

1.2 АСИМПТОТИЧЕСКИЙ АНАЛИЗ АЛГОРИТМОВ

Основная задача анализа алгоритмов –

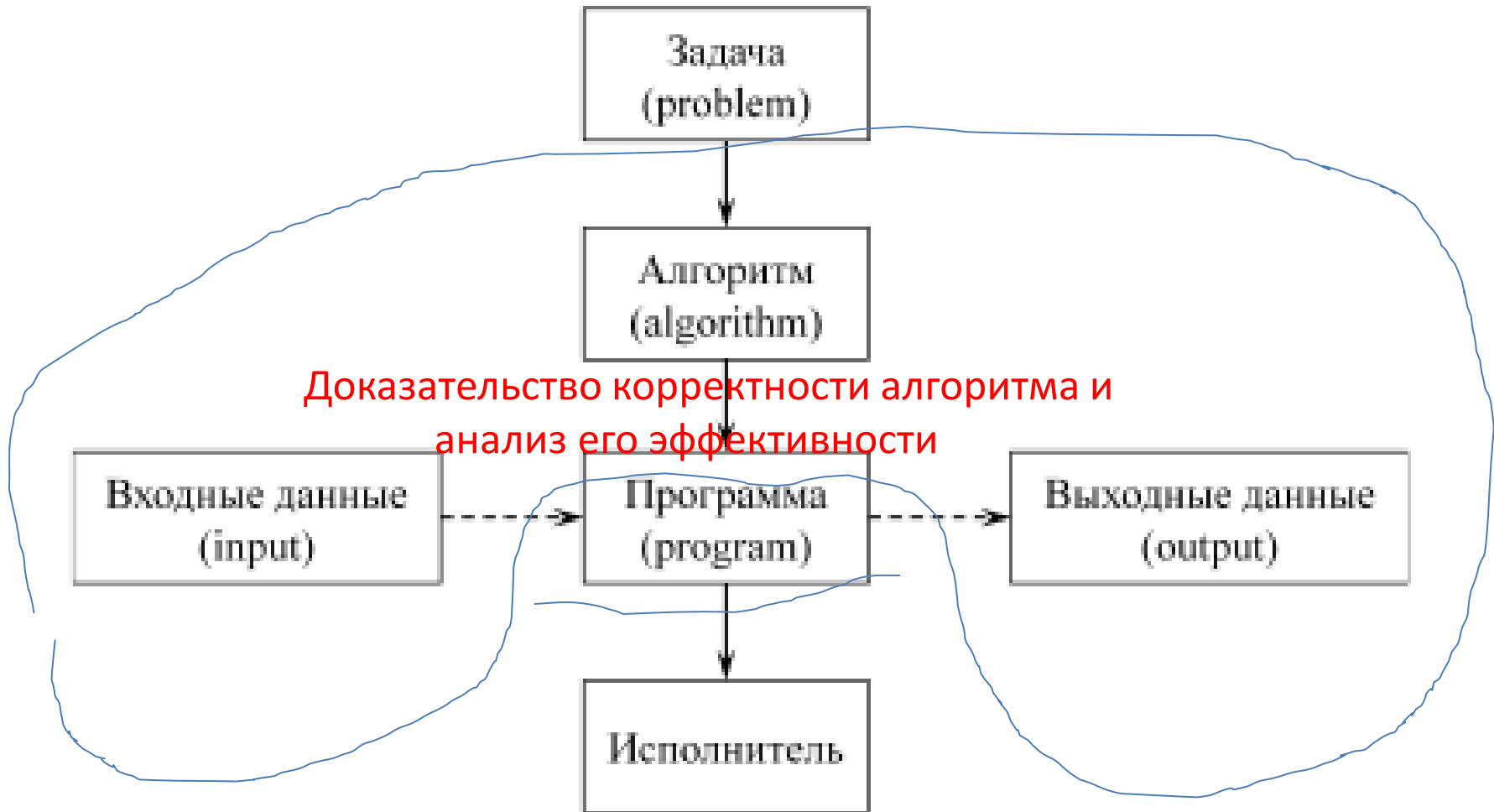
выявление зависимости масштабирования
требований к ресурсам (затраты времени и
памяти) от размера входных данных

Асимптотический анализ проводится в
предположении неограниченного роста
входных данных

1.2.1

**Размерность задачи и
трудоемкость алгоритма**

Процесс решения задачи



Модель и размерность задачи

Опр. 1 Формальная постановка задачи – ее описание в виде функции, на вход которой поступают формальные параметры, задающие входные и выходные данные

!!! **Количество выполненных операций алгоритма напрямую зависит от размера входных данных**

Опр. 2 Размерность L задачи – количество информации, достаточное для ее формального описания

Как измерить информацию?

- История
- Впервые ввести меру информации попытался Р. Хартли в 1928 г. В своих рассуждениях он исходил из интуитивной идеи о том, что сообщение, состоящее из n символов, должно нести в n раз больше информации, чем сообщение, состоящее из одного символа
- Единственной функцией, которая удовлетворяет этому свойству, является логарифмическая

Маловероятные события несут много информации!

$$\begin{array}{l} p(A = a_1) = p_1 \in [0, 1], \\ p(A = a_2) = p_2 \in [0, 1], \\ \dots\dots\dots \\ p(A = a_n) = p_n \in [0, 1]. \end{array} \quad \Rightarrow \quad \sum_{i=1}^n p_i = 1.$$

$$p_1 = p_2 = \dots = p_n, \text{ то } p(A = a_i) = \frac{1}{n}, \forall i = 1, \dots, n$$

$$M(A) = \sum_{i=1}^n a_i \cdot p_i \stackrel{p_i=1/n}{=} \sum_{i=1}^n a_i \cdot \frac{1}{n} = \frac{1}{n} \sum_{i=1}^n a_i$$

Количество информации

- **Опр. 3** Количество информации, которой достаточно для знания того, что событие A произошло (достаточно для разрушения неопределенности об объекте), определяется по формуле

$$i(A) = \log_x \frac{1}{P(A)}, \quad x > 1$$

Бит (ср. с понятиями кубит, кудит)

- **Бит** – единица измерения информации в случае, если основание логарифма равно 2:

$$i(A) = \log_2 \frac{1}{P(A)} = -\log_2 P(A) \text{ (бит)}$$

Размерность L задачи – это минимальное количество бит, достаточное для описания входных данных задачи

$$L = \left\lceil \log_2 \frac{1}{P(A)} \right\rceil = \lceil -\log_2 P(A) \rceil \text{ (бит)}$$

Средняя информация (энтропия)

Энтропия источника, сопоставленного событиям, – это **среднее количество бит**,
необходимое для кодирования информации

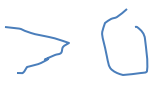
$$H(A) = \sum_{k=1}^n i(a_k) P(a_k)$$

$$H(A) = \frac{1}{n} \sum_{k=1}^n i(a_k) \quad \# P(a_k) = 1/n$$

Замечания

- 1. Если на вход алгоритма поступает некоторое натуральное число n , то предполагается, что оно может быть выбрано из множества **первых n натуральных чисел равномерно**
- 2. Если на вход алгоритма поступает некоторая последовательность чисел, то предполагается, что все ее элементы могут быть выбраны из некоторого **явно** заданного множества X **равномерно**

Примеры

Данные	Формат данных	Размерность
x	целое число из множества $\{1, \dots, x\}$ $x > 1$	$\lceil \log_2 x \rceil$
	целое число из множества $\{0, \dots, x\}$	$\lceil \log_2 (x + 1) \rceil$
	целое число из множества $\{-x, \dots, x\}$	$\lceil \log_2 (2 \cdot x + 1) \rceil$
	рациональное число $x = \frac{a}{b}$ 	$\lceil \log_2 (a \cdot b) \rceil$
x_1, \dots, x_m	целые числа x_i из множества $\{1, \dots, x\}$	$l = m \cdot \lceil \log_2 x \rceil$
	целые числа x_i из множества $\{0, \dots, x\}$	$l = m \cdot \lceil \log_2 (x + 1) \rceil$

Вывод

- Если предположить, что размерность машинного слова достаточна для представления любого числа, то размерность задачи будет ограничена количеством исходных данных в ее формальном описании

Трудоёмкость алгоритма

- **Опр. 4** Трудоёмкость алгоритма – это функция от размерности задачи, которая оценивает сверху требуемое время для ее решения
- Временная сложность – $T(L)$
- Пространственная (ёмкостная) сложность – размерность памяти – объем памяти, требуемый для реализации алгоритма

Как вычислить время, которое алгоритм затрачивает на решение задачи?

- Прежде всего – определить модель вычислительного устройства, которое используется для реализации алгоритма
- **RAM** (РАМ) –модель вычислительного устройства (Random Access Machine-Равнодоступная адресная машина)

Принятые соглашения для RAM (неограниченная память)

1. Арифметические и логические операции выполняются за 1 временной шаг (1 такт)
2. Каждое обращение к ячейке ОП требует 1 такта
3. Выполнение условного перехода требует вычисления логического выражения и одной из ветвей решения
4. Выполнение цикла подразумевает выполнение всех его итераций, выполнение каждой итерации требует вычисления условия завершения цикла и выполнения его тела

Если взять в качестве

Меры асимптотической сложности

число выполняемых команд разветвления
(программа – дв. дерево), то

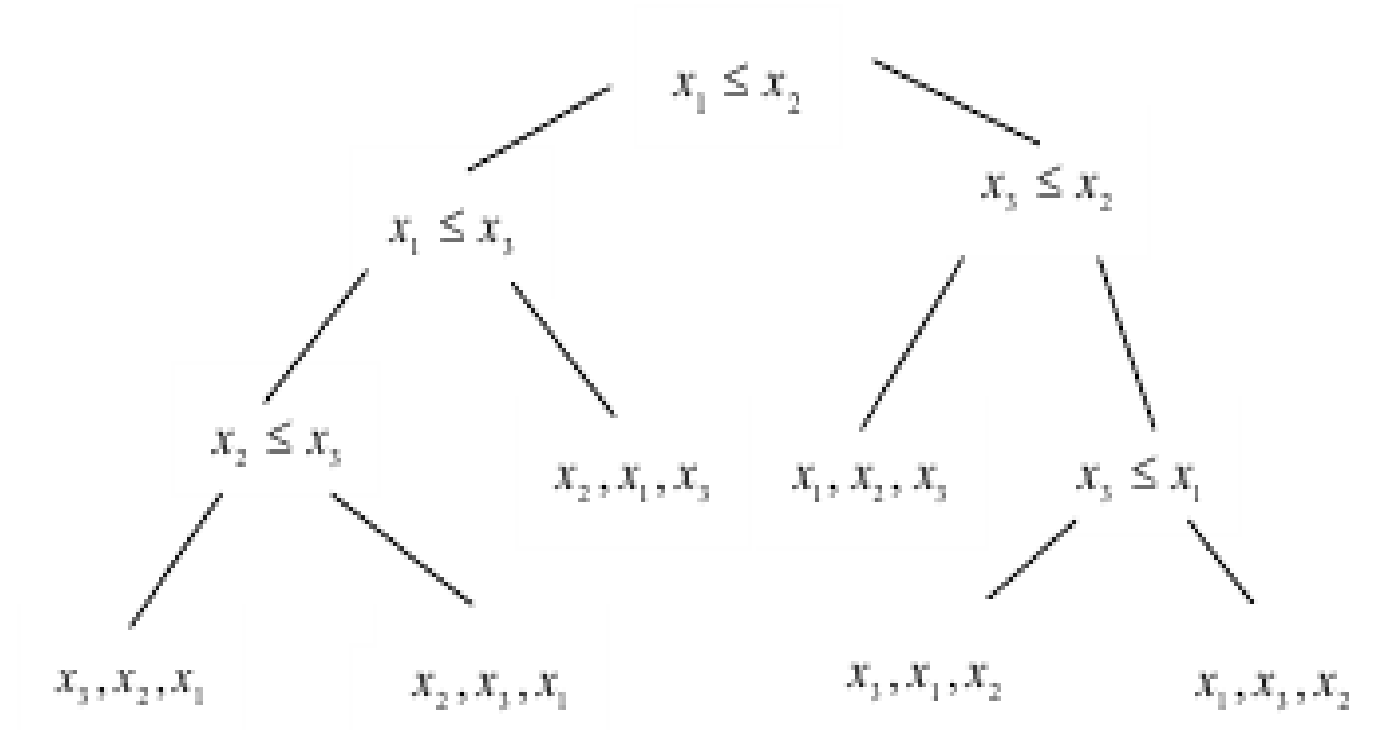
Временная сложность – высота двоичного
дерева



$$T(L) = h$$

Число узлов и листьев при этом могут
значительно превосходить высоту дерева

Дерево решений для сортировки трех чисел ($h!$ листьев)



ВЫВОД

- Для определения трудоемкости алгоритма нужно посчитать количество операций, но при этом нужно иметь в виду, что
- классы входных данных **существенно** влияют на эту величину!

Уровни оценки сложности алгоритма

- 1.Худший случай (max число операций)
- 2.Средний случай
- 3.Лучший случай (min число операций)

Поиск наибольшего элемента (число операций присваивания)

```
‣  $P := \text{combinat}[\text{permute}](4) :$   
‣ for  $k$  to  $\text{nops}(P)$  do  
   $B := P[k]; \text{lrg} := B[1]; j := 0;$   
  for  $i$  from 2 to  $\text{nops}(B)$  do  
    if  $B[i] > \text{lrg}$  then  
       $\text{lrg} := B[i]; j := j + 1;$   
    fi;  
  od;  
   $\text{lrg}, j;$   
od;
```

#j – счетчик операций
присваивания

Пример работы с
одной
перестановкой

$B := [1, 2, 3, 4]$
→ $\text{lrg} := 1$
 $j := 0$
4
3

Результаты эксперимента (n=4)

Количество выполнений операции присваивания	Количество перестановок	Вероятность (частота)
0 – лучший случай	6	1/4
1	11	11/24
2	6	1/4
3 – худший случай	1	1/24

$$f_A(4) = 0 \cdot \frac{1}{4} + 1 \cdot \frac{11}{24} + 2 \cdot \frac{1}{4} + 3 \cdot \frac{1}{24} \approx 1.18(\text{опер})$$

Результаты эксперимента (n=10)

Количество выполнений оператора	Количество перестановок	Вероятность
0	362880	0,1
1	1026576	0,282896825
2	1172700	0,323164683
3	723680	0,199426808
4	269325	0,07421875
5	63273	0,017436343
6	9450	0,002604167
7	870	0,000239749
8	45	1,24008e-05
9	1	2,75573e-07

$$f_A(10) \approx 1,929$$

С использованием
частичной суммы
гармонического ряда



$$f_A(10) \approx 1,933$$

Методичка – начало!!!!

Подсчет среднего времени с использованием рекурсивного алгоритма

- MaxRec(a[1..n])

$$H_n \approx \ln n + 0,58 + \frac{1}{2n}$$

1 Item m

2 If n>1

3 then m:=MaxRec(a[1..n-1])

4 if m<a[n]

5 then m:=a[n] #вып-ся когда наиб. в поз. n

6 return m

7 Return a[1] (на доске вычисления!)

Подсчет операций «в среднем»

1. Разбить входные данные на группы с одинаковым временем (пусть n групп (GR))
2. Вычислить $P_i = P(\text{вх.данные из GR}_i)$, $i=1..n$
3. t_i – время (число операций) на данных из GR_i

4.
$$T_A(n) = \sum_{i=1}^n P_i t_i$$

Замечание. Формула упрощается при одинаковых вероятностях.

Например, для примера выше:

$$T = (0+1+2+3)/4 = 1.5 \text{ (ср. с 1.8 из эксперимента)}$$

Последовательный поиск

элемента в массиве

(оценка трудоемкости по количеству операций сравнения)

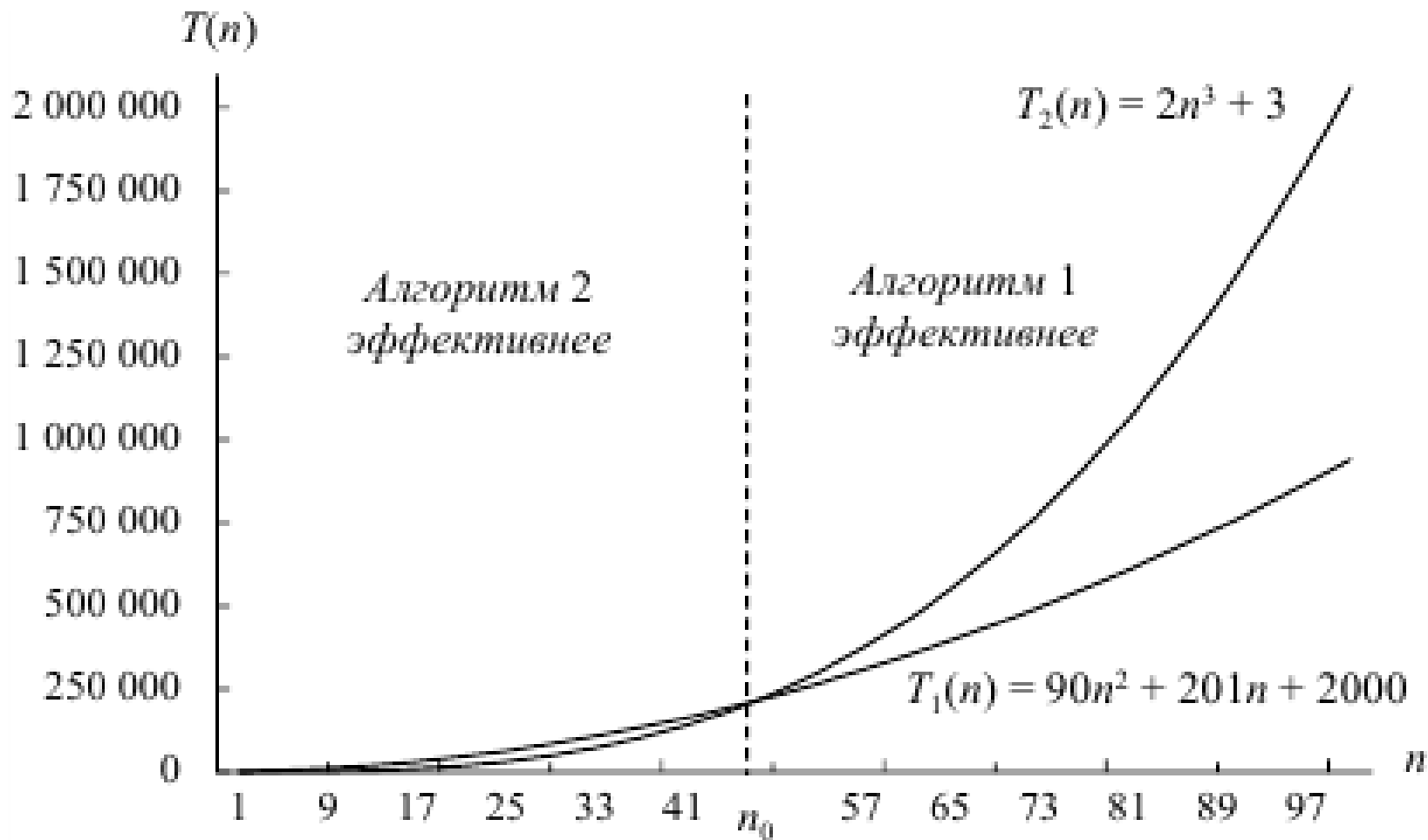
- Разбираем «на доске»

1.2.2. Асимптотические оценки

$$T_1(n) = 90n^2 + 201n + 2000$$

$$T_2(n) = 2n^3 + 3$$

Сравнение трудоемкостей двух алгоритмов



n	$T_1(n)$	$T_2(n)$	Сравнение алгоритмов
10	13 010	2 003	Первый алгоритм в 6.5 раза медленнее второго: $T_1(n)/T_2(n) = 6.5$
20	42 020	16 003	Первый в 2.6 раза медленнее второго
30	89 030	54 003	Первый в 1.6 раза медленнее второго
40	154 040	128 003	Первый в 1.2 раза медленнее второго
50	237 050	250 003	Первый в 1.1 раза быстрее второго
60	338 060	432 003	Первый в 1.3 раза быстрее второго
70	457 070	686 003	Первый в 1.5 раза быстрее второго
80	594 080	1 024 003	Первый в 1.7 раза быстрее второго
90	749 090	1 458 003	Первый в 1.9 раза быстрее второго
100	922 100	2 000 003	Первый в 2.2 раза быстрее второго
1 000	9 0203 000	2 000 000 003	Первый в 22 раза быстрее второго
10 000	9 002 012 000	2 000 000 000 003	Первый в 222 раз быстрее второго

Вывод

- Особую значимость приобретает оценка трудоемкости алгоритма при
 - – больших объемах данных;
 - – размер данных **неизвестен**
- **Основная проблема** – определение характера роста числа операций алгоритма при увеличении размера входных данных

Скорость (порядок) роста функции при увеличении аргумента

- Асимптотические методы («на бесконечности»)

Задача

$$H_n = 1 + \frac{1}{2} + \dots + \frac{1}{n} = \sum_{k=1}^n \frac{1}{k}$$

- **Вопрос :**
- С какой скоростью происходит увеличение частичных сумм гармонического ряда?

Пример в качестве подсказки

Как изменить гармонический ряд, чтобы получить сходящийся ряд? С каким рядом «сравним» гармонический ряд?

$$\sum_{n=1}^{\infty} \left(\frac{1}{n} - \ln \left(\frac{n+1}{n} \right) \right)$$

**Начать – с оценки общего члена ряда,
затем – по определению суммы ряда**

$$a_n = \frac{1}{n} - \ln \left(1 + \frac{1}{n} \right)$$

Нахождение суммы ряда в Maple

Untitled (2)* - [Server 3] - Maple 18

File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

maximize

Start.mw *Untitled (2)*

Text Math Drawing Plot Animation

2D Input Times New Roman 12 B I U

$$> \text{Sum}\left(\frac{1}{n} - \ln\left(\frac{n+1}{n}\right), n = 1 \dots \text{infinity}\right) = \text{sum}\left(\frac{1}{n} - \ln\left(\frac{n+1}{n}\right), n = 1 \dots \text{infinity}\right)$$

$$\sum_{n=1}^{\infty} \left(\frac{1}{n} - \ln\left(\frac{n+1}{n}\right) \right) = \gamma$$

$$> \gamma = \text{evalf}(\gamma)$$

$$\gamma = 0.5772156649$$

Постоянная Эйлера

$$> \text{int}\left(\frac{\ln(x)}{\exp(x)}, x = 0 \dots \text{infinity}\right)$$

$$-\gamma$$

$$>$$

$$>$$

$$>$$

$$>$$

$$>$$

$$>$$

Ready

C:\Program Files\Maple 18 Memory: 20.18M Time: 10.93s Math Mode

Введите здесь текст для поиска

19:56 30.09.2019

Сравнение ББП в матанализе

- Асимптотическое сравнение!

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \begin{cases} 0, & f \ll g \\ Const, & f \approx g \quad (f \sim g) \\ \infty, & f \gg g \end{cases}$$

$$T_1 \ll T_2$$

- Шкала роста:

$$a \ll \log_a n \ll n^\alpha \ll n \ll n^a \ll a^n \ll n! \ll n^n$$

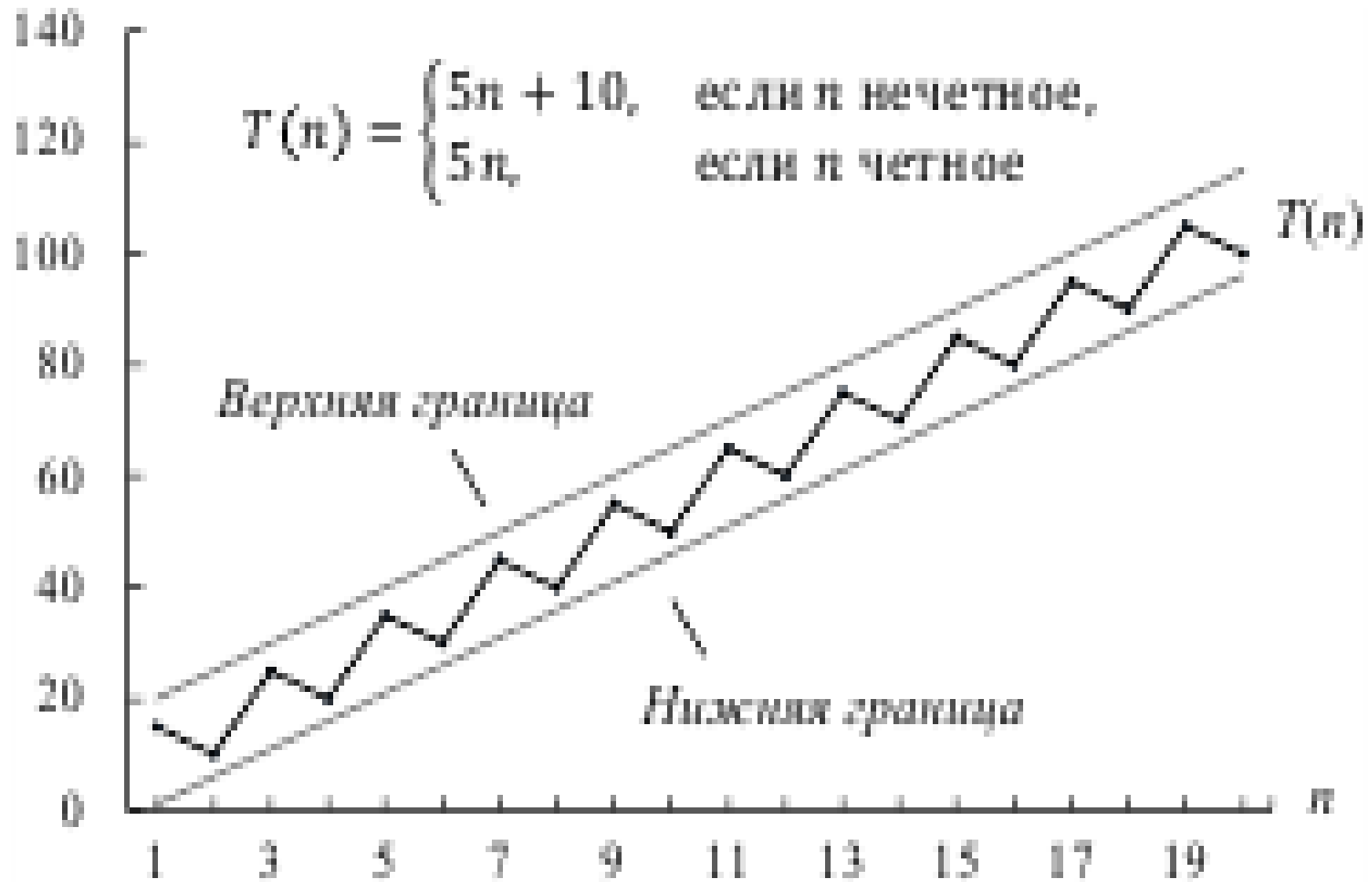
$\lg n$	$\log_2 n$	n	$n \log_2 n$	n^2	2^n	$n!$
0	0	1	0	1	2	1
0.3	1	2	2	4	4	2
0.5	1.6	3	5	9	8	6
0.6	2.0	4	8	16	16	24
0.7	2.3	5	12	25	32	120
0.78	2.6	6	16	36	64	720
0.85	2.8	7	20	49	128	5 040
0.90	3	8	24	64	256	40 320
0.95	3.2	9	29	81	512	362 880
1	3.3	10	33	100	1024	3 628 800
3	10	1 000	9 966	1 000 000		
4	13.3	10 000	132 877	100 000 000		
5	16.6	100 000	1 660 964	10 000 000 000		
6	19.9	1 000 000	19 931 569	1 000 000 000 000		

Цель асимптотического анализа

$$\lim_{n \rightarrow \infty} \frac{T(n)}{f(n)} = \textit{Const} (= 1)$$

Одна из проблем оценки $T(n)$

$$T(n) = \begin{cases} 5n + 10, & \text{если } n \text{ нечетное,} \\ 5n, & \text{если } n \text{ четное} \end{cases}$$

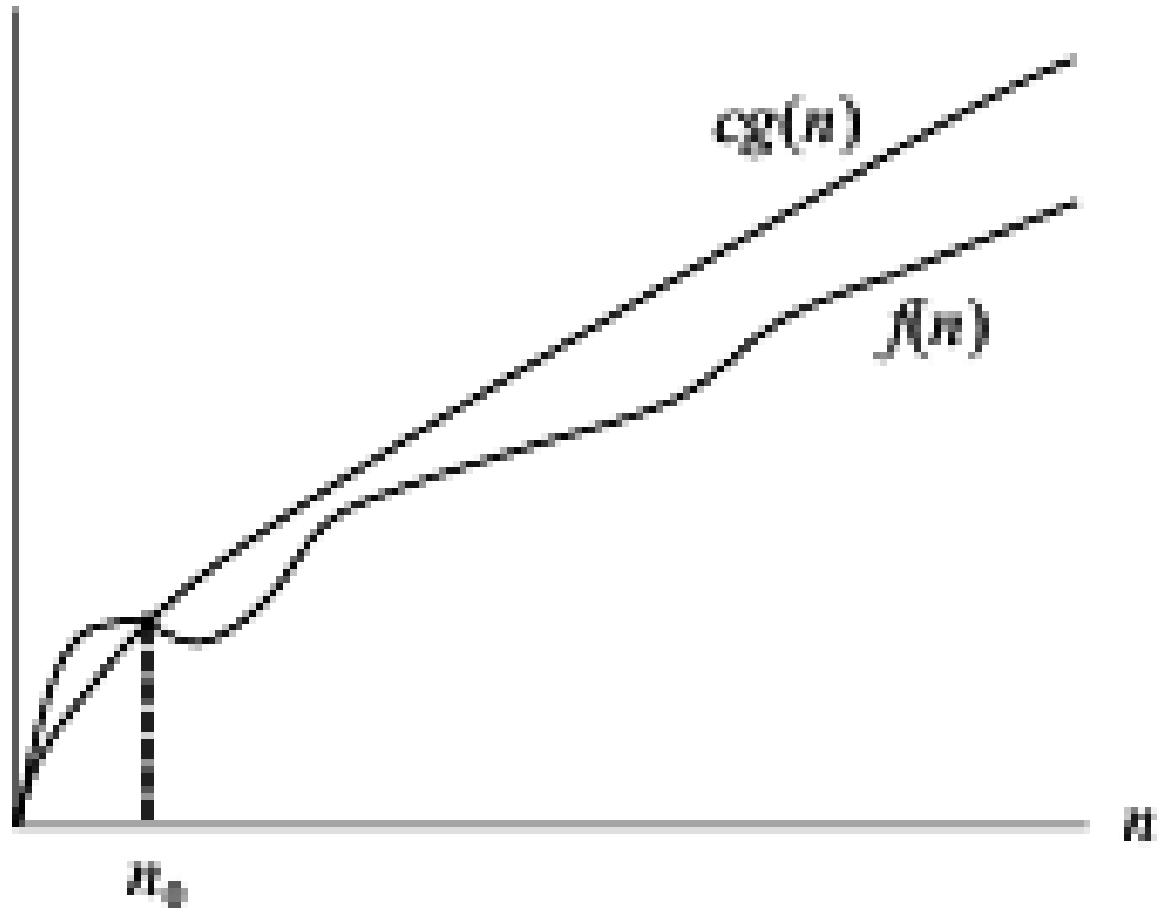


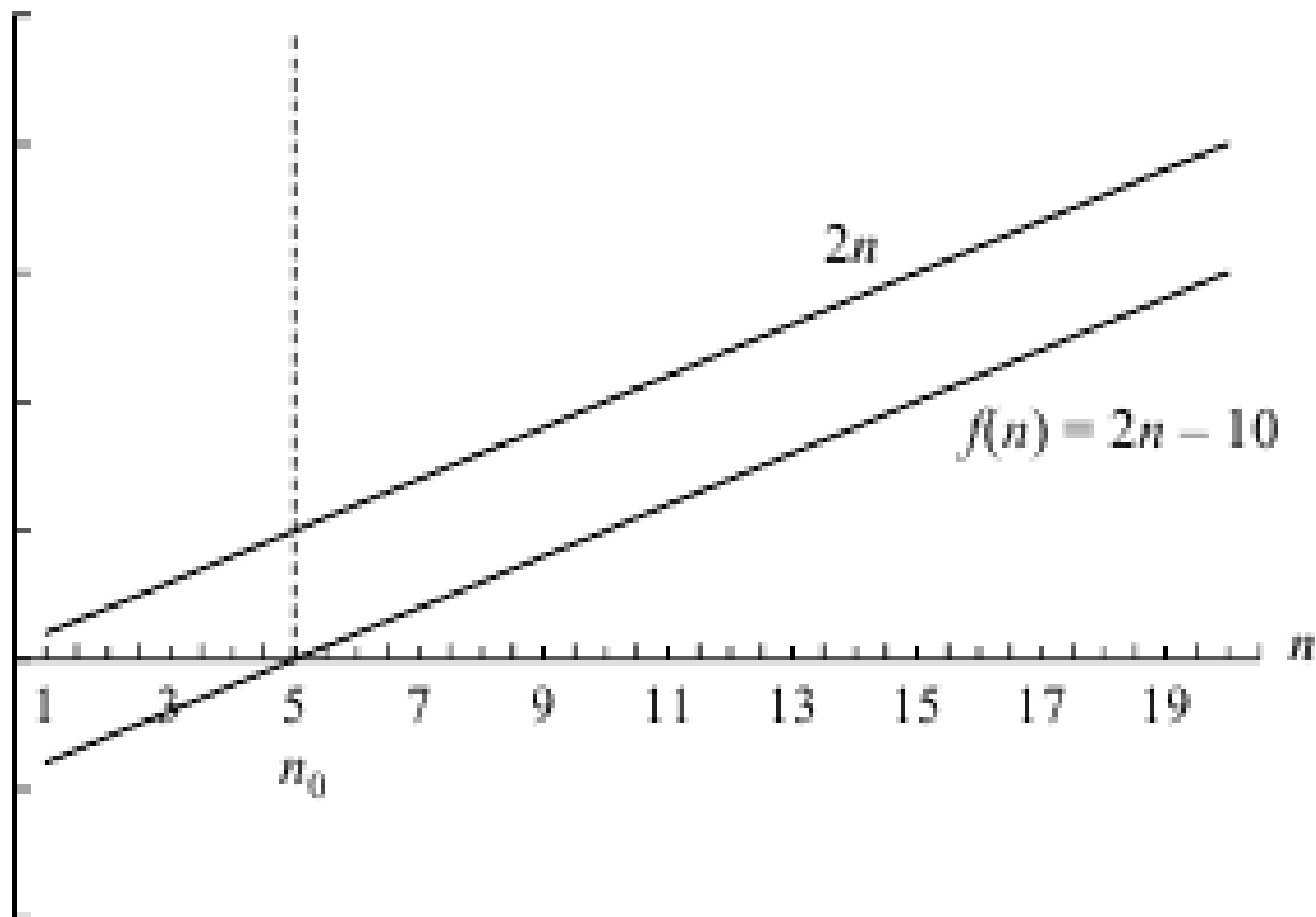
Классы асимптотических оценок

- Определения даны в аудитории при:

$$f(n) > 0, \quad g(n) > 0, \quad n \in \mathbb{N}$$

Верхняя асимптотическая оценка (O)





Start.mw Untitled (2) *наибольший элемент.mw

Text Math Drawing Plot Animation

2D Input Times New Roman 12 B I U

Hide

Favorites

MapleCloud (Off)

Expression

Calculus

Common Symbols

π e i j I
 ∞ Σ Π \int d
 \cap \cup \geq $>$ \neq
 \leq $<$ \neq \neq
 α \approx \sim $=$ \neq
 \equiv \neq \in \notin \subseteq
 \setminus \emptyset \exists \forall \neg
 \wedge \vee \Rightarrow \Leftrightarrow
 \mathbb{R} \mathbb{N} \mathbb{Q} \mathbb{Z} \mathbb{R}
 \S $:=$ \parallel $'$ $+$
 $-$ \times $/$ \pm \mp
 \circ \ast \cdot \cdot ∇
 $!$ \hbar ℓ \perp

Live Data Plots

Variables

Variable	Value
B	$[4, 3, 2, \dots]$
i	5
j	0
k	25
lrg	4
P	$\{(1, 2, \dots)$

Matrix

Units (SI)

Units (FPS)

Layout

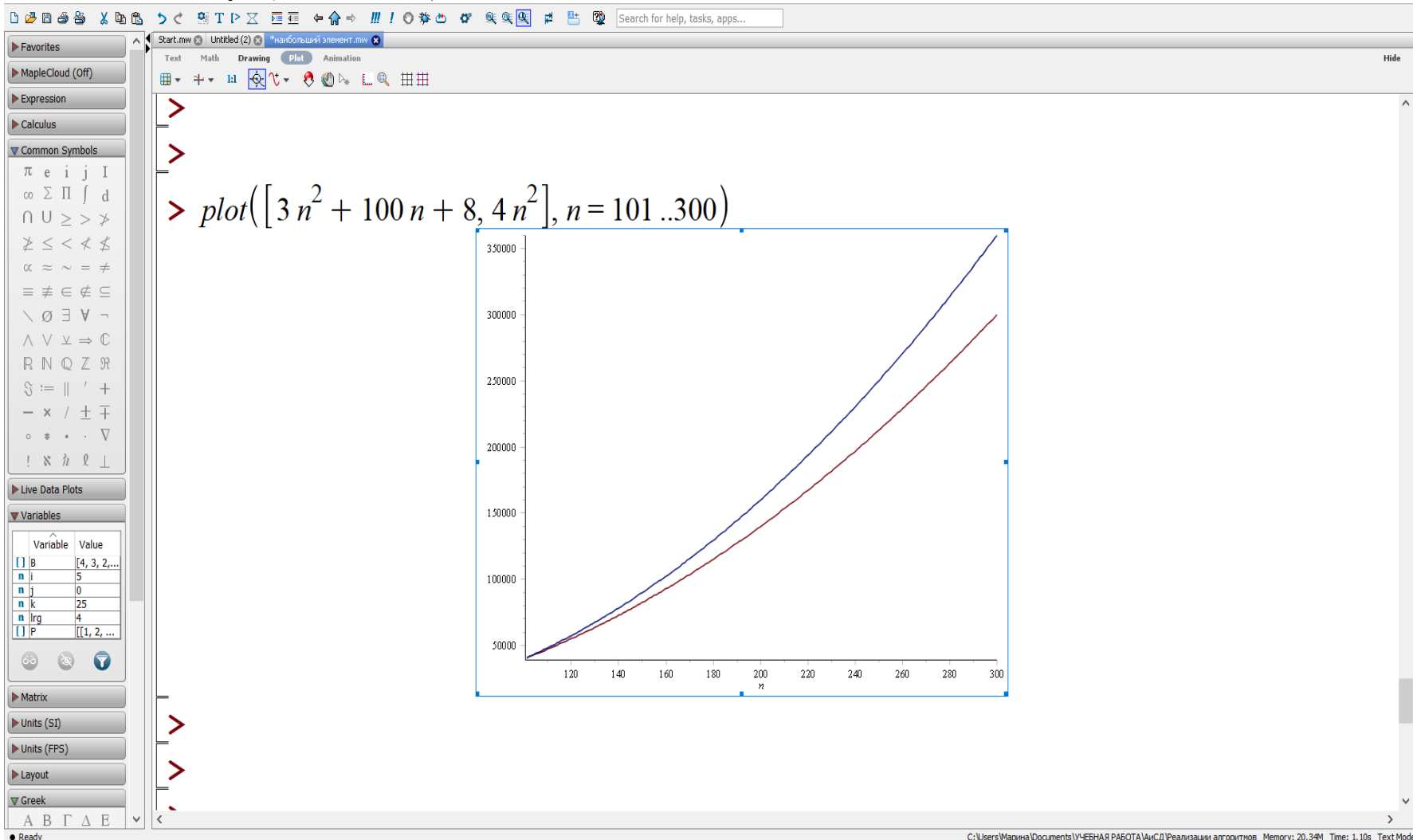
Greek

A B Γ Δ E

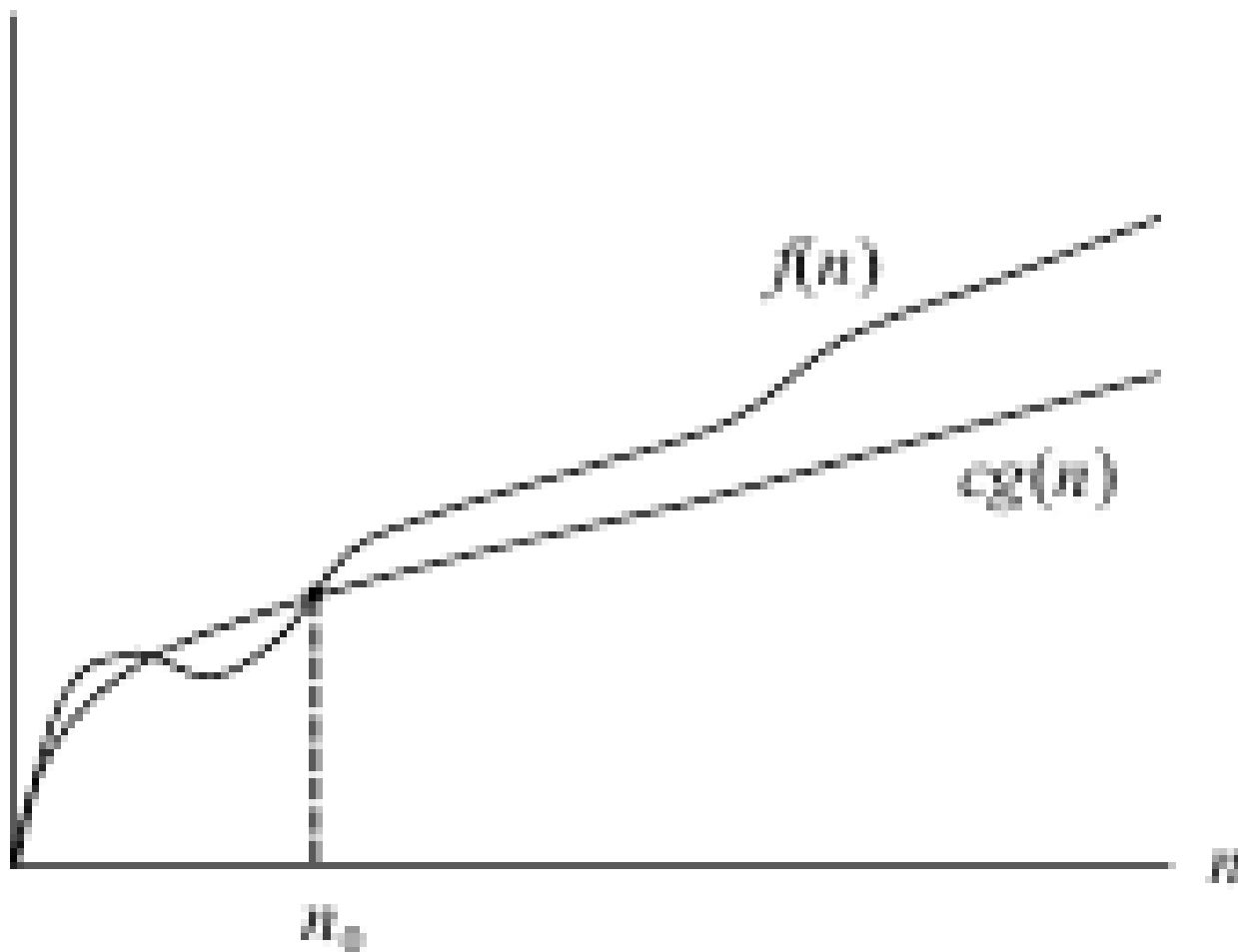
$$\text{solve}(3n^2 + 100n + 8.0 = 4n^2)$$

$$-0.07993610220, 100.0799361$$

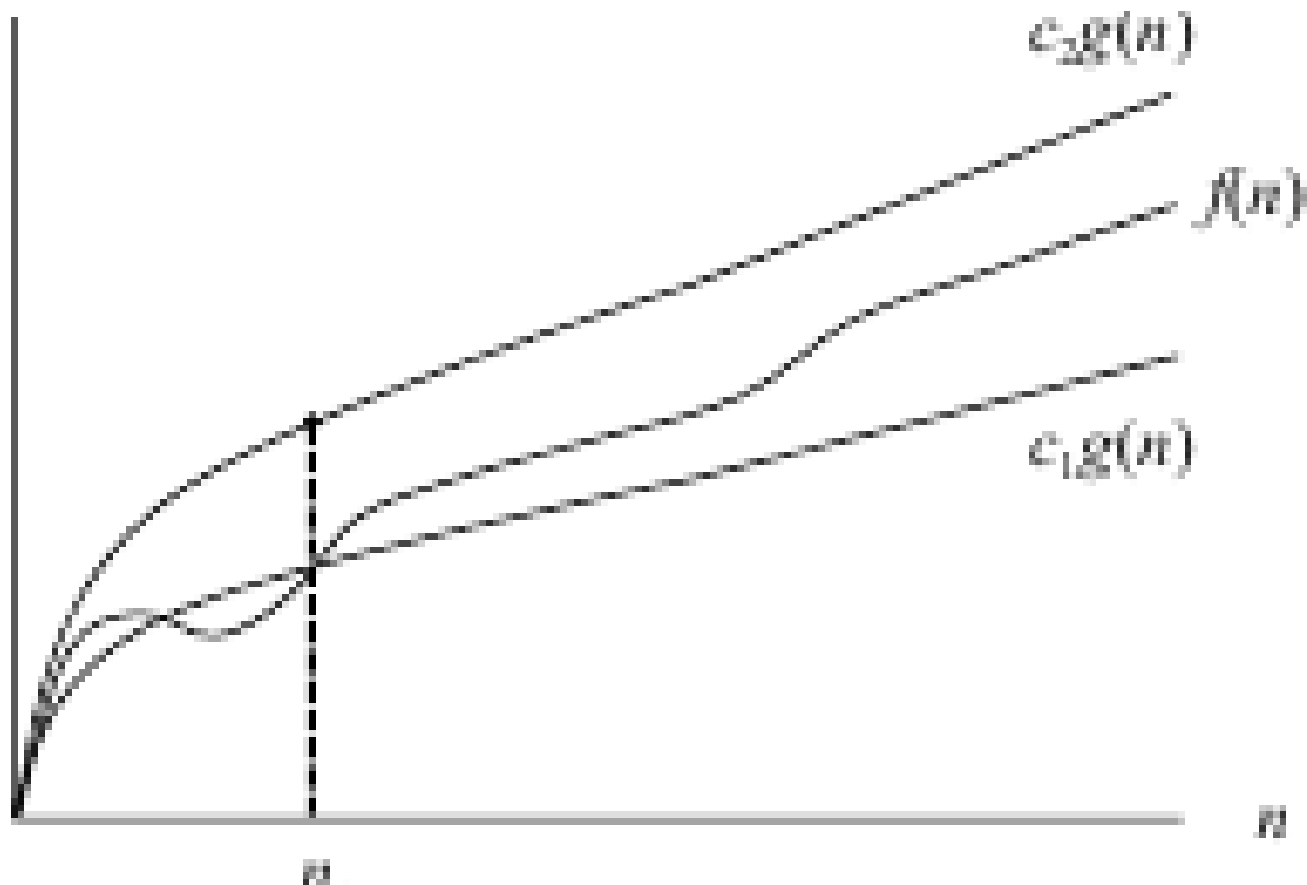
(6)

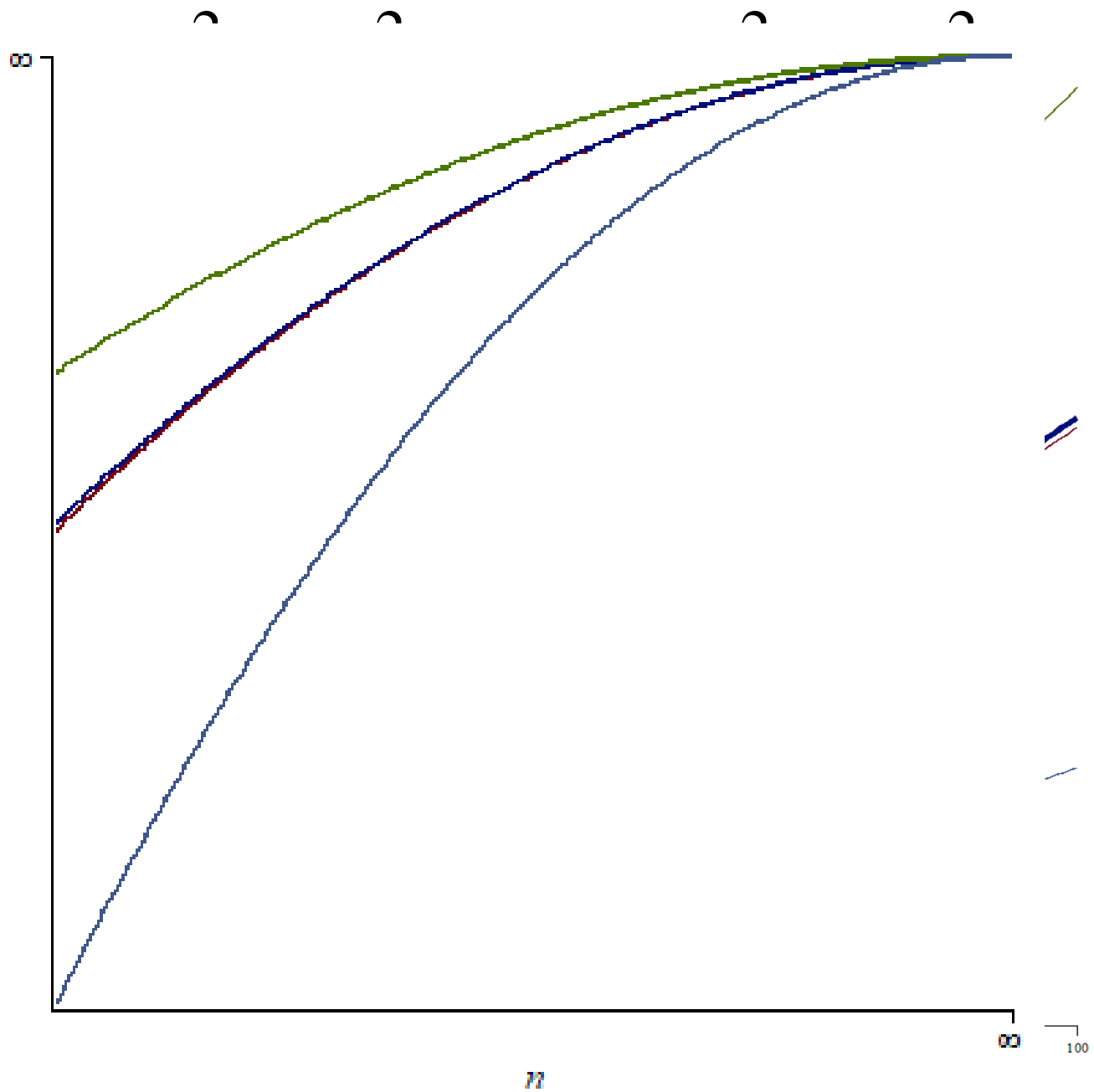


Нижняя асимптотическая оценка (Ω)



Точная асимптотическая оценка (Θ)





Теорема с доказательством!

- О связи асимптотик

Слабая верхняя оценка (o)


$$100n = o(n^2) ?$$

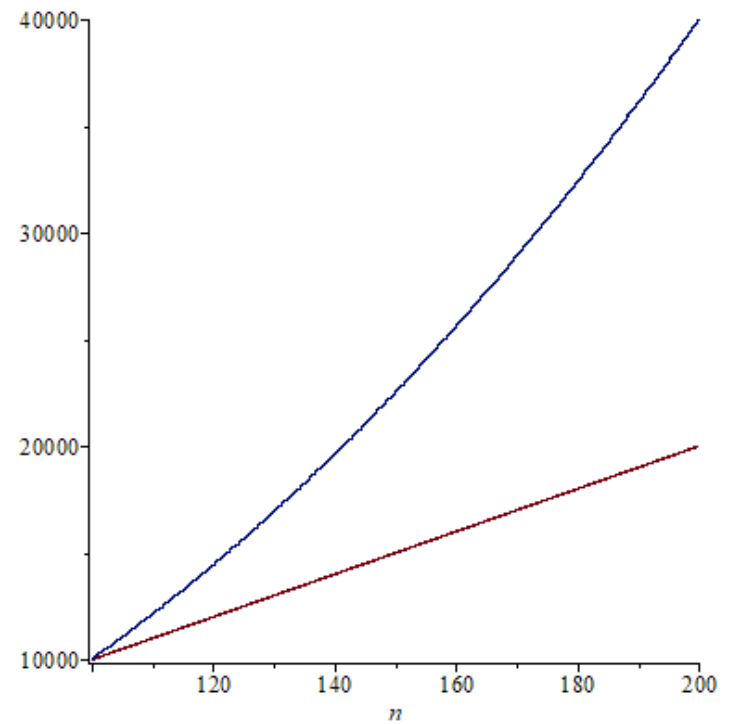
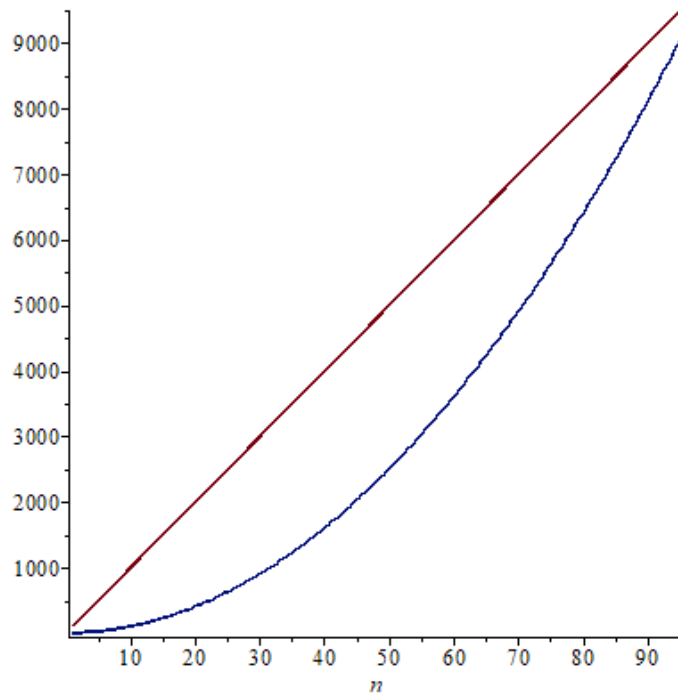
$$4n^2 + 2n = o(n^2) ?$$

o

$$100n \ll n^2$$

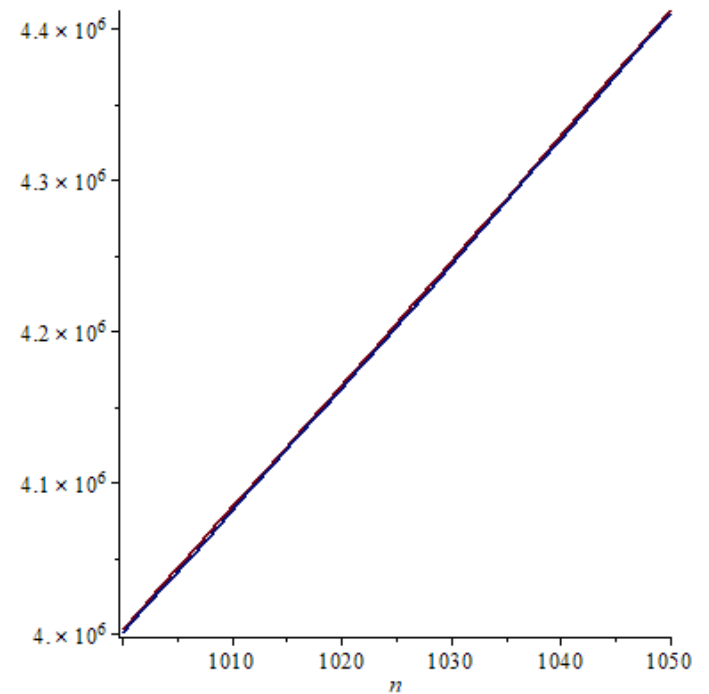
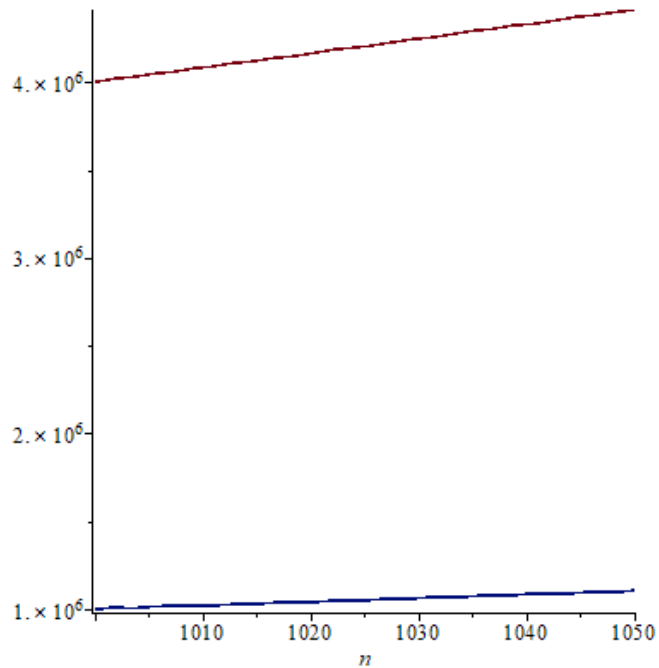
$$4n^2 + 2n \approx n^2$$

$C=1$  $n > 100$



$$100n \ll n^2$$

$$4n^2 + 2nu n^2, \quad 4n^2 + 2nu 4n^2$$



Слабая нижняя оценка (ω)

$$2n^2 = \omega(n) ?$$

$$24n = \omega(n) ?$$

$$\omega$$

$$2n^2 \gg n$$

$$24n \approx n$$

Упражнение

- Постройте графики для сравнения поведения функций при разных константах
- Проанализируйте зависимость аргумента от константы c

Свойства асимптотик (записать и доказать)

- 1. Рефлексивность?
- 2. Симметричность? Перестановочная симметричность?
- 3. Транзитивность?
- 4. Асимптотические оценки результатов операций (умножение на Const, сложение, умножение)

Асимптотическое сравнение функций (по аналогии с $<, >$ для чисел)

1.2.3

- **Базовые функции, используемые при асимптотическом анализе алгоритмов**

- 1. Ближайшие целые
- 2. Деление с остатком
- 3. Многочлены->Степенная функция
- 4. Показательная функция
- 5. Факториал
- 6. Логарифмическая функция
- 7. Функциональная итерация
- 8. Итерированный логарифм

1.2.4

Оценка трудоемкости алгоритма с примерами

Трудоёмкость алгоритма – это функция от размерности задачи, которая оценивает сверху время, требуемое для ее решения

$$f(n) \rightarrow T(L)$$

Простейшая классификация алгоритмов по трудоемкости

Алгоритм называется **полиномиальным**, если его трудоемкость $T(L) = O(p(L))$, где $p(L)$ – некоторый полином или полиномиально ограниченная функция.

Алгоритм называется **экспоненциальным**, если его трудоемкость $T(L) = \Omega(\exp(L))$, где $\exp(L)$ – некоторая экспоненциальная функция

Вопрос. Можно ли в определениях заменить на Θ ?

Этапы определения трудоемкости

- 1 Определить размерность L задачи
- 2 Разработать алгоритм решения задачи и оценить время (число базовых операций) решения в худшем (среднем) случае. При необходимости учесть весовые коэффициенты для операций
- 3 Выразить трудоемкость $T(L)$ и оценить порядок роста при неограниченном увеличении размерности
- 4 Определить класс сложности алгоритма

Задание 1

- Определить трудоемкость алгоритма последовательного вычисления факториала натурального числа n

Fct:=1

for k to n

do Fct:=Fct*k end do

Fct;

#1-p1

#n –p2

#n –p3

#1 –p4

Размерность задачи	$l = \lceil \log_2 n \rceil \quad (2^{l-1} < n \leq 2^l)$
Число арифметических операций	$c_1 n + c_2$
Трудоемкость алгоритма	$T(l) \geq c_3 \cdot n \geq \frac{c_2}{2} \cdot 2^l, (n \geq 2)$ <p>$T(l) = \Omega(2^l)$ – алгоритм экспоненциальный</p>

Задание 2

- Определить трудоемкость алгоритма последовательного вычисления суммы натуральных чисел: $s = a_1 + a_2 + \dots + a_n$

Размерность задачи	$c_1 \cdot n \leq l \leq c_2 \cdot n \quad \left(\frac{1}{c_2} \cdot l \leq n \leq \frac{1}{c_1} \cdot l \right)$
Число арифметических операций	$c_3 n + c_4$
Трудоемкость алгоритма	$T(l) \leq c \cdot n \leq \frac{c}{c_1} \cdot l$ $T(l) = O(l)$ – алгоритм полиномиальный

Задание 3

- Определить трудоемкость алгоритма

проверки числа на простоту:

```
for  $i := 2$  to  $\sqrt{n}$  do  
    if  $n \bmod i = 0$  then exit;
```

Размерность задачи	$l = \lceil \log_2 n \rceil \quad (2^{l-1} < n \leq 2^l)$
Число арифметических операций	$c_1 \sqrt{n} + c_2$
Трудоемкость алгоритма	$T(l) \geq c_3 \sqrt{n} > c_3 (2^{l-1})^{1/2}$ $T(l) = \Omega(2^{l/2})$ – алгоритм экспоненциальный

Задание 4

- Определить трудоемкость алгоритма последовательного вычисления суммы первых n натуральных чисел.

$S := 0;$

for $i := 1$ to n do

$S := S + i;$

ЭКСПОНЕНЦИАЛЬНЫЙ

$$S := \frac{n(n+1)}{2}$$

$O(1)$ – КОНСТАНТНАЯ СЛОЖНОСТЬ