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.

# Требования к разрабатываемым приложениям

1. Для тестирования приложения, подготовьте **набор разнообразных входных данных**, покрывающих все множество возможных входных данных. Для генерации случайных правдоподобных входных данных разработайте утилиту (класс + методы).
2. При работе с программой пользователь должен видеть условие задачи, требования к входным данным, информативные подсказки, в случае некорректного ввода.
3. **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.
   3. **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?**
   4. **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?**