1) Алгоритм: определение, свойства, примеры

2) Сложность алгоритма на конкретном входе

3) RAM-машина

4) Сложность в худшем случае

5) Асимптотические величины: О-большое, Омега-большое, Тетта-большое 6) Сложность в среднем случае

7) Комбинаторные величины

8) Оценка средней сложности алгоритма для конечного числа вариантов

9) Оценка средней сложности алгоритма для бесконечного числа вариантов

10) Метод динамического программирования для решения задач: общий принцип, условия применения

11) Метод динамического программирования для решения задач: стратегия разработки решения

12) Жадные алгоритмы решения задач: общий принцип, условия применения

13) Жадные алгоритмы решения задач: стратегия разработки решения

14) Амортизационный анализ. Метод группового анализа

15) Амортизационный анализ. Метод бухгалтерского учета

16) Амортизационный анализ. Метод потенциалов.

17) Амортизационный анализ. Сравнение методов.

18) Инвариант цикла. Определение, применение.

**Ответы:**

**1 вопрос:**

**Алгори́тм** — набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата.

Основными ***свойствами*** алгоритма являются:

1. детерминированность (определенность). Предполагает получение однозначного результата вычислительного процecca при заданных исходных данных. Благодаря этому свойству процесс выполнения алгоритма носит механический характер;
2. результативность. Указывает на наличие таких исходных данных, для которых реализуемый по заданному алгоритму вычислительный процесс должен через конечное число шагов остановиться и выдать искомый результат;
3. массовость. Это свойство предполагает, что алгоритм должен быть пригоден для решения всех задач данного типа;
4. дискретность. Означает расчлененность определяемого алгоритмом вычислительного процесса на отдельные этапы, возможность выполнения которых исполнителем (компьютером) не вызывает сомнений.

Выделяют три основные схемы:

1)Линейный алгоритм.

2)Ветвящийся алгоритм, или разветвлённый.

3)Циклический.

Возьмем простой пример. Есть ряд чисел от 1 до 100. Нам необходимо найти все простые числа, то есть те, которые делятся на единицу и себя**.**

1. Берем первое число.

2. Проверяем, меньше ли оно 100.

3. Если да, проверяем простое ли это число.

4. Если условие выполняется, записываем его.

Минимальное число в последовательности.

Берём первое число в последовательности за минимальное.

1.Берём второе число в последовательности.

2.Если число меньше минимального, то берём его за минимальное.

Записываем минимальное число.

**Вопрос 2:**

Сложность алгоритма на конкретном входе — это базовое понятие в теории анализа сложности алгоритмов, выражающее число рассматриваемых операций, которое совершает алгоритм, обрабатывая конкретный вход. Сложность на конкретном входе — это всегда число.

**Вопрос 3:**

RAM-машина

Модель машины с одним сумматором, команды программы не могут изменять сами себя. Служит теоретической моделью, в частности, для анализа алгоритмов.

RAM-машина состоит из:

* входной ленты, с которой она может только считывать
* выходной ленты, на которую она может только записывать
* памяти.

Входная лента состоит из последовательности ячеек, в которых записаны целые числа. Каждый раз, когда машина считывает число с входной ленты, головка передвигается на следующую ячейку вправо.

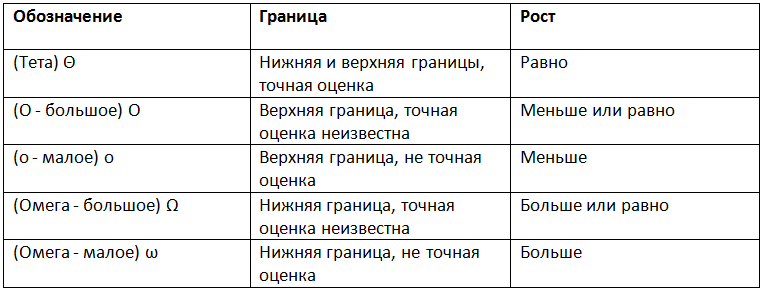
**Вопрос 4:**

Сложность в худшем — это функция, зависящая от нормы входа (меры линейного размера входа). Сложность в худшем случае — это максимум затрат, которые можно ожидать от данного алгоритма, если перебрать все возможные входы.

Сложность в худшем определяется как функция

|  |  |
| --- | --- |
| , |  |

**Вопрос 5:**



1. (О — большое) — верхняя граница, в то время как (Омега — большое) — нижняя граница. Тета требует как (О — большое), так и (Омега — большое), поэтому она является точной оценкой (она должна быть ограничена как сверху, так и снизу). К примеру, алгоритм требующий Ω (n logn) требует не менее n logn времени, но верхняя граница не известна. Алгоритм требующий Θ (n logn) предпочтительнее потому, что он требует не менее n logn (Ω (n logn)) и не более чем n logn (O(n logn)).
2. f(x)=Θ(g(n)) означает, что f растет так же как и g когда n стремится к бесконечности. Другими словами, скорость роста f(x) асимптотически пропорциональна скорости роста g(n).
3. f(x)=O(g(n)). Здесь темпы роста не быстрее, чем g (n). O большое является наиболее полезной, поскольку представляет наихудший случай.

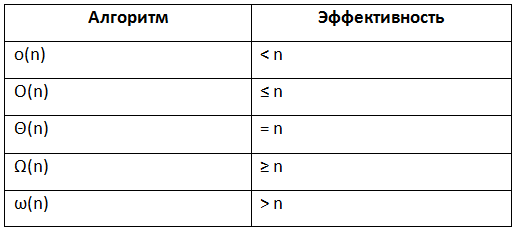
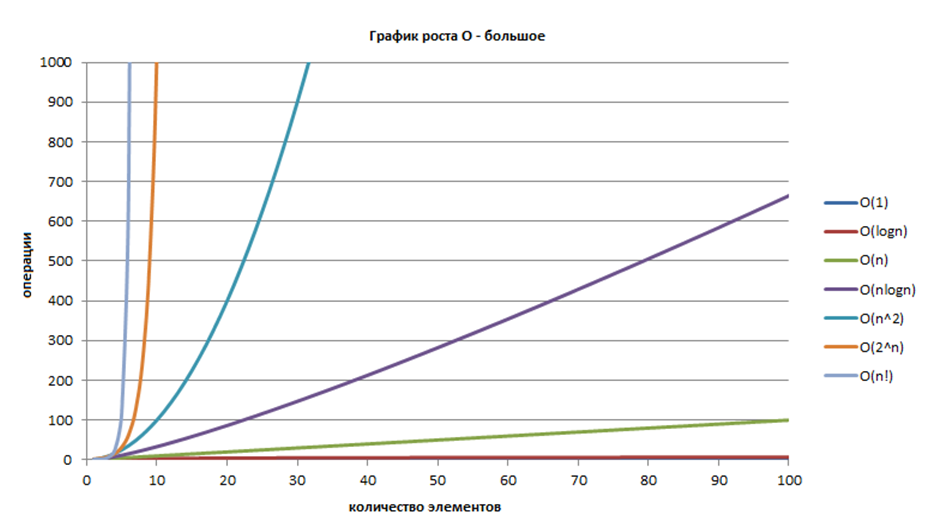
Короче говоря, если алгоритм имеет сложность \_\_ тогда его эффективность \_\_  


График роста O — большое



**Вопрос 6:**

Сложность алгоритма в среднем — это количество неких вычислительных ресурсов (обычно — время), требуемое для работы алгоритма, усреднённое по всем возможным входным данным. Понятие часто противопоставляется сложности в худшем случае , где рассматривается максимальная сложность алгоритма по всем входным данным.

2. Как вычислить сложность в среднем случае?

Понятие сложности в среднем случае вычисляется по следующей формуле:

𝑇¯(𝑁)=∑𝑃|𝑥|=𝑁(𝑋)𝐶𝑇(𝑥),

3. Какие выводы можно сделать с помощью сложности в среднем случае для сравнения двух алгоритмов?

Из полученных результатов можно только сделать вывод о том, какой именно алгоритм лучше решает данную задачу на данном входе в среднем случае.

4. Как вычислять сложность рекурсивных алгоритмов?

Если задача имеет рекурсивное решение и представима в виде: 𝑇̅(𝑛)= 𝑎𝑇̅(𝑛/𝑏)+𝑓(𝑛)

Здесь f(n) – суммарная сложность разбиения задачи на подзадачи и склеивания полученных ответов, то

1) Если 𝑓(𝑛)∈𝑂(𝑛log𝑏𝑎−𝜀) для некоторого положительного 𝜀, то 𝑇(𝑛)∈𝑂(𝑛log𝑏𝑎)

2) Если 𝑓(𝑛)∈𝜃(𝑛log𝑏𝑎), то 𝑇(𝑛)∈𝜃(𝑛log𝑏𝑎log𝑛)

3)Если 𝑓(𝑛)∈𝛺(𝑛log𝑏𝑎−𝜀), то 𝑇(𝑛)∈𝜃(𝑓(𝑛))

**Вопрос 7:**

Комбинаторные величины

Конечные множества, на элементы которых могут накладываться определённые ограничения, такие как: различимость или неразличимость элементов, возможность повторения одинаковых элементов и т. п.

**Вопрос 8:**

**Вопрос 9:**

**Вопрос 10:**

**Вопрос 11:**

**Вопрос 12:**

**Вопрос 13:**

**Вопрос 14:**

**Вопрос 15:**

**Вопрос 16:**

**Вопрос 17:**

**Вопрос 18:**

Инвариант цикла — логическое выражение, истинное после каждого прохода тела цикла (после выполнения фиксированного оператора) и перед началом выполнения цикла, зависящее от переменных, изменяющихся в теле цикла. Инварианты используются для доказательства правильности результата, полученного циклическим алгоритмом.