Сложность вычислений

Что такое сложность вычислений?

- **Требования к алгоритму**:
  - Быстродействие – временная сложность
  - Минимальный расход памяти - пространственная сложность

Временна́я сложность

- T – количество элементарных операций универсального исполнителя (компьютера)
- 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]`
- $T(n) = 3$ (так как 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)

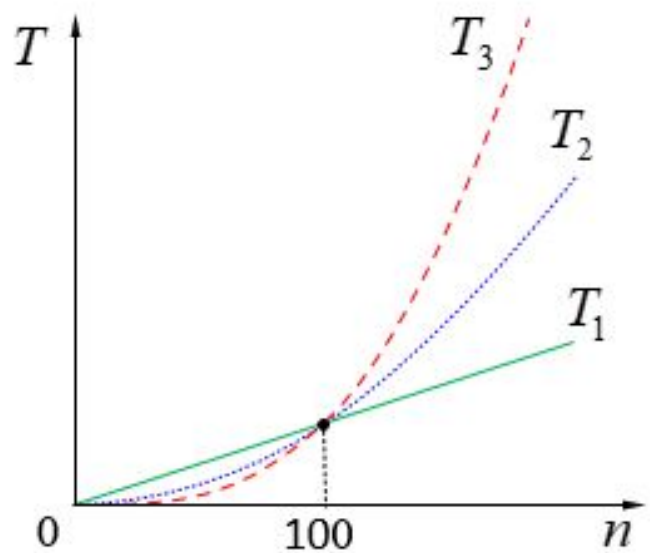
Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
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: $T_p(n) \le n-1$

Сравнение алгоритмов по сложности

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



при $n < 100$:

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

при $n > 100$:

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

Асимптотическая сложность

- **Асимптотическая сложность** – это скорость роста количества операций при больших значениях n.
- сложность $O(n)$ $\Leftrightarrow$ $T(n) \le c \cdot n$ для $n \ge n_0$
- сумма элементов массива:
- $T(n) = 2 \cdot n - 1 \le 2 \cdot n$ для $n \ge 1$ $\Rightarrow$ $O(n)$
- сложность $O(n^2)$ $\Leftrightarrow$ $T(n) \le c \cdot n^2$ для $n \ge n_0$
- сортировка методом выбора:
- $T_c(n) = \frac{1}{2}n^2 - \frac{1}{2}n \le \frac{1}{2}n^2$ для $n \ge 0$ $\Rightarrow$ $O(n^2)$

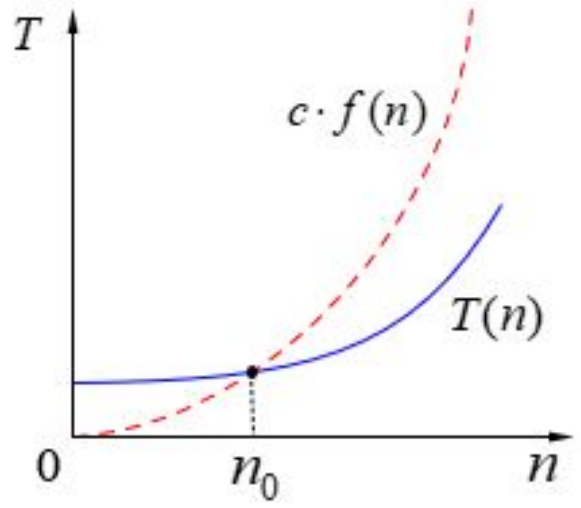
Асимптотическая сложность

• сложность $O(n^3)$ ⇔ $T(n) \le c \cdot n^3$ для $n \ge n_0$

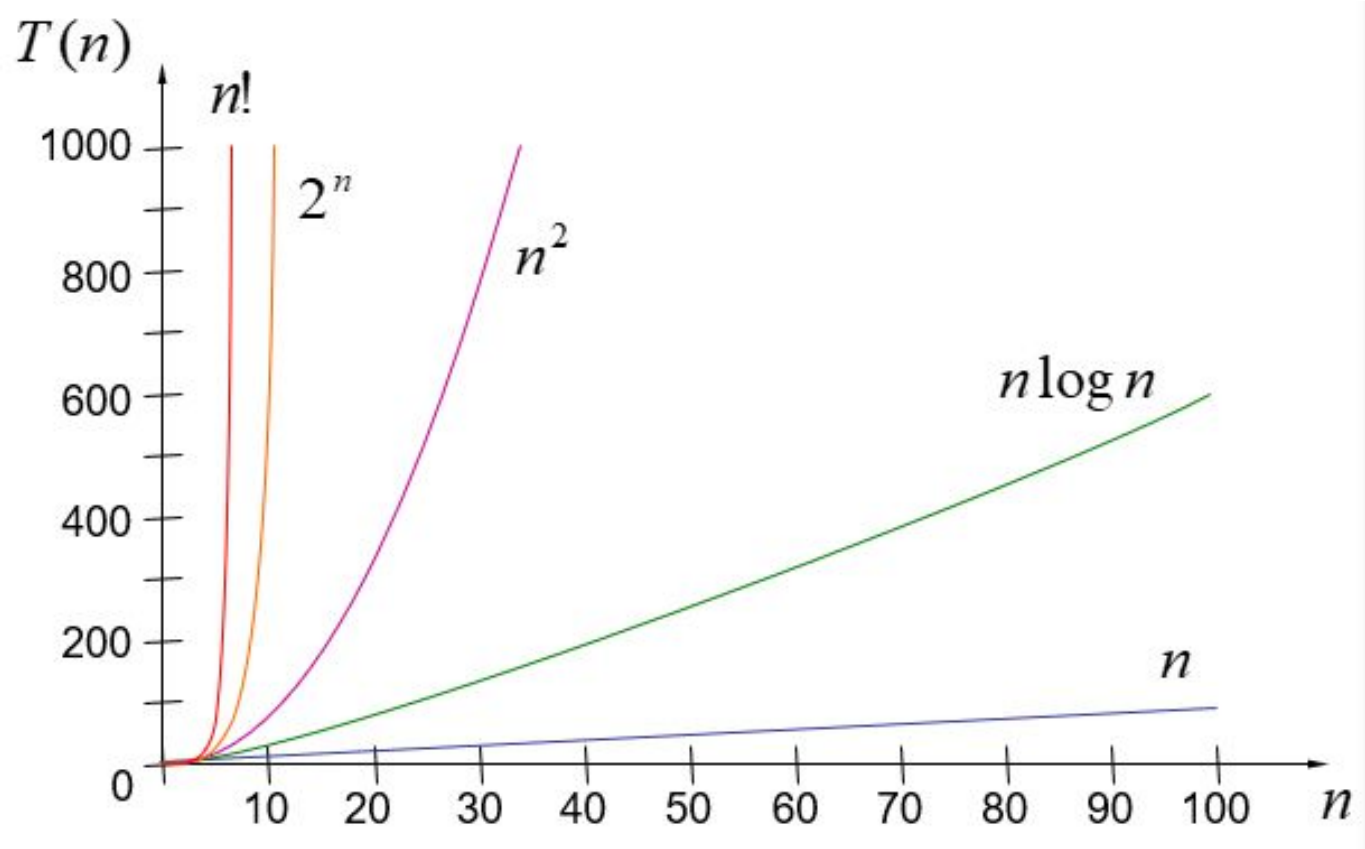
сложность $O(2^n)$
сложность $O(n!)$
} задачи оптимизации, полный перебор вариантов

Алгоритм имеет **асимптотическую сложность** $O(f(n))$, если найдется такая постоянная c, что начиная с некоторого $n = n_0$ выполняется условие

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



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 $O(n)$

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
```

- $n = 2^m$ шагов $T(n) = m + 1$
- $T(n) = \log_2 n + 1$
- сложность $O(\log n)$ основание логарифма роли не играет

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

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
```

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

•сравнений:

•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]  Все значения [1,MAX]!
•нц для i от 1 до MAX
•  C[i]:=0                    обнулить массив счётчиков
•кц
•нц для i от 1 до n
•  C[A[i]]:=C[A[i]]+1    подсчитать, сколько каких чисел
•кц
•k:=1
•нц для i от 1 до MAX
•  нц для j от 1 до C[i]
•    A[k]:=i                  заполнить массив заново
•    k:=k+1                   сложность O(n)
•  кц
•кц
```

Алгоритмы сортировки

- При использовании операций «сравнить» и «переставить» сложность не может быть меньше $O(n \cdot \log n)$!
- **Сортировка слиянием** (*Merge sort*) $O(n \cdot \log n)$
- **Пирамидальная сортировка** (*Heap sort*) $O(n \cdot \log n)$
- **Быстрая сортировка** (*Quick sort*)
- в среднем $O(n \cdot \log n)$!
- в худшем случае $O(n^2)$
-