Основы программного конструирования

ЛЕКЦИЯ №2

13 ФЕВРАЛЯ 2023

Временная сложность

- Как считать сложность?

Временная сложность

- Как считать сложность?
- Количество требуемых операций (время): Q(n)=O(f(n)), где n – размер структуры данных.

f(n)=O(g(n)) :

- Как считать сложность?
- Количество требуемых операций (время): Q(n)=O(f(n)), где n – размер структуры данных.

f(n)=O(g(n)) $\leftrightarrow \exists\, n_0, M: \forall n \geq n_0:$

$|f(n)| \leq M \cdot |g(n)|$

Временная сложность

- Как считать сложность?
- Количество требуемых операций (время): Q(n)=O(f(n)), где n – размер структуры данных.

f(n)=O(g(n)) :

- O(1) – константная.

- Как считать сложность?
- Количество требуемых операций (время): Q(n)=O(f(n)), где n – размер структуры данных.

f(n)=O(g(n)) $\leftrightarrow \exists\, n_0, M: \forall n \geq n_0:$
$|f(n)| \leq M \cdot |g(n)|$

- O(1) – константная.
- O(log(n)) – логарифмическая.
- O(n) – линейная.
- O(n*log(n)) – квази-линейная.
- $O(n^2)$ – квадратичная.
- $O(2^n)$ – экспоненциальная.

Временная сложность

- Как считать сложность?
- Количество требуемых операций (время): Q(n)=O(f(n)), где n – размер структуры данных.

f(n)=O(g(n)) :

- O(1) – константная.
- O(log(n)) – логарифмическая.
- O(n) – линейная.
- O(n*log(n)) – квази-линейная.
- O) – квадратичная.
- O() – экспоненциальная.

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10						
100						
10000						

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A					
100	A					
10000	A					

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A	B				
100	A	2B				
10000	A	4B				

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A	B	C			
100	A	2B	10C			
10000	A	4B	1000C			

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A	B	C	D		
100	A	2B	10C	20D		
10000	A	4B	1000C	4000D		

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A	B	C	D	E	
100	A	2B	10C	20D	100E	
10000	A	4B	1000C	4000D		

Масштабируемость

n	O(1)	O(log(n))	O(n)	O(n*log(n))	O)	O()
10	A	B	C	D	E	F
100	A	2B	10C	20D	100E	1000F
10000	A	4B	1000C	4000D		

О скорости и тормозах

Для n=10 процесс выполняется 1 сек.

Зависимость	Время для n=1000
O(1)	1 сек

О скорости и тормозах

Для n=10 процесс выполняется 1 сек.

Зависимость	Время для n=1000
O(1)	1 сек
O(log(n))	3 сек

О скорости и тормозах

Для n=10 процесс выполняется 1 сек.

Зависимость	Время для n=1000
O(1)	1 сек
O(log(n))	3 сек
O(n)	2 мин

О скорости и тормозах

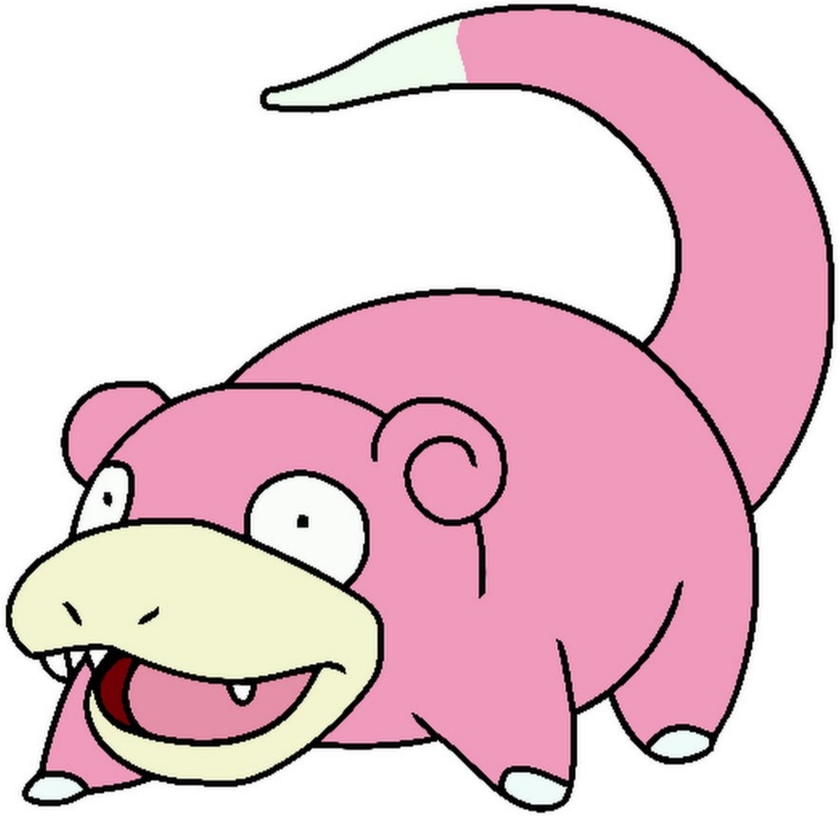
Для n=10 процесс выполняется 1 сек.

Зависимость	Время для n=1000
O(1)	1 сек
O(log(n))	3 сек
O(n)	2 мин
O(n*log(n))	5 мин

О скорости и тормозах

Для n=10 процесс выполняется 1 сек.

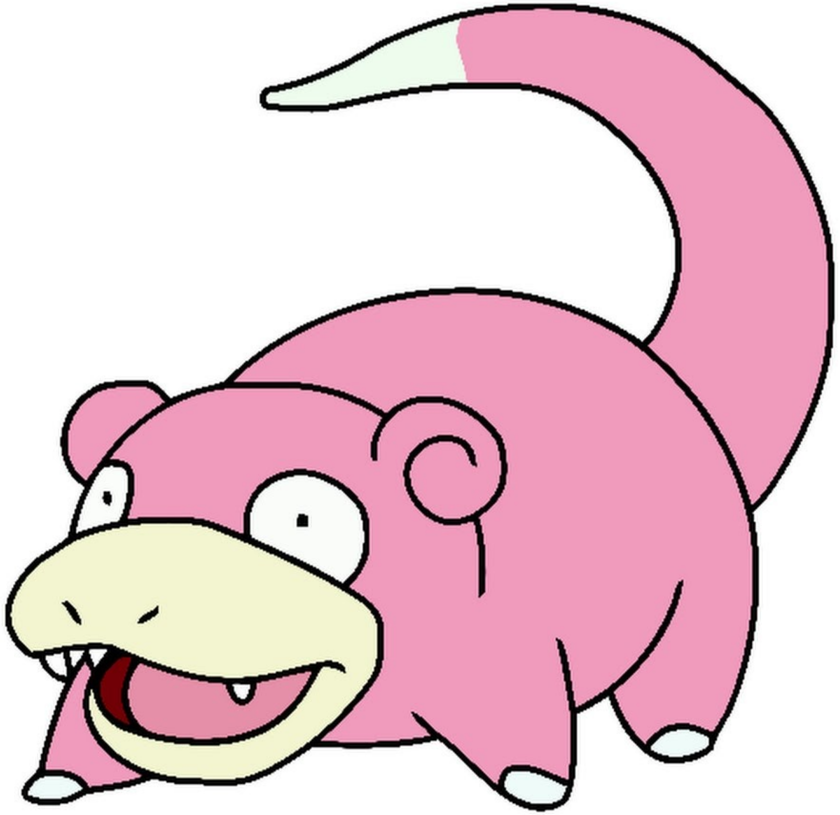
Зависимость	Время для n=1000
O(1)	1 сек
O(log(n))	3 сек
O(n)	2 мин
O(n*log(n))	5 мин
O)	3 часа



О скорости и тормозах

Для n=10 процесс выполняется 1 сек.

Зависимость	Время для n=1000
O(1)	1 сек
O(log(n))	3 сек
O(n)	2 мин
O(n*log(n))	5 мин
O)	3 часа
O()	12 суток



О скорости и тормозах

Экспоненциальная сложность

Q(n)=O()

- Для n=10 процесс выполняется:

1 сек.

О скорости и тормозах

Экспоненциальная сложность

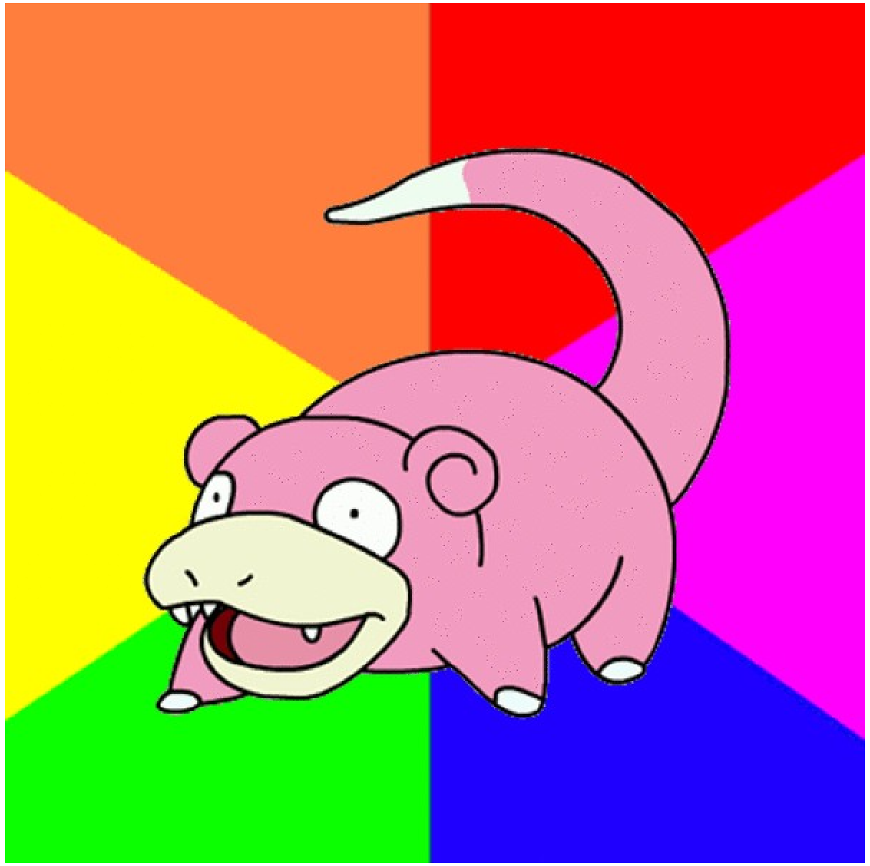
Q(n)=O()

- Для n=10 процесс выполняется:

1 сек.

- Для n=100 процесс выполняется:

лет.



- Для n=10 процесс выполняется:

1 сек.

- Для n=100 процесс выполняется:

10^{27} сек. $\approx 4 * 10^{19}$ лет.

- Для n=1000 процесс выполняется:

10^{298} сек. $\approx 3 * 10^{290}$ лет.

О скорости и тормозах

Экспоненциальная сложность

Q(n)=O()

- Для n=10 процесс выполняется:

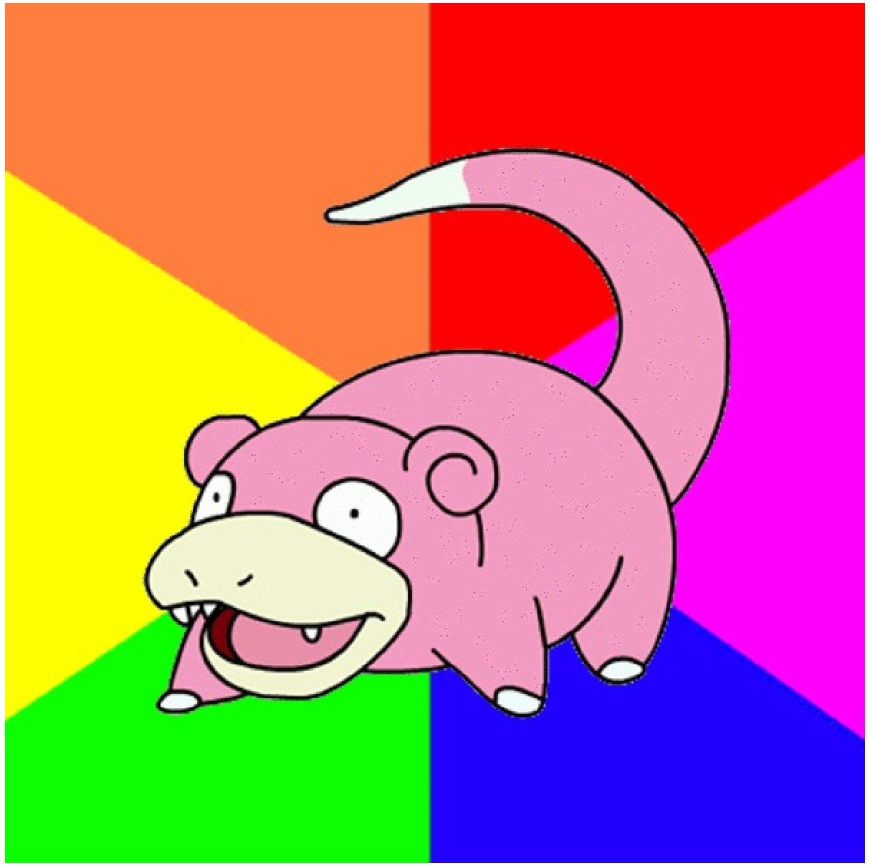
1 сек.

- Для n=100 процесс выполняется:

лет.

- Для n=1000 процесс выполняется:

лет.



Примеры

- Вычисление суммы двух чисел – O(1) (хотя на самом деле зависит от размера чисел).

- Подсчет длины строки – O(n) (один цикл).

- Умножение двух матриц – O() (три вложенных цикла).

- Полный перебор всех n-битных чисел – O()

O, ,

f(n)=O(g(n)) :

f(n)=(g(n)) :

f(n)=(g(n)) :

Случаи

Сложность в …

- в лучшем случае (best-case)
- в среднем случае (average-case)
- в худшем случае (worst-case) (вариант по умолчанию)

- best-case average-case worst-case

Задача сортировки

- Записи
- Ключи
- Требуется расставить записи так, чтобы ключи шли в неубывающем порядке.

Идиотская сортировка (BOGOSORT)

1) Проверить, является ли массив уже отсортированным. Если да, то алгоритм завершается.
2) Случайным образом перемешать записи в массиве и перейти к шагу 1.

В среднем: O(n*n!)

Тупая сортировка (BOZOSORT)

1) Проверить, является ли массив уже отсортированным. Если да, то алгоритм завершается.
2) Случайным образом поменять местами две записи в массиве и перейти к шагу 1.

В среднем: O(n!)

Сортировка выбором (Selection Sort)

1) На **j**-ой итерации первые (**j-1**) элементов уже на своих местах.
2) Найти наименьший ключ из .
3) Поменять местами и (тогда первые **j** элементов на своих местах).
4) Если **j** равен **n**, то массив отсортирован, иначе переходим на шаг 1.

Сортировка выбором (Selection Sort)

1) На **j**-ой итерации первые (**j-1**) элементов уже на своих местах.
2) Найти наименьший ключ из .
3) Поменять местами и (тогда первые **j** элементов на своих местах).
4) Если **j** равен **n**, то массив отсортирован, иначе переходим на шаг 1.

O()

Сортировка выбором (Selection Sort)

https://www.toptal.com/developers/sorting-algorithms

Пузырьковая сортировка(Bubble Sort)

$O(n^2)$

1) Пройти массив от начала к концу, сравнивая на каждом шаге пару соседних ключей и . Если , то записи и меняются местами.
2) Если на предыдущем шаге была хотя бы одна перестановка, перейти к шагу 1. (в противном случае – массив отсортирован).

Бонус: Чередовать проходы в прямую и обратную стороны (cocktail sort).

O()

Пузырьковая сортировка(Bubble Sort)

Сортировка вставками (Insertion Sort)

1) На j-ой итерации часть записей уже отсортирована:
   … , .
2) Меняем с элементами слева от него, пока не встретим элемент меньший . После этого последовательность отсортирована.
3) Если **j** равен **n**, то массив отсортирован, иначе переходим на шаг 1

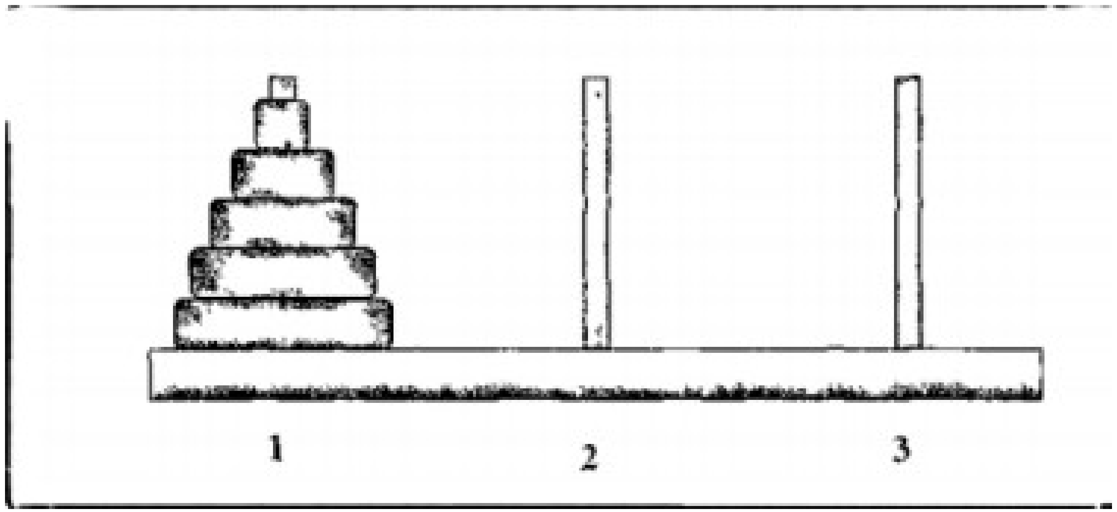
O()

Сортировка вставками (Insertion Sort)

«Разделяй и властвуй»

- **Разделение** задачи на несколько подзадач.
- **Покорение:** решение подзадач.
- **Комбинирование** решения исходной задачи из решений подзадач.

Задача о ханойской башне

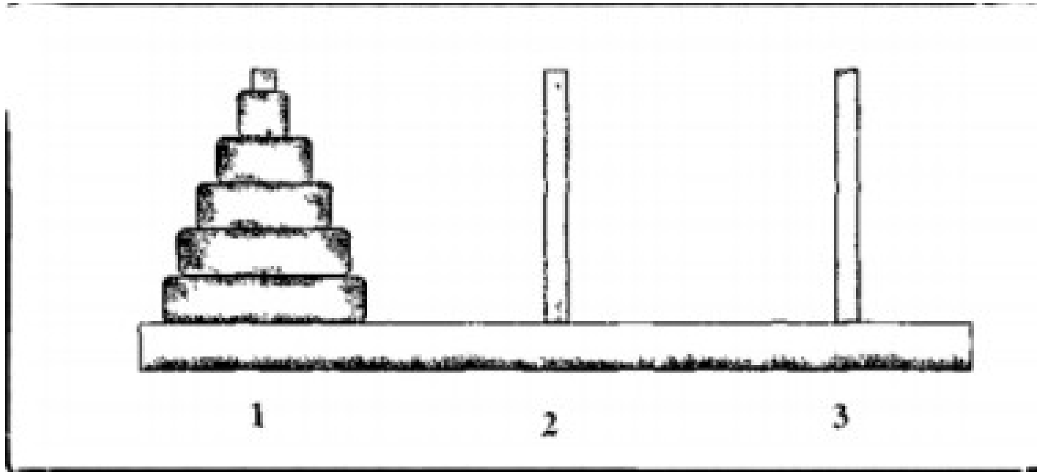


- Задача: перенести **n** дисков с палки **1** на палку **3**, используя палку **2** как свободную.
- Можно класть только меньший диск на больший!

Решение

Решение:

1. Перенести **n-1** дисков с **1** на **2**, пользуясь свободной **3**.
2. Перенести самый большой (**n**-й по счету) диск с **1** на **3**.
3. Перенести **n-1** дисков с **2** на **3**, пользуясь свободной **1**.



Получение всех перестановок

Как получить все перестановки чисел **1…n**?

Получение всех перестановок

Как получить все перестановки чисел **1…n**?

1. Как-то получаем все перестановки **1…n-1**.
2. Есть **n** способов добавить число **n** к каждой из перестановок, полученных на шаге **1**.

Алгоритм Карацубы для умножения длинных чисел

Есть два длинных числа: и. Нужно получить произведение.

Алгоритм Карацубы для умножения длинных чисел

Есть два длинных числа: и. Нужно получить произведение.

Алгоритм Карацубы для умножения длинных чисел

Есть два длинных числа: и. Нужно получить произведение.

, где

, - аналогично.

Есть два длинных числа: $a_{2n}a_{2n-1}\dots a_1$ и $b_{2n}b_{2n-1}\dots b_1$. Нужно получить произведение.

$$C = \left(10^{2n}a_{2n} + 10^{2n-1}a_{2n-1} + \dots + a_1\right) * \left(10^{2n}b_{2n} + 10^{2n-1}b_{2n-1} + \dots + b_1\right)$$

$C = (10^n A_1 + A_0) * (10^n B_1 + B_0)$, где

$$A_1 = 10^n a_{2n} + 10^{n-1} a_{2n-1} + \dots + a_{n+1}$$

$$A_0 = 10^n a_n + 10^{n-1} a_{n-1} + \dots + a_1$$

B_0, B_1 - аналогично.

Алгоритм Карацубы для умножения длинных чисел

Умножение чисел размерности 2n сведено к четырем умножениям n-размерных чисел и комбинированию посредством сдвигов и сложений.

Алгоритм Карацубы для умножения длинных чисел

Умножение чисел размерности 2n сведено к четырем умножениям n-размерных чисел и комбинированию посредством сдвигов и сложений.

Хитрость! Вычисляем 3 умножения:

Тогда

Умножение чисел размерности 2n сведено к четырем умножениям n-размерных чисел и комбинированию посредством сдвигов и сложений.

Хитрость! Вычисляем 3 умножения:

$D_1 = A_0 B_0$

$D_2 = A_1 B_1$

$D_3 = (A_0 + A_1)(B_0 + B_1)$

Тогда $C = (10^{2n} D_2 + 10^{n} (D_3 - D_2 - D_1) + D_1)$

Основная теорема о рекуррентных соотношениях (The master method)

Количество подзадач

Сложность разделения и объединения

Пусть **1, b 2, d 0**

Размер каждой подзадачи

Основная теорема о рекуррентных соотношениях (The master method)

Количество подзадач

Сложность разделения и объединения

Пусть **1, b 2, d 0**

Размер каждой подзадачи

, если

, если

, если

Примеры рекуррентных соотношений

- Прямолинейное умножение длинных чисел:

- Алгоритм Карацубы:

Линейный поиск

Дан массив **A** длины **n** и ключ **K**. Требуется определить положение элемента с данным ключом в массиве или установить, что его там нет.

Последовательно сравниваем ключи, пока не найдем совпадающий ключ или массив не кончится.

Сложность: **O(n)**.

Двоичный поиск

Дан **отсортированный** массив **A** длины **n** и ключ **K**. Двоичный поиск (он же бинарный поиск, он же поиск делением пополам):

Возьмем серединный элемент массива (**m**-й) и сравним его ключ с :

- Если , то ключ найден.
- Если , то продолжаем поиск в правой половине **A[m+1…n-1]**
- Если , то продолжаем поиск в левой половине **A[0…m1-]**

Двоичный поиск

Дан **отсортированный** массив **A** длины **n** и ключ **K**. Двоичный поиск (он же бинарный поиск, он же поиск делением пополам):

Возьмем серединный элемент массива (**m**-й) и сравним его ключ c :

- Если , то ключ найден.
- Если , то продолжаем поиск в правой половине **A[m+1…n-1]**
- Если , то продолжаем поиск в левой половине **A[0…m1-]**
- Сложность: **T(n) = T(n/2) + O(1) = O(log(n))**.

Рекурсия

- Рекурсия – см. рекурсия.

Рекурсия

- Рекурсия – см. рекурсия.
- Рекурсия – вызов функцией самой себя.

Рекурсия

- Рекурсия – см. рекурсия.
- Рекурсия – вызов функцией самой себя.
- Есть опасность бесконечной рекурсии.
- В большинстве случаев расходуется машинный стек.
- Взаимодействие только через аргументы и возвращаемое значение.
- Реализация может быть итеративной (своя реализация стека или оптимизация хвостовой рекурсии).

Другие применения?

Удачно ли?

На сегодня все