ВЫСШАЯ ШКОЛА ЭКОНОМИКИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Построение и анализ алгоритмов

Семинар 6

Анализ трудоемкости алгоритмов

Анализ трудоемкости алгоритма

- В основе сравнительного анализа алгоритмов лежат как теоретические, так и экспериментальные оценки требуемой памяти и числа базовых операций, задаваемых алгоритмом в выбранной модели вычислений — функции трудоемкости для лучшего, среднего и худшего случаев при фиксированной длине входа.

- Результаты теоретического анализа трудоёмкости алгоритма должны быть подтверждены экспериментальным исследованием.

Модель вычислений

Модель вычислений представляет собой совокупность

- информационной алгебры,
- механизма реализации и
- априорного набора базовых операций, в терминах которых и формулируется алгоритм решения задачи

Машина Поста, машина Тьюринга, нормальные алгорифмы Маркова, могут быть представлены в виде модели вычислений.

Модель вычислений

Специальные модели вычислений для оценки сложности алгоритмов

Алгебраические деревья вычислений

Информационные графы

Машина с произвольным доступом к памяти

Объектный и алгоритмический базисы

Модель вычислений

Объектным базисом будем называть тройку, состоящую из множества объектов, процессора и множества выполняемых им базовых операций:

$$B_R = \{\Sigma, R, C\}$$

Алгоритмическим базисом для заданного объектного базиса *Bp* будем называть двойку, включающую в себя конечное множество обозначений базовых операций *S* и конечное множество обозначений алгоритмических конструкций *T*:

$$B_A = \{S, T\}.$$

Теоретический анализ трудоемкости алгоритма

1) Алгоритм должен правильно решать поставленную задачу
2) При теоретическом анализе алгоритма определяется сложность по времени и сложность по памяти
3) Сложность по времени - примерное количество операций (вычислительная сложность алгоритма)
4) Важна зависимость количества операций алгоритма от размера входных данных. Алгоритмы решения одной и той же задачи сравнивают по скорости роста количества операций
5) Анализ алгоритма можно проводить с использованием
    1) машины Тьюринга (трудоемко).
    2) Записи алгоритма на языке высокого уровня

Теоретический анализ трудоемкости алгоритма

1) Выбираются базовые операции. Например,
   - сравнения,
   - арифметические операции,
   - обращение к элементу массива по индексу
   - Присваивание
   - Логические операции
   - Взятие содержимого по адресу
2) Не различаются арифметические операции :
   - аддитивные, мультипликативные;
   - над целыми || вещественными

Алгоритм Эвклида, реализованный вычитаниями и взятием остатка целочисленного деления выполняется с разной скоростью, т.к. вторая операция занимает больше тактов CPU.

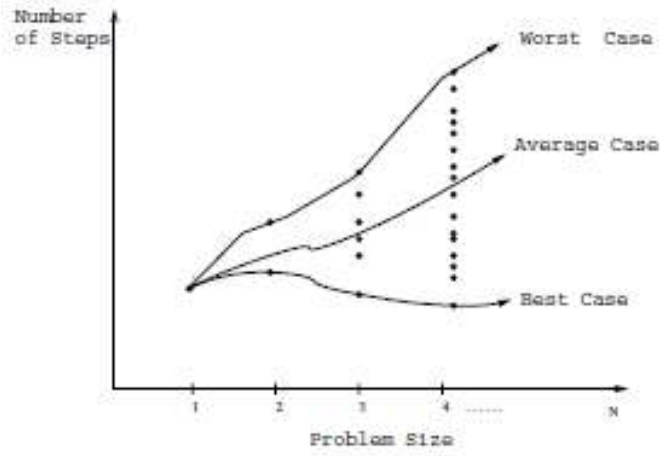
Маленькое задание: Реализовать два варианта алгоритма Эвклида для всех комбинаций a,b из интервала [1;9999], определить среднее количество тактов.

Теоретический анализ трудоемкости алгоритма

Входные множества разбивают на классы в зависимости от поведения алгоритма на каждом множестве

Ищем входы, обеспечивающие выполнение алгоритма

- самое быстрое – наилучший случай,
- самое медленное – наихудший случай,
- средний случай



Оценки в лучшем, среднем и худшем случаях

Теоретический анализ трудоемкости алгоритма

Для среднего случая определяются

- *m* групп входных данных (сложность для входов одной группы должна быть одинаковая)
- Вероятности p_i принадлежности входа группе *i*

$$A(n) = \sum_{i=1}^{m} p_i t_i$$

Полезные формулы 1

$$\sum_{i=L}^{N} i = \sum_{i=0}^{N-L} (i+L) = \sum_{i=0}^{N} i - \sum_{i=0}^{L-1} i$$

$$\sum_{i=0}^{N} N - i = \sum_{i=0}^{N} i$$

$$\sum_{i=1}^{N} C = CN$$

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\sum_{i=1}^{N} \frac{1}{i} \approx \ln N$$

$$\sum_{i=1}^{N} \log_2 i \approx N \log_2 N - 1,5$$

Полезные формулы 2

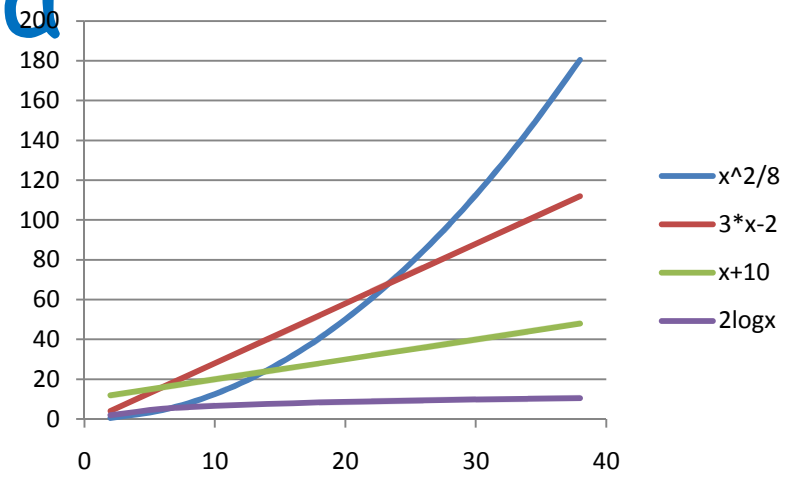
$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} = \frac{2N^3+3N^2+N}{6}$$

$$\sum_{i=0}^{N} 2^i = 2^{N+1} - 1$$

$$\sum_{i=1}^{N} A^i = \frac{A^{N+1}-1}{A-1}$$

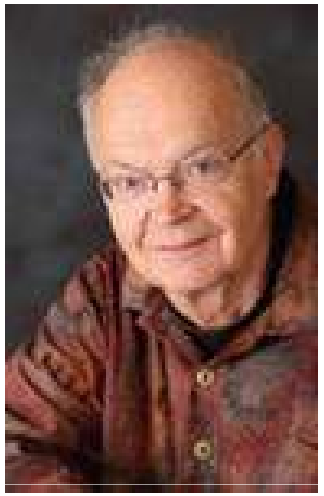
$$\sum_{i=1}^{N} i2^i = (N-1)2^{N-1} + 2$$

Скорости роста



n	$\log_2 n$	n	n log2 n	n^2	n^3	2^n
1	0,0	1	0,0	1	1	2
2	1,0	2	2,0	4	8	4
5	2,3	5	11,6	25	125	32
10	3,3	10	33,2	100	1000	1024
15	3,9	15	58,6	225	3375	32768
20	4,3	20	86,4	400	8000	1048576
30	4,9	30	147,2	900	27000	1073741824
40	5,3	40	212,9	1600	64000	1099511627776
50	5,6	50	282,2	2500	125000	1125899906842620
60	5,9	60	354,4	3600	216000	1152921504606850000
70	6,1	70	429,0	4900	343000	1180591620717410000000

Основные сложностные классы задач



- Середина 1960-х – начало 1970-х - теория сложности алгоритмов и вычислений (Д. Кнут)

- Основной задачей теории сложности вычислений при анализе того или иного алгоритма решения задачи является получение асимптотических верхних оценок временной и емкостной сложности алгоритмов и оценок в среднем.

Основные сложностные классы задач

Обозначения в асимптотическом анализе функций.

- При анализе сложности алгоритмов и в теории ресурсной эффективности используются принятые в математике асимптотические обозначения, позволяющие показать главный порядок функции, маскируя при этом конкретные коэффициенты и аддитивные компоненты неглавного порядка.
- Поскольку число объектов на входе алгоритма, количество учитываемых операций и объем требуемой памяти - положительные числа, асимптотические обозначения будут введены в предположении, что функции $f(n)$ и $g(n)$ есть функции положительного целочисленного аргумента $n>=1$, имеющие положительные значения.

Оценка Θ (тета)

Функция $f(n) = \Theta(g(n))$, если:

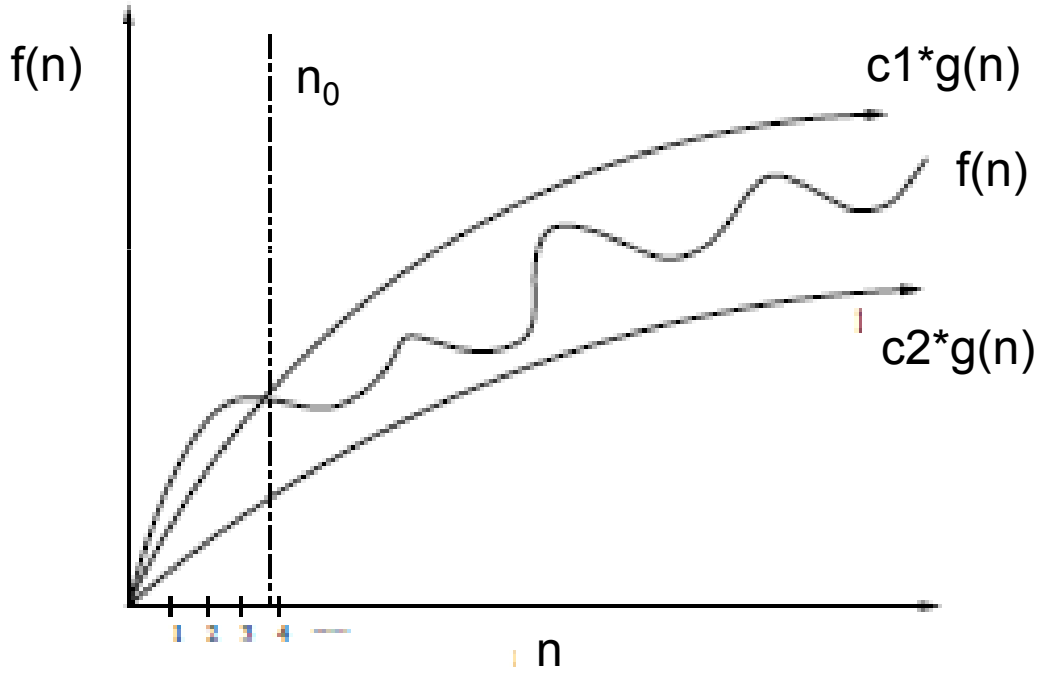
$$\exists\, c_1 > 0, c_2 > 0, n_0 > 0 : \forall n > n_0 \quad c_1 g(n) \le f(n) \le c_2 g(n). \qquad (1)$$

Функция $g(n)$ является асимптотически точной оценкой функции $f(n)$, т. к. в силу (1) функция $f(n)$ отличается от функции $g(n)$ на положительный ограниченный множитель при всех значениях аргумента $n > n_0$.

Запись $f(n) = \Theta(1)$ означает, что функция $f(n)$ или равна константе, не равной нулю, или ограничена двумя положительными константами при любых значениях аргумента $n > n_0$.

Обозначение $\Theta(g(n))$ есть обозначение класса функций, каждая из которых удовлетворяет условию (1).

Оценка Θ



Оценка O (O большое)

Оценка O требует, чтобы функция $f(n)$ не превышала значения функции $g(n)$ при $n > n_0$ с точностью до положительного постоянного множителя, а именно: $f(n) = O(g(n))$, если:

$$\exists\, c > 0, n_0 > 0 : \forall n > n_0 \quad 0 \le f(n) \le cg(n). \qquad (2)$$

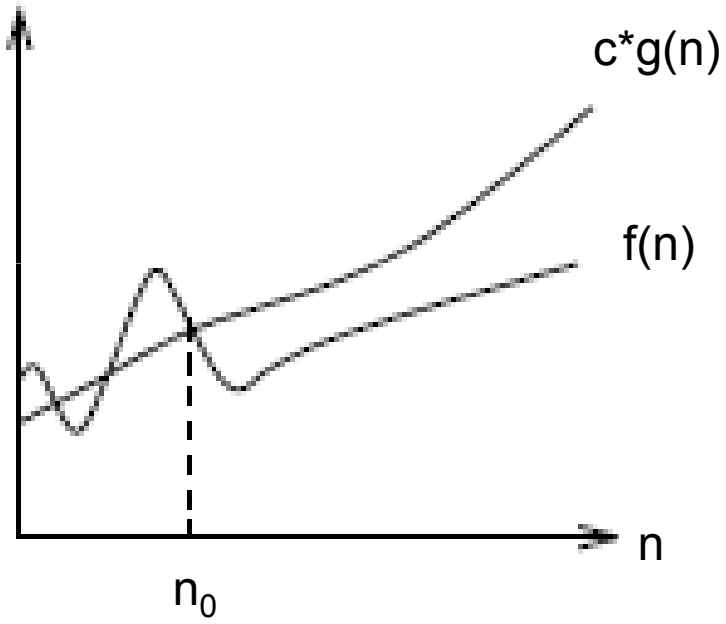
$O(g(n))$ обозначает класс функций, таких, что все они растут не быстрее, чем функция $g(n)$ с точностью до положительного постоянного множителя. Говорят, что функция $g(n)$ мажорирует функцию $f(n)$. Например, для всех функций:

$$f_1(n) = 12,\ f_2(n) = 5n + 23,\ f_3(n) = n\ln n,\ f_4(n) = 7n^2 + 12n - 34$$

будет справедлива оценка $O(n^2)$.

Следует указывать наиболее «близкую» мажорирующую функцию, поскольку, например, для функции $f(n) = 12n^3$ справедлива оценка $O(2^n)$, однако практически она будет мало пригодна.

Оценка O



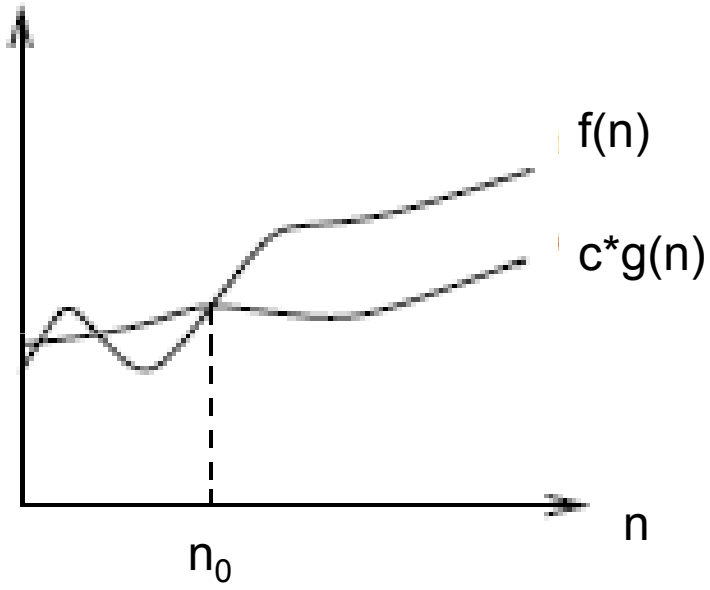
Оценка Ω (омега)

Оценка Ω является оценкой снизу — т. е. определяет класс функций, которые растут не медленнее, чем функция $g(n)$ с точностью до положительного постоянного множителя: $f(n)=\Omega(g(n))$, если:

$$\exists c>0, n_0>0 : \forall n>n_0 \quad 0 \le cg(n) \le f(n). \qquad (3)$$

Например, запись $\Omega(n\ln n)$ обозначает класс функций, которые растут не медленнее, чем $g(n)=n\ln n$, в этот класс попадают, например, все полиномы со степенью больше единицы.

Оценка Ω



Цели исследования

— экспериментальное подтверждение полученной теоретически методами теории ресурсной эффективности функции трудоёмкости алгоритма в среднем;

— получение экспериментальной зависимости трудоёмкости в среднем от длины входа (в т.ч. для определения методами регрессионного анализа оценок коэффициентов такой функциональной зависимости, если на этапе теоретического исследования были получены только асимптотические оценки)

Задачи исследования

— экспериментальное исследование алгоритма при фиксированной длине входа для построения гистограммы относительных частот трудоёмкости, определения доверительной трудоёмкости и количественных мер информационной чувствительности;

— экспериментальное подтверждение теоретически полученной функции трудоемкости в среднем, с учетом принадлежности алгоритма к одному из классов по влиянию характеристических особенностей исходных данных на трудоемкость алгоритма в рамках принятых допущений о частотной встречаемости входов ;

— экспериментальное подтверждение теоретически полученной функции емкостной сложности на исследуемом диапазоне длин входа или построение такой функции на основе данных эксперимента методами регрессионного анализа;

— исследование времени выполнения программной реализации алгоритма в среде выбранного языка, компилятора, операционной системы и аппаратных средств, в целях последующего прогнозирования временной эффективности в диапазоне длин входов, определяемом особенностями применения алгоритма в разрабатываемой программной системе. Отметим, что эта задача может быть решена только на основе экспериментального исследования.

Этапы исследования

— ***модификация исходного текста программной*** реализации алгоритма (расстановка счетчика базовых операций) для определения конкретных значений трудоёмкости — количества выполненных базовых операций при данном входе;

— ***генерация входов алгоритма***, обеспечивающей репрезентативность выборки, т. е. входов, соответствующих особенностям применения исследуемого алгоритма в данной программной системе; значения трудоемкости алгоритма для входов из этого множества и составляют в данном случае генеральную совокупность;

— ***планирование эксперимента***, состоящее в определении необходимого (минимального) объема выборки для фиксированной длины входа, определение границ исследуемого сегмента длин и шага изменения длины входа;

— ***проведение вычислительного эксперимента***, в ходе которого могут быть получены как экспериментальные значения функции трудоемкости, так и экспериментальные времена выполнения; дополнительная задача этого этапа состоит в получении значений функции объёма памяти, если алгоритм обладает значимой ёмкостной чувствительностью;

— ***обработка результатов эксперимента*** — получение методами математической статистики выборочного среднего, выборочной дисперсии, выборочного коэффициента вариации и статистического распределения выборки, для исследуемого множества длин входа.

Планирование эксперимента

— определение шага и границ области исследуемых длин входов;

— определение минимального объема выборки — количества экспериментов с программной реализацией алгоритма при фиксированной длине входа решаемой задачи.

Генерация входов

Основная задача—обеспечение репрезентативности выборки, т. е. генерации таких входов, которые соответствуют особенностям применения данного алгоритма и задачам эксперимента

— обеспечение соответствия типов данных

— обеспечение соответствия ограничениям задачи (например, ограничения на знаки чисел, диапазоны их изменения, алфавиты символов для входных строк и т. д., т.е. область допустимых входов данного алгоритма, в которой гарантируется получение правильных решений);

— обеспечение соответствия области применения. Два подхода:

- случайная выборка из множества известных реальных входов;
- генерация соответствующих входных наборов данных

— учет особенностей алгоритма и задачи эксперимента, например, при генерации входов для алгоритма сортировки методом индексов необходимо ограничивать максимальное значение массива в соответствии с доступным объёмом оперативной памяти.

Расстановка счетчика базовых операций

Правила включение счетчиков для каждой базовой операции принятой модели вычислений:

— для ***ветвления*** строка счетчика, учитывающая базовые операции в сравнении, должна предшествовать сравнению, иначе должна быть включена в оба блока ветвления;

// наращивание счетчика на количество базовых операций в условии

If <условие> then {}; [else{};]

— операции инициализации ***цикла*** должны быть учтены до входа в цикл, операции организации цикла на один проход — внутри тела цикла;

c ← c + 4 *(1 операция инициализации и 3 операции выхода из цикла)*

For *i ← 2* ***to*** *n*

c ← c + 3 *(3 операции на проход: i← i+1, if i≤n)*

Расстановка счетчика базовых операций

Правила включение счетчиков для каждой базовой операции принятой модели вычислений:

— для ***циклов по условию*** необходимо учитывать отдельно трудоёмкость операции сравнения, приводящей к выходу из цикла, в связи с чем строку приращения целесообразно размещать после оператора завершения цикла;

```
c ← c + 2                    (2 операции: j ← i-1)
j ← i-1
While (j>0) and (A[j]>key)
        c ← c + 4            (4 операции: (j>0) and (A[j]>key))
        …
end while
c ← c + 4  (4 операции:(j>0) and (A[j]>key) выхода из цикла)
```

— базовые операции в строках, не влияющих на организацию вычислительного процесса, могут быть учтены в счетчике перед исполняемой строкой.

```
c ← c + 4                    (4 операции: A[j+1] ← A[j])
A[j+1] ← A[j]
c ← c + 2                    (2 операции: j ← j-1)
j ← j — 1
```

Пример – сортировка методом прямого включения (простой вставки)

```
Sort_Ins (A, n)
    c ← 0                    (инициализация счетчика)
    c ← c + 4                (1 операция инициализации и 3 операции выхода из цикла)
    For i ← 2 to n
        c ← c + 3                (3 операции на проход: i← i+1, if i≤ n )
        c ← c + 2                (2 операции: key ← A[i])
        key ← A[i]
        c ← c + 2                (2 операции: j ← i-1)
        j ← i-1
        While (j>0) and (A[j]>key)
            c ← c + 4                (4 операции: (j>0) and (A[j]>key))
            c ← c + 4                (4 операции: A[j+1] ← A[j])
            A[j+1] ← A[j]
            c ← c + 2                (2 операции: j ← j-1)
            j ← j — 1
        end while
        c ← c + 4    (4 операции:(j>0) and (A[j]>key) выхода из цикла)
        c ← c + 3    (3 операции : A[j+1] ← key)
        A[j+1] ← key
    end for
(передача значения c = f_A(D) в программу реализации эксперимента)
End.
```

Оценка памяти и времени выполнения

Модуль оценки динамической памяти

Для оценки объема динамической памяти, требуемой алгоритмом, необходимо получение ***доступа к функциям операционной системы***, осуществляющим выделение областей памяти. Этот модуль должен на основе перехвата обращений программы по выделению памяти к операционной системе определить наибольший объем динамической памяти за всё время выполнения, затребованный программной реализацией алгоритма при решении определенной задачи.

Модуль исследования времени выполнения

Функция модуля — на основе ***средств измерения времени***, обеспечиваемых операционной системой, или на основе обращения к тактовому счетчику, получить время выполнения программой реализации алгоритма на исходных данных, полученных от модуля генерации входов.

Обработка экспериментальных результатов

— первичная обработка результатов эксперимента — получение статистического распределения выборки, выборочного среднего, выборочной дисперсии и выборочного коэффициента вариации по каждому компоненту комплексной оценки качества алгоритма для исследуемого сегмента длин входов;

— подтверждение теоретической функции трудоемкости алгоритма в среднем случае с использованием методов математической статистики;

— подбор функциональных зависимостей по экспериментальным результатам или подбор коэффициентов таких зависимостей на основе функционального вида, указанного пользователем — решение задачи полного или частичного регрессионного анализа. При этом, как правило, если асимптотические оценки функции трудоёмкости получены в ходе теоретического анализа, то задача сводится к определению коэффициентов, обеспечивающих наилучшую аппроксимацию — для этих целей может использоваться, например, метод наименьших квадратов или его специальные модификации;

— прогнозирование времени выполнения программной реализации алгоритмов на основе измерений времени выполнения и функции трудоемкости

И т.д.

Сравнительный анализ алгоритмов

Основная задача— определение областей применения множества исследуемых алгоритмов на основе полученных экспериментальных результатов.

Если получены функциональные зависимости емкостной, временной эффективности, доверительной трудоёмкости и количественных мер информационной чувствительности для различных алгоритмов решения определенной задачи, и на их основе задан комплексный критерий оценки качества алгоритма, то такой сравнительный анализ может быть выполнен, например, с использованием аппарата анализа функций на конечном сегменте.

Результат - рекомендации по выбору рационального алгоритма в зависимости от сегмента длины входа при выбранном комплексном критерии оценки качества.

Полученные результаты могут быть использованы при разработке комбинированных алгоритмов, если на исследуемом сегменте длин входов рациональными оказываются несколько алгоритмов в различных подсегментах этого сегмента.

Домашнее задание

- Составить таблицу асимптотических оценок трудоемкости алгоритмов в лучшем, среднем, худшем случаях
- Расставить счетчики операций в функциях сортировок.
- Провести эксперимент, определить среднее количество операций для разных сортировок, построить графики. При этом для каждого вызова сортировки генерировать новый массив
- Составить таблицы и представить графики экспериментальных оценок алгоритмов