Элементы теории алгоритмов

§ 36. Сложность вычислений

Что такое сложность вычислений?

Задачи теории алгоритмов:

- существует ли алгоритм решения задачи?
- можно ли им воспользоваться?

Шахматы:

- алгоритм существует (конечное число позиций)
- полный перебор нереален

Требования к алгоритму:

- быстродействие
- минимальный расход памяти

временна́я сложность

пространственная сложность

Обычно эти требования противоречивы!

Временна́я сложность

T – количество элементарных операций универсального исполнителя (компьютера)

T зависит от размера входных данных n!

Временная сложность алгоритма – функция $T(n)$.

Задача 1. Вычислить сумму первых трёх элементов массива (при $n \geq 3$).

```
Sum:= A[1] + A[2] + A[3]
```

$T(n) = 3$

2 сложения + запись в память

Задача 2. Вычислить сумму всех элементов массива.

```
Sum:= 0
нц для i от 1 до n
   Sum:= Sum + A[i]
кц
```

$T(n) = 2n + 1$

n сложений, $n+1$ операций записи

Временна́я сложность

Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
нц для i от 1 до n-1
  nMin:=i;
  нц для j от i+1 до n
    если A[i]<A[nMin] то nMin:=j все
  кц
  если nMin<>i то
    c:=A[i]; A[i]:=A[nMin]; A[nMin]:=c
  все
кц
```

Число сравнений: $T_c(n) = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

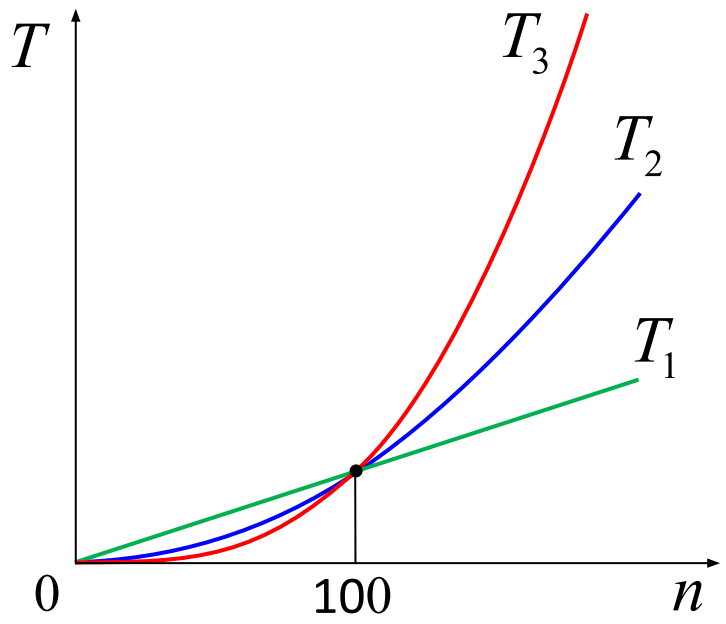
Число перестановок: $T_p(n) \le n-1$

зависит от данных

Сравнение алгоритмов по сложности

$T_1(n) = 10000 \cdot n$ $T_2(n) = 100 \cdot n^2$ $T_3(n) = n^3$

? Какой алгоритм выбрать?



при $n < 100$:

$$T_3(n) < T_2(n) < T_1(n)$$

при $n > 100$:

$$T_3(n) > T_2(n) > T_1(n)$$

! Нужно знать размер данных!

Асимптотическая сложность

Асимптотическая сложность – это оценка скорости роста количества операций при больших значениях N.

линейная

постоянная

сложность $O(N)$ $\Leftrightarrow$ $T(N) \le c \cdot N$ для $N \ge N_0$

сумма элементов массива:

$$T(N) = 2 \cdot N - 1 \le 2 \cdot N \text{ для } N \ge 1 \Rightarrow$$

квадратичная

сложность $O(N^2)$ $\Leftrightarrow$ $T(N) \le c \cdot N^2$ для $N \ge N_0$

сортировка методом выбора:

$$T_c(N) = \frac{1}{2}N^2 - \frac{1}{2}N \le \frac{1}{2}N^2 \text{ для } N \ge 0 \Rightarrow O(N^2)$$

Асимптотическая сложность

кубичная

сложность $O(N^3)$ $\Leftrightarrow$ $T(N) \le c \cdot N^3$ для $N \ge N_0$

сложность $O(2^N)$

сложность $O(N!)$

задачи оптимизации, полный перебор вариантов

Факториал числа *N*: $N! = 1 \cdot 2 \cdot 3 \ldots \cdot N$

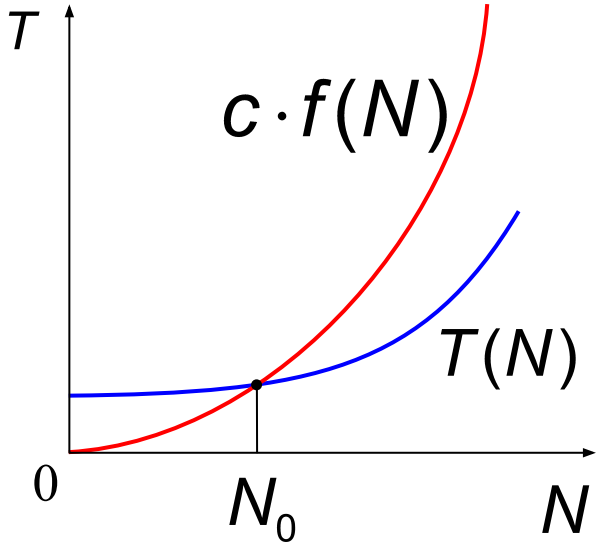
$T(N)$	**время выполнения**
N	100 нс
N^2	10 мс
N^3	0,001 с
2^N	10^{13} лет

N = 100,
1 млрд оп/с

Асимптотическая сложность

Алгоритм относится к классу $O(f(N))$, если найдется такая постоянная c, что начиная с некоторого $N = N_0$ выполняется условие

$$T(N) \le c \cdot f(N)$$



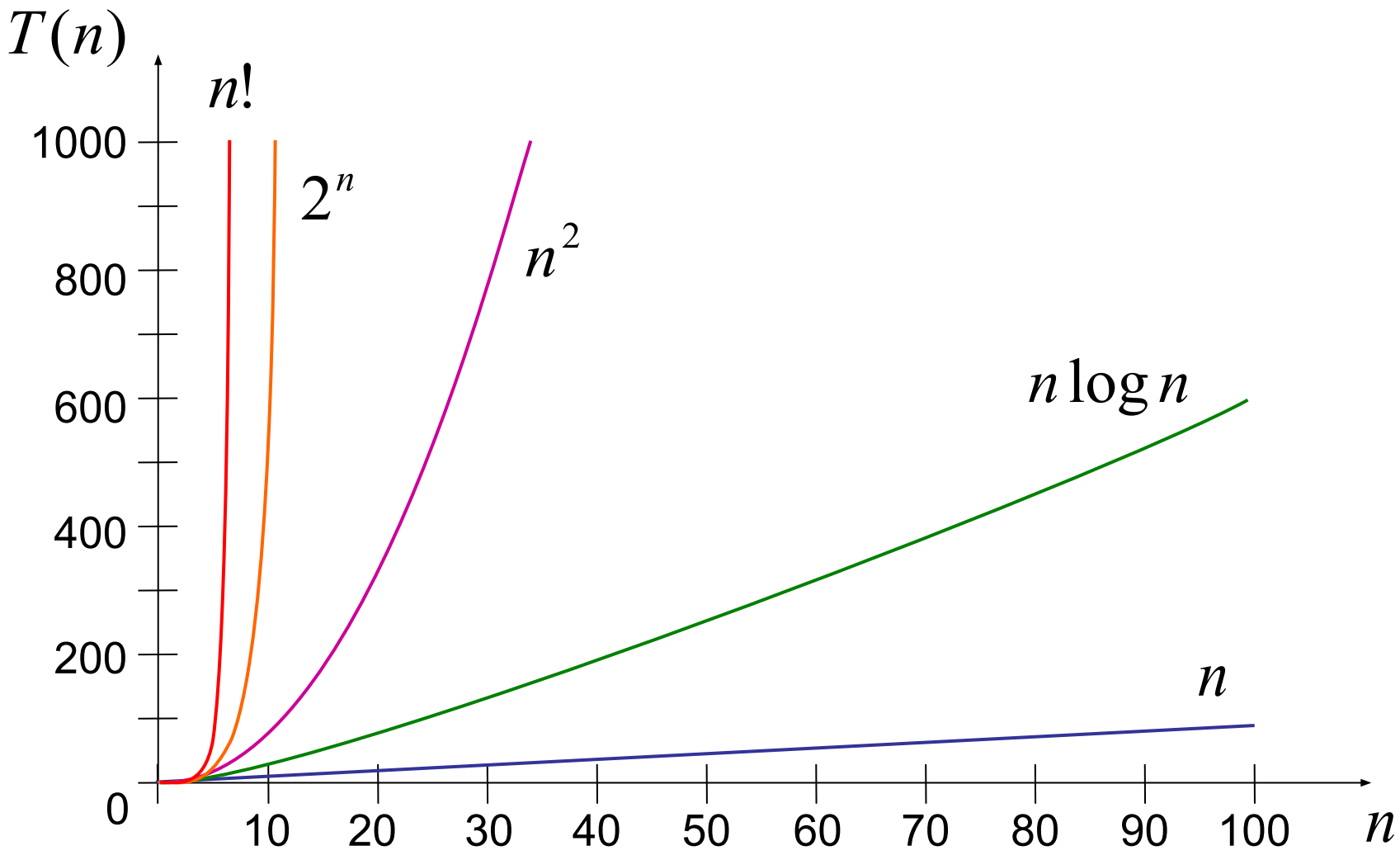
это верхняя оценка!

$$O(N) \Rightarrow O(N^2) \Rightarrow O(N^3) \Rightarrow O(2^N)$$

«Алгоритм имеет сложность $O(N^2)$».

обычно – наиболее точная верхняя оценка!

Асимптотическая сложность



Алгоритмы поиска

Линейный поиск

```
nX:= 0
нц для i от 1 до n
   если A[i] = X то
      nX:= i
      выход
   все
кц
```

? Сложность?

сложность $O(n)$

Алгоритмы поиска

Двоичный поиск

```
L:= 1; R:= n + 1
нц пока L < R - 1
   c:= div(L + R, 2)
   если X < A[c] то
      R:= c
   иначе
      L:= c
   все
кц
```

$\log_2 n$

$$n = 2^m$$

? Сколько шагов?

$$T(n) = m + 1$$

$$T(n) = \log_2 n + 1$$

сложность $O(\log n)$

основание роли не играет

$$\log_a n = \frac{1}{\log_b a} \cdot \log_b n$$

? Какой алгоритм поиска лучше?

Алгоритмы сортировки

Метод «пузырька»

```
нц для i от 1 до n-1
  нц для j от n-1 до i шаг -1
    если A[j] > A[j+1] то
      c:= A[j]; A[j]:= A[j+1]; A[j+1]:= c;
    все
  кц
кц
```

сравнений: $T_c(n) = (n-1) + (n-2) + \ldots + 2 + 1 = \frac{n(n-1)}{2} = \frac{1}{2}n^2 - \frac{1}{2}n$

присваиваний при перестановках:

$$T_p(n) = 3 \cdot \frac{n(n-1)}{2} = \frac{3}{2}n^2 - \frac{3}{2}n$$

сложность $O(n^2)$

Алгоритмы сортировки

Сортировка подсчётом

```
цел C[1:MAX]
нц для i от 1 до MAX
  C[i]:= 0
кц
```

Все значения [1,MAX]!

обнулить массив счётчиков

```
нц для i от 1 до n
  C[A[i]]:= C[A[i]] + 1
кц
```

n

подсчитать, сколько каких чисел

```
k:= 1
нц для i от 1 до MAX
  нц для j от 1 до C[i]
    A[k]:= i
    k:= k + 1
  кц
кц
```

n

заполнить массив заново

сложность $O(n)$

За счёт чего?

Алгоритмы сортировки

При использовании операций «сравнить» и «переставить» сложность не может быть меньше $O(n \cdot \log n)$!

Сортировка слиянием (*Merge sort*) $O(n \cdot \log$

Пирамидальная сортировка (*Heap sort*) $O(n \cdot \log$

$n)$

Быстрая сортировка (*Quick sort*)

в среднем $O(n \cdot \log$

в худшем случае $O(n^2)$