Методы анализа сложности рекурсивных алгоритмов.

- **Итерационные алгоритмы**
  Анализ сложности - определение трудоемкости конструкций «следование», «ветвление», «цикл» с использованием правил суммы и произведения.
- **Рекурсивные алгоритмы**

Основные элементы:

1)метод рекуррентных соотношений (только временная сложность)

2) Теоретико - графовый метод исследования дерева рекурсии.

Метод рекуррентных соотношений .

С каждой рекурсивной процедурой связывают временную функцию t(n), где n определяет объем аргументов процедуры. Затем пытаются построить и решить рекуррентное соотношение, которому удовлетворяет функция t(n).

Рекуррентное соотношение – это уравнение или неравенство, описывающее функцию с использованием её самой, но только с меньшими аргументами. Обычно рекуррентное соотношение описывается в виде системы граничных условий и формулы для общего случая.

Метод не является универсальным, существует ряд ограничений!

- Применим только для оценки временной сложности.
- Позволяет получить только верхнюю оценку для t(n).
- Рекуррентное соотношение для t(n) удается найти только тогда, когда преобразование, уменьшающее значение параметра рекурсии n, линейно относительно n.
- Если рекуррентное соотношение для t(n) найдено, нет ни какой гарантии, что удастся получить асимптотическую оценку так как ***общих методов решения рекуррентных соотношений - нет!***

Методы решения рекуррентных соотношений

1. Метод математической индукции

Заключается в нахождении функции f(n), которая мажорировала бы t(n) для всех значений n (т.е. для всех $n \geq 1$ должно выполняться неравенство $t(n) < f(n)$). Часто, для начала, определяется только вид функции f(n), предполагая, что она зависит от некоторых пока неопределенных параметров (например, $f(n) = an^2$, где a — неопределенный параметр), затем подбираются такие значения параметров, чтобы для всех значений n выполнялось неравенство $t(n) < f(n)$.

Метод состоит из трех шагов:

- делается догадка о виде решения;
- с помощью метода математической индукции доказывается, что решение правильное;
- вычисляются константы.

Пример 1: (точное решение).

$$\begin{cases} f(0)=1 \\ f(n+1)=2*f(n) \end{cases}$$

Предположение: $f(n)=2^n$

Базис: если $f(n)=2^n$, то $f(0)=1$

Индукция: Пусть $f(n)=2^n$, тогда для n+1 $\Rightarrow$ $f(n+1)= 2 * 2^n = 2^{n+1}$

Заметим, что базис существенно влияет на решение, так, например:

Если $f(0)=0$, то $f(n)=0$;

если $f(0)=1/7$, то $f(n)=(1/7)*2^n$; если $f(0)=1/64$, то $f(n)=(2)^{n-6}$

- **Пример 2:** (асимптотическая оценка).

$$T(n) \le \begin{cases} c_1, & \text{если } n = 1, \\ 2T(n/2) + c_2 n, & \text{если } n > 1 \end{cases} \qquad (*)$$

Предположим, что T(n)≤an log(n) , где a пока неопределенный параметр. При n=1 эта оценка «не работает», т.к. выражение an log(n)=0 независимо от значения a. Добавим константу к функции: T(n)≤an log(n)+b. При n = 1 эта оценка правильная, если положить b ≥ c_1.

Далее в соответствии с методом математической индукции предполагаем, что для всех k < n выполняется неравенство

t(k) ≤ ak log(k) + b и попытаемся доказать, что t(n) ≤ an log(n) + b.

Пусть эта оценка верна для k=n/2, т.е. T(n/2)≤a(n/2)log(n/2)+b.

Подставим

$$T(n) \le 2\left(a\frac{n}{2}\log\frac{n}{2} + b\right) + c_2 n =$$

$$= an\log n - an + c_2 n + 2b \le an\log n + b.$$

Последнее неравенство получено в предположении, что a ≥ c_2 + b.

Таким образом, оценка $t(n) < an \log n + b$ справедлива, если будут выполняться неравенства $b \geq c_1$ и $a \geq c_2 + b$. В данном случае можно удовлетворить этим неравенствам, если положить $b = c_1$ и $a = c_1 + c_2$. Вывод: для всех $n > 1$ выполняетс $T(n) \leq (c_1 + c_2) n \log n + c_1$ и следовательно t(n) имеет порядок O(nlog(n)).

2. Оценка решения рекуррентного соотношения методом подстановки.

Суть метода состоит в последовательной подстановке рекурсивного определения, с последующим выявлением общих закономерностей.

В рекуррентном соотношении в правую часть последовательно подставляются выражения для $t(m)$, $m < n$, так, чтобы исключить из правой части все выражения $t(m)$ для $m > 1$, оставляя только $t(1)$. Поскольку $t(1)$ всегда является константой, то в результате получим формулу для $t(n)$, содержащую только n и константы. Такая формула называется "замкнутой формой" для $t(n)$.

Пример 3.

Пусть f(n)=3*f(n/4)+n, тогда:

f(n)=n+3*f(n/4)=n+3*[n/4+3*f(n/16)]=n+3*[n/4+3{n/16+3*f(n/64) }],

и раскрывая скобки, получаем:

$f(n)=n+3*n/4+9*n/16+27*n/64+\ldots+3^i*n/4^i$

Коэффициенты 3^i ни на что не влияют. Справа содержится слагаемое $f(n/4^i)$, остановка рекурсии будет при $f(1)$. Следовательно, приравняв $n/4^i = 1$, получим $i>=\log_4 n$, в этом случае последнее слагаемое не больше чем

$$3^{\log_4 n} * \frac{n}{4^{\log_4 n}} = n^{\log_4 3} O(1)$$

$$f(n) = n\left(\left(\frac{3}{4}\right)^0 + \left(\frac{3}{4}\right)^1 + \left(\frac{3}{4}\right)^2\right) + \cdots + n^{\log_4 3} O(1) =$$

$$= n\sum\left(\frac{3}{4}\right)^k + n^{\log_4 3} O(1), \text{окончательно } f(n) = O(n)$$

Пример 4: рекурсивная программа вычисления факториала.

```
function fact(n:integer):integer;
  begin
(1)    if n<=1 then
(2)                fact:=1
                 else
(3)                fact:=n*fact(n-1)
  end;
```

Естественной мерой объема входных данных для функции fact является значение n. Обозначим через T(n) время выполнения программы. Время выполнения для строк (1) и (2) имеет порядок O(1), а для строки (3) – O(1)+T(n-1). Таким образом, для некоторых констант c и d имеем

$$T(n)=\begin{cases} c+T(n-1), & \text{если } n>1, \\ d, & \text{если } n\le 1. \end{cases} \quad (**)$$

Полагая, что n>2, и раскрывая в соответствии с соотношением(**) выражение T(n-1) (т.е. подставляя в (**) n-1 вместо n), получим T(n)=2c+T(n-2). Аналогично, если n>3, раскрывая T(n-2), получим T(n)=3c+T(n-3). Продолжая этот процесс, в общем случае для некоторого i, n>i, имеем T(n)=ic+T(n-i). Положив в последнем выражении i=n-1, окончательно получаем

T(n)=c(n-1)+T(1)=c(n-1)+d. (***)

Из(***) следует, что T(n) имеет порядок O(n).

Пример 5: (Вернемся к рекуррентному соотношению(*).

$$T(n) \le \begin{cases} c_1, & \text{если } n = 1, \\ 2T(n/2) + c_2 n, & \text{если } n > 1. \end{cases} \quad (*)$$

Заменим в этом соотношении n на n/2, получим $t(n/2) \le 2t(n/4) + c_2 n/2$ и подставим это соотношение вместо $t(n/2)$ в (*), получим:

$$T(n) \le 2(2T(n/4) + c_2 n/2) + c_2 n = 4T(n/4) + 2c_2 n.$$

Аналогично, заменяя в неравенстве (*) n на $n/4$, получаем оценку для $T(n/4)$: $T(n/4) \le 2T(n/8) + c_2 n/4$. Подставляя эту оценку в предыдущее неравенство получим

$$T(n) \le 8T(n/8) + 3c_2 n.$$

Индукцией по i для любого i можно получить следующее соотношение:

$$T(n) \le 2^i T(n/2^i) + ic_2 n.$$

Предположим, что n является степенью числа 2, например $n = 2^k$. Тогда при $i = k$ в правой части неравенства будет стоять $T(1)$:

$$T(n) \le 2^k T(1) + kc_2 n.$$

Поскольку $n = 2^k$, то $k = \log n$. Так как $T(1) \le c_1$, то $T(n) \le c_1 n + c_2 n \log n$.

$T(n)$ имеет порядок роста не более $O(n \log n)$.