Алгоритмы и структуры данных

Асимптотический анализ. Разделяй-и-властвуй DaC



Практическое занятие **05**

2026–2027 учебный год

План

1. Большой **revision**:
   - метод математической индукции
     для решения рекуррентных соотношений
   - построение дерева рекурсии — поиск лучшей оценки
2. Применение мастер-теорем и метода Акра-Баззи
3. Практический временной сложности
   рекурсивных алгоритмов и перекрестная рекурсия

Warm-Up

FAQ Разделяй-и-властвуй, асимптотика

1. На оптимизацию числа каких операций направлены «разделяй-и-властвуй» алгоритмы для умножения целых чисел и матриц?
2. В чем состоит задача интерполяции многочлена $P(x)$ степени $n - 1$?
3. Для рекуррентных соотношений какого вида применимы мастер-теоремы и что они дают?

Методы решения рекуррентных соотношений

конструктивный подход

Задача 1



Метод математической индукции

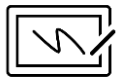
С помощью метода математической индукции подтвердите или опровергните следующие предположения об оценке сложности алгоритмов, которые характеризуются следующими рекуррентными соотношениями:

1. $T(n) = T(n-1) + 2 = \Theta(n)$.
2. $T(n) = 2T(\lfloor n/2 \rfloor + 17) + 3n = O(n \log n)$.

Не забывайте, что некоторые операции [в частности, округления и возмущения аргумента] в рекурсивной части соотношения **не ведут** к изменению асимптотической оценки.

Задача 2





Дерево рекурсии

С помощью дерева рекурсии найдите **как можно более точную** асимптотическую оценку временной сложности алгоритмов, время работы которых определяется следующими рекуррентными соотношениями:

1. $T(n) = 5T(n/4) + \log n.$
2. $T(n) = T(n-a) + T(a) + cn$, где $a = \text{const} \geq 0$ и $c > 0.$

Методы решения рекуррентных соотношений

быстрый подход

Задача 3



Мастер-теорема разделяй-и-властвуй

$$T(n) = \begin{cases} a \cdot T\left(n/b\right) + O\left(n^k \cdot f(n)\right), n > 1 \\ O(1), n = 1 \end{cases}$$

разделяй — властвуй

$k = \log_b a \Rightarrow T(n) = O\left(n^k \cdot f(n) \cdot \log n\right)$
$k < \log_b a \Rightarrow T(n) = O\left(n^{\log_b a} \cdot f(n)\right)$
$k > \log_b a \Rightarrow T(n) = O\left(n^k \cdot f(n)\right)$

Задача 3

Мастер-теорема разделяй-и-властвуй

$T(n) = \begin{cases} a \cdot T\left(n/b\right) + O\left(n^k \cdot f(n)\right), n > 1 \\ O(1), n = 1 \end{cases}$ (разделяй, властвуй)	$k = \log_b a \Rightarrow T(n) = O\left(n^k \cdot f(n) \cdot \log n\right)$
	$k < \log_b a \Rightarrow T(n) = O\left(n^{\log_b a} \cdot f(n)\right)$
	$k > \log_b a \Rightarrow T(n) = O\left(n^k \cdot f(n)\right)$

- $T(n) = 16\,T(n/4) + n$
- $T(n) = 4\,T(n/2) + n^2$
- $T(n) = 2^n\,T(n/2) + n^n$
- $T(n) = 2\,T(n/2) + n \log n$
- $T(n) = 2\,T(n/2) + n/\log n$
- $T(n) = 2\,T(n/2) + 8/9\;T(3n/4) + n$

1. Оцените верхнюю границу временной сложности представленных рекуррентных соотношений.
2. Если применение мастер-теоремы невозможно, поясните почему.

Задача 4

Мастер-теорема уменьшай-и-властвуй

$$T(n) = \begin{cases} a \cdot T(n-b) + O\left(n^k \cdot f(n)\right), n > 0 \\ O(1), n = 0 \end{cases}$$

уменьшай — властвуй

$a = 1 \Rightarrow T(n) = O\left(n^{k+1} \cdot f(n)\right)$
$a < 1 \Rightarrow T(n) = O\left(n^k \cdot f(n)\right)$
$a > 1 \Rightarrow T(n) = O\left(a^{n/b} \cdot n^k \cdot f(n)\right)$

Задача 4

Мастер-теорема уменьшай-и-властвуй

$$T(n) = \begin{cases} a \cdot T(n-b) + O\left(n^k \cdot f(n)\right), n > 0 \\ O(1), n = 0 \end{cases}$$

уменьшай — властвуй

$a = 1 \Rightarrow T(n) = O\left(n^{k+1} \cdot f(n)\right)$
$a < 1 \Rightarrow T(n) = O\left(n^k \cdot f(n)\right)$
$a > 1 \Rightarrow T(n) = O\left(a^{n/b} \cdot n^k \cdot f(n)\right)$

- $T(n) = 2\,T(n-1) + n$
- $T(n) = T(n-3) + \log n$
- $T(n) = 3\,T(n-2) + \sqrt{n}$
- $T(n) = T(n-1) + T(n-2) + 2n$

1. Оцените верхнюю границу временной сложности представленных рекуррентных соотношений.
2. Если применение мастер-теоремы невозможно, поясните почему.

Задача 5

Это ограубление!

Для временной сложности $T(n)$ алгоритма справедливо, что $T(m) \leq T(n)$ для любого m. Используя этот факт совместно с подходящей мастер-теоремой оцените верхнюю и нижнюю границу временной сложности:

1. $T(n) = 2\,T(n/2) + 8/9\;T(3n/4) + n$
2. $T(n) = T(n-1) + T(n-2) + 2n$

Задача 5





Это огр~~а~~убление!

Для временной сложности $T(n)$ алгоритма справедливо, что $T(m) \leq T(n)$ для любого m. Используя этот факт совместно с подходящей мастер-теоремой оцените верхнюю и нижнюю границу временной сложности:

1. $T(n) = 2\,T(n/2) + 8/9\;T(3n/4) + n$
2. $T(n) = T(n-1) + T(n-2) + 2n$

[Q] Каков основной недостаток такого подхода?

Задача 6

Метод Акра-Баззи

$$T(n) = \begin{cases} \sum_{i=1}^{k} a_i \cdot T(b_i \cdot n) + g(n), n > n_0 \\ \Theta(1), 1 \le n \le n_o \end{cases}$$

$$T(n) = \Theta\left(n^p \cdot \left(1 + \int_1^n \frac{g(u)}{u^{p+1}} du\right)\right)$$

$$a_1 \cdot b_1^p + a_2 \cdot b_2^p + \cdots + a_k \cdot b_k^p = 1$$

Задача 6

Метод Акра-Баззи

$$T(n) = \begin{cases} \sum_{i=1}^{k} a_i \cdot T(b_i \cdot n) + g(n), n > n_0 \\ \Theta(1), 1 \le n \le n_o \end{cases}$$

$$T(n) = \Theta\left(n^p \cdot \left(1 + \int_1^n \frac{g(u)}{u^{p+1}} du\right)\right)$$

$$a_1 \cdot b_1^p + a_2 \cdot b_2^p + \cdots + a_k \cdot b_k^p = 1$$

Определите точную асимптотическую оценку $\Theta(g(n))$ временной сложности $T(n)$ с помощью обобщенного метода:

$$T(n) = 2\,T(n/2) + 8/9\;T(3n/4) + n.$$

Интеграл можно вычислять любым доступным способом.

[Q] Отличается ли точная оценка от найденных ранее?

Перекрестная рекурсия
mutual recursion

Вспомним

Типы рекурсивных алгоритмов

TAIL RECURSION.cpp

```cpp
void func(int n) {
    if (n > 0) std::cout << n << " ";

    func(n-1);
}

int main() {
    int x = 5;
    func(x);
    return 0;
}
```

ХВОСТОВАЯ РЕКУРСИЯ

Рекурсивный вызов происходит после выполнения всех вычислений в теле рекурсивной функции.

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

Хвостовая рекурсия проще всего переводится в обычный цикл.

Вспомним

Типы рекурсивных алгоритмов

HEAD RECURSION.cpp

```cpp
void func(int n) {
    if (n > 0) {
        func(n-1);
        std::cout << n << " ";
    }
}

int main() {
    int x = 5;
    func(x);
    return 0;
}
```

ГОЛОВНАЯ РЕКУРСИЯ

Рекурсивный вызов происходит до выполнения всех вычислений в теле рекурсивной функции.

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

Головная рекурсия тоже переводится в обычный цикл, но это уже сложнее.

Вспомним

Типы рекурсивных алгоритмов

TREE RECURSION.cpp

```cpp
void func(int n) {
    if (n > 0) {
        std::cout << n << " ";
        func(n - 1);
        func(n - 1);
    }
}

int main() {
    int x = 3;
    func(x);
    return 0;
}
```

ДРЕВЕСНАЯ РЕКУРСИЯ

В теле функции выполняется несколько рекурсивных вызовов.

$$T(n) = 2\,T(n-1) + f(n)$$

Размеры рекурсивных задач в древесной рекурсии не обязаны быть одинаковыми.

Вспомним

Типы рекурсивных алгоритмов

NESTED RECURSION.cpp

```cpp
int func(int n) {
    if (n > 100)  return n - 10;

    func(func(n + 11));
}

int main() {
    int x = 95;
    std::cout << func(x);
    return 0;
}
```

Функция Маккарти

ВЛОЖЕННАЯ РЕКУРСИЯ

В теле функции выполняются вложенные рекурсивные вызовы.

$$T(n) = \begin{cases} n - 10, n > 100 \\ T\big(T(n+1)\big), \text{иначе} \end{cases}$$

- Решение вложенного соотношения не укладывается в стандартные методы.
- Можно применить метод **мат. индукции** для вывода в явном [closed] виде.

Вспомним

Типы рекурсивных алгоритмов

NESTED RECURSION.cpp

```cpp
int func(int n) {
    if (n > 100)  return n - 10;

    func(func(n + 11));
}

int main() {
    int x = 95;
    std::cout << func(x);
    return 0;
}
```

Функция Маккарти

ВЛОЖЕННАЯ РЕКУРСИЯ

В теле функции выполняются вложенные рекурсивные вызовы.

$$T(n) = \begin{cases} n - 10, n > 100 \\ 91, \text{иначе} \end{cases}$$

- Решение вложенного соотношения не укладывается в стандартные методы.
- Можно применить метод **мат. индукции** для вывода в явном [closed] виде.

Вспомним

Типы рекурсивных алгоритмов

INDIRECT RECURSION.cpp

```cpp
void func2(int n);

void func1(int n) {
    if (n > 0) {
        std::cout << n;
        func2(n - 1);
    }
}

void func2(int n) {
    if (n > 1) {
        std::cout << n;
        func1(n / 2);
    }
}

int main() {
    func1(20);
    return 0;
}
```

ПЕРЕКРЕСТНАЯ РЕКУРСИЯ

Рекурсивные вызовы нескольких функций выполняются в циклической последовательности.

$$T_A(n) = \begin{cases} 0, n \leq 0 \\ 1 + T_B(n-1), \text{иначе} \end{cases}$$

$$T_B(n) = \begin{cases} 0, n \leq 1 \\ 1 + T_A(n/2), \text{иначе} \end{cases}$$

Решение системы рекуррентных соотношений состоит в выражение $T_A(n)$ через саму себя.

Задача 7

Сложность перекрестной рекурсии

$$T_A(n) = \begin{cases} 0, n \leq 0 \\ 1 + T_B(n-1), \text{иначе} \end{cases}$$

$$T_B(n) = \begin{cases} 0, n \leq 1 \\ 1 + T_A(n/2), \text{иначе} \end{cases}$$

1. Перейдите от системы рекуррентных соотношений так, чтобы время работы $T_A(n)$ зависело только от этой же функции.
   Представьте полное рекуррентное соотношение со всеми граничными условиями.
2. Оцените асимптотические границы временной сложности любым подходящим способом.

Резюме

1. Асимптотический анализ рекурсивных алгоритмов DaC
   - конструктивные методы — **мат. индукция** и **дерево рекурсии**
   - быстрые методы — **мастер-теоремы** и **метод Акра-Баззи**
2. Таксономия рекурсивных алгоритмов
   - **хвостовая** [головная] и **древесная** рекурсия
   - **вложенная** рекурсия
   - **перекрестная** рекурсия
3. Решение системы рекуррентных соотношений