Лабораторная работа

«Анализ алгоритмов»

# 1 Трудоемкость алгоритмов

Под ***алгоритмом*** понимается единый метод решения определенного класса однотипных задач, обладающий свойством дискретности, массовости, определенности, результативности и оперирующий конструктивными объектами.

Под ***размерностью задачи***, которую в дальнейшем будем обозначать через *l*, понимают то количество информации, которое достаточно для формального описания задачи. Мы будем полагать, что размерность задачи – это минимальное количество бит, которого достаточно для описания входных данных задачи. ***Зам***. *Далее предполагаем, что размерность машинного слова достаточна для представления любого числа, и размерность задачи будет ограничена количеством исходных данных в ее формальном описании.*

***Трудоемкость алгоритма*** – это функция от размерности задачи, которая оценивает сверху время, требуемое для решения задачи.

Трудоемкость алгоритма (сложность алгоритма) измеряется двумя параметрами: T(*l*) (временная сложность), S(*l*) (пространственная, емкостная сложность, или требования к памяти).

Учитывая сделанное ранее предположение о том, что для представления любого числа нам достаточно одного машинного слова, получаем, что емкостная сложность алгоритма (размерность памяти) для задания *n* элементов не превосходит *n*.

Если в качестве модели вычислений взять неветвящуюся программу и предположить, что алгоритм – это последовательность арифметических операций и все арифметические операции (аддитивные: +, inc, dec; мультипликативные: \*, /) эквивалентны, т. е. затрачивают на свое выполнение одну единицу времени (хотя на самом деле известно, что мультипликативные операции работают дольше), то трудоемкость (временная сложность) алгоритма определяется как функция с количеством операций, требующимся для решения задачи, выраженным через ее размерность.

Однако, при оценке трудоемкости алгоритма точное значение количества операций не имеет для нас существенного значения, более важна скорость роста этого числа при возрастании объема входных данных, т.е. нас интересуют асимптотические оценки.

Алгоритм называется ***полиномиальным***, если его трудоемкость Т(*l*) = O(*р*(*l*)), где *р*(*l*) – некоторый полином или полиноминально ограниченная функция.

Алгоритм называется ***экспоненциальным***, если его трудоемкость Т(*l*) = Ω(ехр(*l*)), где ехр(*l*) – некоторая экспоненциальная функция.

# 2 Рекуррентные уравнения

Для некоторой задачи под *подзадачей* мы будем понимать ту же задачу, но меньшим числом параметров, *или* задачу с тем же числом параметров, но при этом хотя бы один из параметров имеет меньшее значение.

Соотношения, связывающие одни и те же функции, но с различными аргументами, называются ***рекуррентными уравнениями***.

Рекуррентное уравнение называется ***правильным*** если значения аргументов у любой из функций правой части соотношения меньше значения аргументов любой из функций левой части соотношения; если аргументов несколько, то достаточно уменьшение одного из них.

Правильное рекуррентное уравнение называется ***полным***, если оно определено для всех допустимых значений аргументов. Т(*n*) = Т(*n* -1) + Т(*n* - 2), Т(1)= 1, T(2)=1.

***Методы решений рекуррентных уравнений:***

1. Оценка решения рекуррентного уравнения. Метод подстановок.
2. Метод итераций.
3. Метод рекурсивных деревьев.

# 3 Самостоятельная работа «Асимптотический анализ»

Для выполнения работы рекомендуется ознакомится с главами 1, 2, 3, 4 (п.4.3 – 4.6) [1].

**Вариант 1**:

1. Предположим, на одной и той же машине проводится сравнительный анализ реализаций двух алгоритмов сортировки, работающих по методу вставок и по методу слияния. Для сортировки *n* элементов методом вставок необходимо 8*n*2 шагов, а для сортировки методом слияния – 64*n* lg *n* шагов. При каком значении *n* время сортировки методом вставок превысит время сортировки методом слияния?
2. Расположите следующие функции по порядку в соответствии с нотацией большого О (обоснуйте это расположение). Сгруппируйте, например, с помощью подчеркивания, функции, которые являются Θ оценкой друг другу:
   1. ;
   2. ;
   3. ;
   4. ;
   5. ;
   6. ;
   7. .
3. Оцените скорость роста функций при росте . Результат представьте в виде таблицы.
4. Пусть *f*(*n*) и *g*(*n*) – асимптотически неотрицательные функции. Докажите с помощью базового определения Θ-обозначений, что max (*f*(*n*), *g*(*n*)) = Θ(*f*(*n*)+ *g*(*n*)).
5. Пусть *f*(*n*) и *g*(*n*) – асимптотически положительные функции. Докажите или опровергните справедливость каждого из приведенных ниже утверждений:
   1. Из *f*(*n*) = O (*g*(*n*)) следует, что *g*(*n*) = O (*f*(*n*)).
   2. *f*(*n*) + *g*(*n*) = Θ(min (*f*(*n*), *g*(*n*)).
   3. Из *f*(*n*) = O (*g*(*n*)) следует log (*f*(*n*)) = O (log (*g*(*n*)), где при достаточно больших *n* верны неравенства log (*g*(*n*)) ≥ 1 и *f*(*n*) ≥ 1.
   4. Из *f*(*n*) = O (*g*(*n*)) следует 2*f*(*n*) = O (2*g*(*n*)).
6. Выразите функцию *f*(*n*) = *n*3/1000 - 100 *n*2 - 100 *n* + 3 в Θ обозначениях.
7. Решите следующие рекуррентные уравнения: T(*n*) = T(*n*-1) + *n*-1, *n* > 2, T(1) = 3.
8. Определите верхнюю и нижнюю асимптотические границы функции T (*n*) для каждого из представленных ниже рекуррентных соотношений. Считаем, что T (*n*) – константа при *n* ≤ 2. Обоснуйте свой ответ.
9. T (*n*) = T (9*n*/10)+ *n*;
10. T (*n*) = 6 T (*n*/2)+ *n*2;
11. T (*n*) = 16 T (*n*/4)+ *n*2.

**Вариант 2**:

1. При каком минимальном значении *n* алгоритм, время работы которого определяется формулой 100*n*2, работает быстрее, чем алгоритм, время работы которого выражается как 2*n*, если оба алгоритма выполняются на одной и той же машине?
2. Расположите следующие функции по порядку в соответствии с нотацией большого О (обоснуйте это расположение). Сгруппируйте, например, с помощью подчеркивания, функции, которые являются Θ оценкой друг другу:
   1. 10 *n* log *n*;
   2. 2100;
   3. 200log*n*;
   4. *n* 2*n*;
   5. log(*n*!);
   6. *n*!;
   7. *n*log log *n*.
3. Для каждой функции *f*(*n*) = {log *n*, *n*1/2, *n*, *n* log *n*, *n*2, 2*n*, *n*!} и времени *t* (1 секунда, 1 час, 1 неделя, 1 год, 1 столетие) определите наибольший размер *n* задачи, которая может быть решена за время *t*, при условии, что для ее решения алгоритму необходимо *f*(*n*) микросекунд. Результат представьте в виде таблицы.
4. Покажите, что для любых действительных констант *a* и b, где *a*, *b* > 0, справедливо соотношение (*n*+*a*)*b*= Θ (*nb*).
5. Пусть *f*(*n*) и *g*(*n*) – асимптотически положительные функции. Докажите или опровергните справедливость каждого из приведенных ниже утверждений:
   1. *f*(*n*) = O (*f*(*n*)2);
   2. из *f*(*n*) = O (*g*(*n*)) следует, что *g*(*n*) = Ω (*f*(*n*));
   3. *f*(*n*) = Θ (*f*(*n*/2));
   4. из *f*(*n*) + o(*f*(*n*))= Θ(*f*(*n*)).
6. Выразите функцию *f*(*n*)=3*n*2/1000+10 *n* log*n*+200 *n*+3 в Θ обозначениях.
7. С помощью метода рекурсивных деревьев, докажите, что решение рекуррентного соотношения T (*n*) = T (*n*/3) + T (2*n*/3) + c *n* ведет себя как Ω (*n* log*n*), где с – константа.
8. Определите верхнюю и нижнюю асимптотические границы функции T (*n*) для каждого из представленных ниже рекуррентных соотношений. Считаем, что T (*n*) – константа при достаточно малых *n*. Обоснуйте свой ответ.
   1. T (*n*) = 2 T (*n*/2)+ *n*3;
   2. T (*n*) = 7 T (2*n*/5)+ *n*2;
   3. T (*n*) = 2 T (*n*/4)+ *n*1/2.

# 4 Самостоятельная работа «Разработка алгоритмов»

**Простые задачи**

1. Даны три целых числа A, B, C. Если какая-то перестановка этих чисел соответствует арифметической прогрессии, вывести на экран сообщение «Да», иначе – сообщение «Нет».
2. Даны шесть действительных чисел X, Y, Z, W, V, U в качестве координат трех точек A (X,Y), B( Z,W), C(V,U), образующих треугольник на плоскости. Если точки А, B, C являются вершинами равнобедренного треугольника, то вывести на экран сообщение «Да», иначе – сообщение «Нет».
3. Даны три целых числа A, B, C. Если все числа различные, то заменить меньшее на сумму двух других. Вывести на экран новые значения чисел A, B, C.
4. Даны четыре целых числа A, B, C, D. Если сумма каких-то двух чисел равна сумме двух оставшихся, то вывести на экран сообщение «Да», иначе – сообщение «Нет».
5. Даны шесть действительных чисел X, Y, Z, W, V, U в качестве координат трех точек A (X,Y), B( Z,W), C(V,U), образующих треугольник на плоскости. Вывести на экран угол между высотой и медианой, которые проходят через точку А.
6. Даны четыре целых числа A, B, C, D. Если какие-то три из них образуют арифметическую прогрессию, вывести на экран сообщение «Да», иначе – сообщение «Нет».
7. Рассмотрим область |x|+|y|1 и прямую Ax + By = 1. Даны числа A и B. Если область и прямая имеют общую точку, то вывести на экран сообщение «Да», иначе – сообщение «Нет».
8. Даны три целых числа A, B, C. Вывести на экран произведение двух наибольших чисел или сообщение «Нет», если два максимальных числа равны.
9. Даны три целых числа A, B, C. Вывести на экран среднее арифметическое для четных чисел. Если четных чисел нет, то вывести на экран сообщение «Нет четных чисел».

**Циклы с параметром**

1. Составить программу возведения натурального числа в квадрат, используя следующую закономерность:  
   *12 = 1;  
   22 = 1 + 3;*

*32 = 1 + 3 + 5;  
42 = 1 + 3 + 5 + 7;  
…  
n2 = 1 + 3 + 5 + 7 + … + 2n-1.*

1. Получить все числа Армстронга, состоящие из трех и четырех цифр. (Натуральное число из n цифр является числом Армстронга, если сумма его цифр, возведенных в n-ю степень, равна самому числу).
2. Получить все натуральные числа в заданном диапазоне, которые являются полными квадратами.
3. В последовательности a1, a2, a3, …, an каждый член, начиная с четвертого, равен последней цифре суммы трех предыдущих. Найти n-й элемент последовательности.
4. Определить все трехзначные числа, которые обладают следующим свойством: как само число, так и его перевертыш (т.е. число, записанное теми же цифрами, но в обратном порядке) делятся на все свои цифры.
5. Дана последовательность из 20 целых чисел. Определить количество чисел в наиболее длинной последовательности из подряд идущих нулей.
6. Составить программу, которая для заданного значения х вычисляет по схеме Горнера значение следующего многочлена:
   1. x10 + 2x9 + 3x8 + … + 10x + 1
   2. 11x10 + 10x9 + 9x8 + … + 2x + 1
7. Имеется n бактерий красного цвета. Через 1 такт времени красная бактерия меняется на зеленую, затем через 1 такт времени делится на красную и зеленую. Определить, сколько будет всех бактерий через k тактов времени?
8. Разработать алгоритм определения простоты числа n.
9. Разработать алгоритм вычисления факториала: n!.

# Контрольные вопросы

1. Дайте определение понятия «алгоритм», определите его свойства. Раскройте смысл этих свойств с помощью примеров.
2. Чем отличается современная трактовка понятия алгоритма от значения этого слова в прошлом? Чем можно объяснить историческое изменение значения этого понятия?
3. Какие способы описания алгоритмов существуют.
4. Что означает понятие «правильный алгоритм»?
5. Перечислите этапы разработки программ.
6. Приведите примеры семантических, синтаксических, логических ошибок в программе.
7. Дайте определение понятию «количество информации». Определите какого количества информации достаточно для установления номера выпавшего значения кубика.
8. Дайте определение понятию «размерность задачи».
9. Дайте определение понятию «сложность алгоритмы». С какой целью проводится анализ сложности алгоритма и зачем применяется система сравнительных оценок алгоритмов?
10. Определите понятия асимптотических оценок f(n)=O(g(n)), f(n)=Ω(g(n)), f(n)=Θ(g(n)). С какой целью проводится асимптотический анализ функций трудоѐмкости алгоритмов.
11. Сформулируйте свойства транзитивности, рефлексивности и симметричности асимптотических оценок.
12. Дайте определение понятий «экспоненциальный алгоритм», «полиномиальный алгоритм». Приведите примеры.
13. Объясните понятие «равнодоступная адресная машина». Приведите примеры ее использования.
14. Определите трудоемкость последовательной конструкции «ветвление» и «цикл».
15. Определите трудоемкость конструкции «цикл» со вложенным циклом.
16. Определите трудоемкость конструкции «цикл» с *k* вложенными циклами.
17. Дайте определение понятию «рекуррентное уравнение». Приведите примеры рекуррентных уравнений.
18. Какое рекуррентное уравнение называется правильным? Приведите примеры правильных рекуррентных уравнений.
19. Опишите «метод итераций» для решения рекуррентных уравнений.
20. Опишите метод оценки решения рекуррентного уравнения: «метод подстановок».
21. Опишите «метод рекурсивных деревьев» для решения рекуррентных уравнений.
22. Сформулируйте и докажите основную теорему о решении рекуррентного уравнения.

# Литература

1. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – Москва: Вильямс, 2013. – 1328 с. [PDF (167 Мб)](https://yadi.sk/i/IMPynFi0WSOqSw) [DjVu (18 Мб)](https://yadi.sk/i/luHQRgdK2uHWhw)

# Требования к разрабатываемым приложениям

1. Задания, в которых требуется разработать приложение или реализовать алгоритм, необходимо разрабатывать на объектно-ориентированном языке программирования, следуя принципам объектно-ориентированного проектирования.
2. Обосновывайте выбор структур данных и оценивайте сложность алгоритма. Разрабатывайте наиболее эффективный алгоритм.
3. Исходные данные могут быть произвольных типов, поэтому, при разработке приложений используйте параметризованные классы.
4. Используйте систему контроля версий. Приветствуется совместная работа над проектом командой из двух человек.
5. Для тестирования приложения, подготовьте набор разнообразных входных данных, покрывающих все множество возможных входных данных. Для генерации случайных правдоподобных входных данных разработайте утилиту (класс + методы).
6. При работе с программой пользователь должен видеть условие задачи, требования к входным данным, информативные подсказки, в случае некорректного ввода.
7. Разработанная программа должна обеспечивать возможность работы в двух режимах:
   1. ручной ввод данных – выполнять алгоритм для одного набора входных данных (в этом случае ввод данных и вывод результата осуществляется в полях формы или консоль);
   2. ввод данных из файла, вывод результата – в файл.

# Основные требования к оформлению кода:

1. Программа, написанная на языке java должна соответствовать «java code conventions». Программы на других языках должны следовать этим требования, насколько это возможно.
2. Каждая команда должна быть представлена на отдельной строке. Любое объявление свойства или переменной также должно быть выполнено на отдельной строке.
3. Структура программы должна быть ярко выражена с помощью использования необходимого количества отступов в строке (от 2х до 4х), для отражения структурной вложенности языковых конструкций.
4. Код программ необходимо сопровождать комментариями, в количестве, достаточном для быстрого понимания алгоритма.
5. Счетчики в циклах традиционно называют i, j, k, l, m, n.
6. Вид доступа к члену класса всегда должен быть явно указан.
7. Смысл названий классов, свойств, методов, переменных должны быть понятен при чтении (самодокументирующийся код).