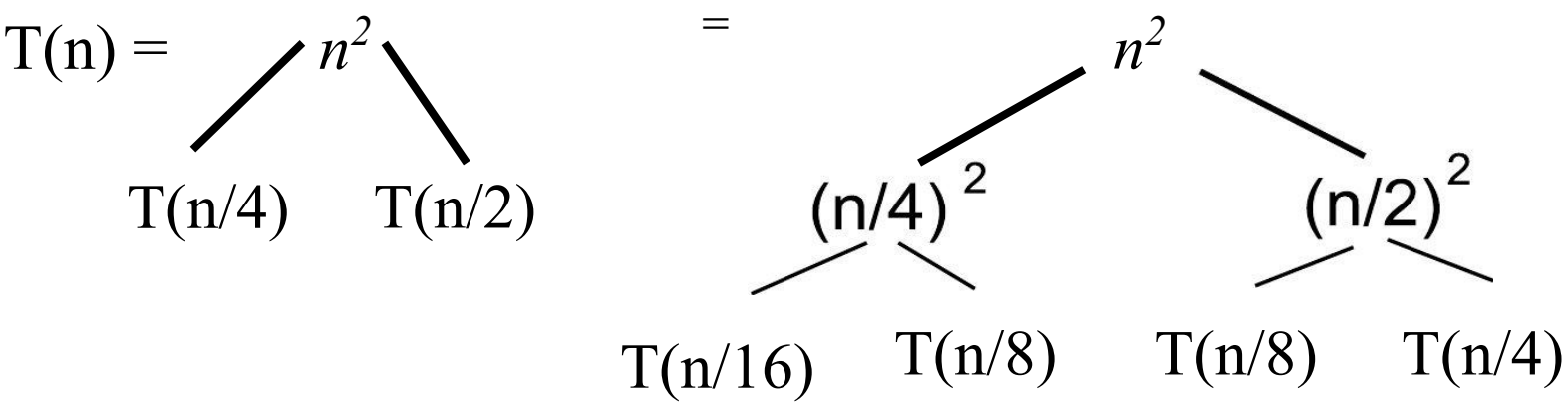
$$S = \frac{1}{1 - \frac{1}{2}} = 2$$

$b_1 = 1$, $q = 1/2$

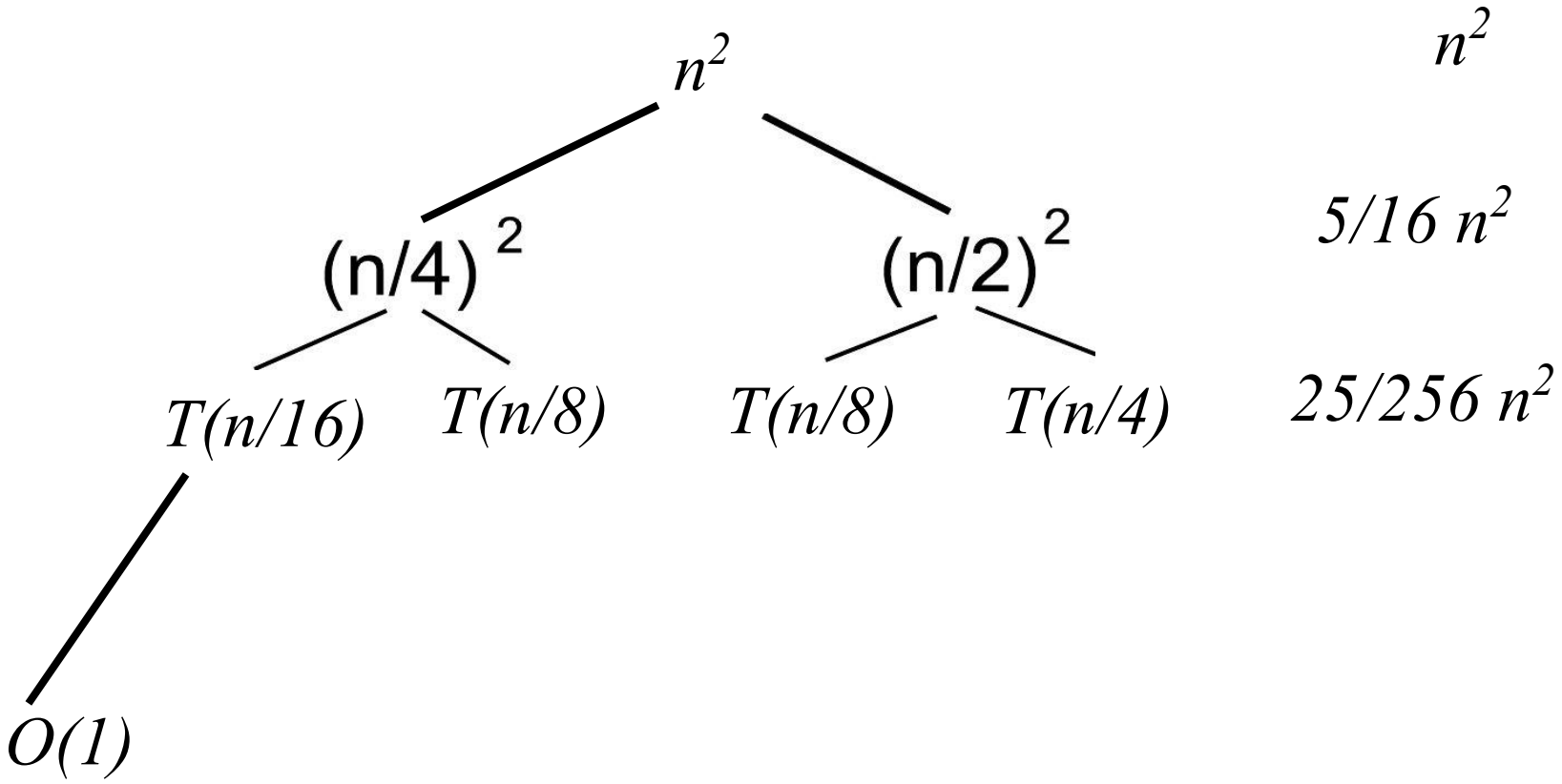
$T(n) = \theta(n^2)$

Дерево рекурсии

$$T(n) = T(n/4) + T(n/2) + n^2$$



Дерево рекурсии



Подсчет оценки

Сумма: по уровням

$(1 + 5/16 + 25/256 + \ldots + 5^k/16^k + \ldots)n^2 = ?$

$$S = \frac{1}{1 - \frac{5}{16}} \approx 1.5$$ $b_1=1$, $q= 5/16$

$T(n)=\theta(n^2)$

Основная теорема о рекуррентных оценках

T (n) = aT (n/b) + f (n), *a ≥ 1, b > 1, f (n) − (f (n) > 0, n > n0)*

1. $f(n) = O(n^{\log_b a - \varepsilon}), \varepsilon > 0 \Rightarrow T(n) = \Theta(n^{\log_b a})$

2. $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} * \log n)$

3. $f(n) = \Omega(n^{\log_b a + \varepsilon}), \varepsilon > 0, a * f(n/b) \le c * f(n), c < 1, n \to \infty \Rightarrow$
$T(n) = \Theta(f(n))$

Подсчет оценки

1.

$T(n) = 4T(n/2) + n$

$a=4,\ b=2,\ f(n)=n$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

При $\varepsilon=1$ выполняется условие для первого случая теоремы

$T(n)=\theta(n^2)$

Подсчет оценки

2.

$T(n) = 4T(n/2) + n^2$

$a=4,\ b=2,\ f(n)=n^2$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$

$$f(n) = n^{\log_b a}$$

второй случай теоремы

$T(n)=\theta(n^2 * \log n)$

Подсчет оценки

3.

$T(n) = 3T(n/4) + n*\log n$

$a=3,\ b=4,\ f(n)= n*\log n$

$$n^{\log_b a} = n^{\log_4 3} = O(n^{0,79})$$

$$f(n) > n^{\log_b a}$$

$$\varepsilon \approx 0.2$$

третий случай теоремы

условие регулярности

$a*f(n/b)=3*(n/4)*\log(n/4) \le (3/4)*n*\log n = c*f(n),\ c=3/4$

$T(n)=\theta(n*\log n)$