

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

13 лекций (четверг, 2-4 нед, 10-35, ауд. 04-4)

13 практических занятий

Контрольная работа

ЭКЗАМЕН!

НЕТ – «автоматам»!

- **ДА – облегченному формату аттестации, если:**
 1. Выполнение всех заданий с отметками ≥ 7
 2. Системная и постоянная работа по вопросам учебной программы в аудитории и дома
 3. Изучение дополнительной литературы
 4. Пропуски занятий \rightarrow MIN

ВВОДНЫЕ ЗАМЕЧАНИЯ

Фундаментальные понятия

- Алгоритмы
- Структуры данных

Связь алгоритмов с представлением данных

- Численные методы



- Арифметические операции
- Достаточно простые структуры данных

- Другие методы



- Хранение и обработка информации
- Структуры данных со сложной иерархией

Никлаус Вирт

Алгоритмы и структуры данных

- «Сутью искусства программирования обычно считается
- умение составлять операции.
- Однако ... не менее важно
- умение составлять данные »

Учебные задачи

1. Выяснение влияния алгоритма на эффективность работы программы
(время (скорость)+дополнительная память)
Точное и приближенное решение – баланс?
2. Освоение элементарных способов оценки сложности алгоритмов
3. Выявление связи между эффективностью алгоритмов и представлением данных

План учебной программы

- 1. Математические основы анализа алгоритмов (детерминированных)
- 2. Анализ базовых алгоритмов поиска и сортировки
- 3. Структуры данных (абстракции)
- 4. Поиск

Литература

- 1. Ахо, А. ... Структуры данных и алгоритмы. – М. : Вильямс, 2003. – 384 с.
- 2. **Кормен, Т.... Алгоритмы: построение и анализ** М. : Вильямс, 2007. – 1296 с. (Кормен Алгоритмы Вводный курс, 2014 , – 208с)
- 3. Кнут, Д. Э. Искусство программирования. Том 1-3. Сортировка и поиск – М. : Вильямс, 2011. – 824 с.
- 4. Седжвик, Р. Алгоритмы на C++. – М. : Вильямс, 2011. – 1056 с.
- 5. Скиена Рук-во по разработке алгоритмов. -2011, ок. 700 с.(есть презентации)
- 6. Д.Макконелл Основы современных алгоритмов 2004
- 7. Н. Вирт АлгД (прим. На Обероне) -2016. ок. 350 с.
- 8. Subero, A. Codeless DS & A –Springer. – 2020. –143 p.(совсем упрощенное изложение!)
- 9. Storer, J. An introduction to DS&A –Springer. – 2002+ переиздания –609 p.
- 10. Knebl, H. A&DS Foundations and probabilistic methods for design and analysis - –Springer. – 2020 – 356 p.

Глава 1

МАТЕМАТИЧЕСКИЕ ОСНОВЫ АНАЛИЗА АЛГОРИТМОВ

1.1 БАЗОВЫЕ ПОНЯТИЯ И АЛГОРИТМИЧЕСКИЕ СТРАТЕГИИ

Структура данных

- Способ хранения и организации данных, облегчающий к ним доступ и модификацию

Понятие алгоритма (Д. Э. Кнут)

«Алгоритм – это не просто конечный набор правил, который определяет последовательность выполнения операций для решения задачи определенного типа.

Он обладает пятью важными чертами:

- конечность,
- детерминированность,
- ввод,
- вывод,
- эффективность»

Еще одно описание понятия *вычислительный алгоритм*

Алгоритм – это заданное на некотором языке
конечное предписание,

задающее конечный процесс выполнимых
элементарных операций для получения
корректного результата решения задачи,

общее для класса возможных исходных
данных

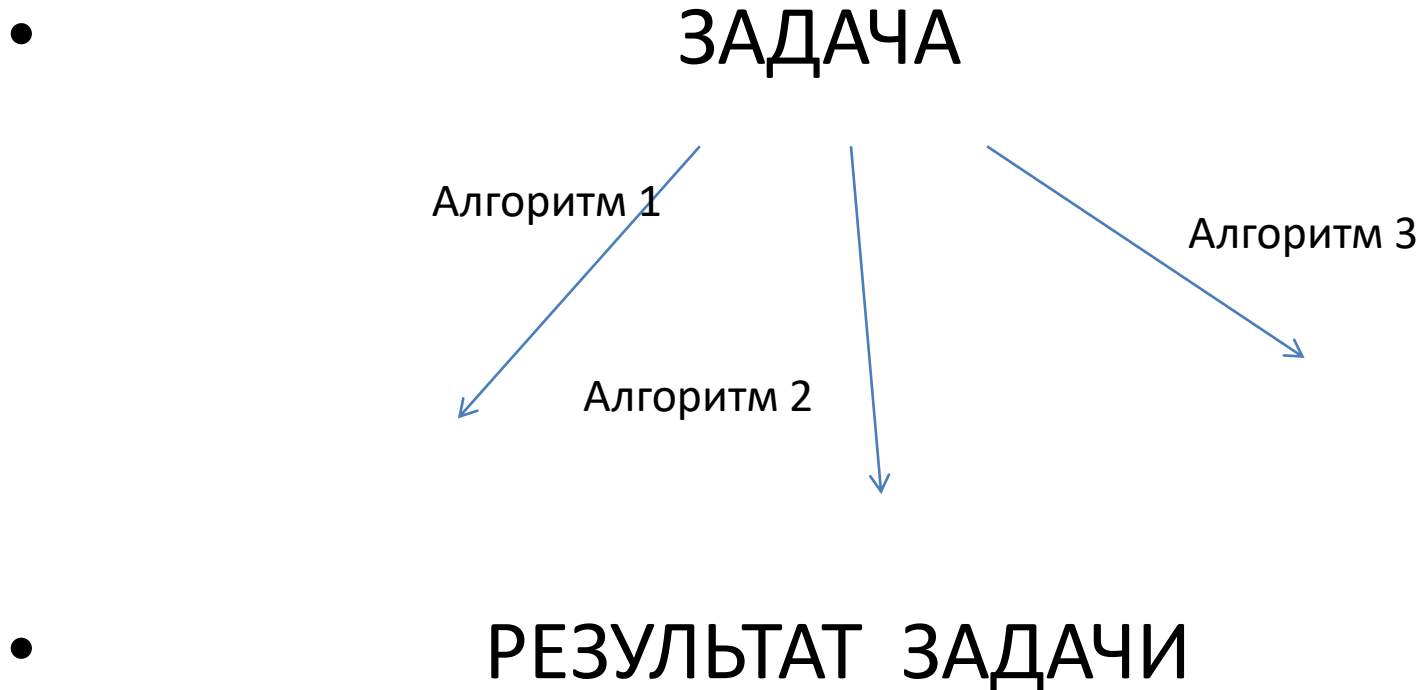
$$f : D(input) \rightarrow E(output)$$

- Вычислительный алгоритм – метод, доведенный до степени детализации, позволяющей получить его программную реализацию на ЯП



1. Абстрактный вычислительный алгоритм
2. Программа на разных ЯП

Корректные алгоритмы для одной и той же задачи



Свойства корректного алгоритма

(Кнут – трассировка)

- 1. Позволяет после КОНЕЧНОГО числа элементарных операций преобразовать любые допустимые наборы данных в результат
- 2. Устойчив к малым возмущениям входных данных
- 3. Результат обладает вычислительной устойчивостью

Некорректный алгоритм для НЕКОТОРЫХ входных данных может:

- не завершить работу (проблема останова!)
- выдать неправильный (неожиданный!) ответ

Но!

Некоторые некорректные алгоритмы
оказываются полезными, если удастся
контролировать частоту появления ошибки

Примеры некорректных алгоритмов

- 1. Деление чисел «углом» без определения критерия останова
- 2. Вычисление нулей квадратного многочлена без реализованной операции нахождения квадратных корней из отрицательного числа
- 3. Определение, является ли натуральное число простым (скорость или правильность ответа?)
- Решето Эратосфена, небольшая оптимизация – точно и медленно
- Приближенно и быстро с одной ошибкой в 2^{50} случаях


Сложность алгоритма

- 1. Комбинационная сложность – минимальное число конструктивных элементов для реализации алгоритма в виде схемы вычислительного устройства
- 2. Описательная сложность – длина описания алгоритма на некотором ЯП
- 3. **Вычислительная сложность** – количество элементарных операций, исполняемых алгоритмом для каких-то входных данных

Основные характеристики вычислительной сложности корректного алгоритма

1. *Временная сложность* (ключевая роль!)

Приблизительное число операций 

Зависимость числа операций от ГЛАВНОГО ПАРАМЕТРА (например, от размера входных данных) 

Показатель роста (асимптотика зависимости)

2. *Пространственная (ёмкостная) сложность*

Max объем использованной памяти+

+Дополнительная память (при необходимости)

Сравнение временной сложности двух алгоритмов

n	1	3	5	7	8	9	10		11	20	50
A, 2^n	2	8	32	128	256	512	1024		2048	1048576	1125899906842624
B, n^3	1	27	125	343	512	729	1000		1333	8000	125000
										131	900719925

На практике при малых размерах данных алгоритмы с худшей эффективностью часто показывают хороший результат

Классификация алгоритмов по стратегии решения задачи

- 1. Алгоритмы грубой силы (полного перебора)
- 2. Алгоритмы декомпозиции («разделяй и властвуй» или триада «анализ-декомпозиция-синтез»)
- 3. «Жадные» алгоритмы (оптимизационные алгоритмы: лучшее решение в любой момент времени)
- 4. Динамические алгоритмы (принятие решения с учетом будущих последствий на основе полученных ранее результатов)

Вывод 3-й и 4-й подходы используют 1-й, но «жадные» алгоритмы учитывают только то, что лучше всего в данный момент времени, а динамический подход рассматривает несколько решений проблемы, вычисляет их, сохраняет, а затем вызывает их для повторного использования.

Трудноразрешимые задачи

- **Задача о рюкзаке, о светофоре, о выполнимости КНФ, о коммивояжере, о СВЯЗНЫХ КОМПОНЕНТАХ и т.д.**

Задача о наполнении рюкзака

1. Формулировка задачи
2. «Жадная» стратегия
3. Перебор вариантов

1) Постановка задачи

Пусть имеется n неделимых предметов, каждому из которых поставлены в соответствие объём V_i и стоимость C_i

Требуется поместить в рюкзак вместимостью V набор предметов максимальной стоимости, суммарный объём которых не превышает объёма рюкзака

2) Жадный алгоритм (пример)

- 1 – 2 и 10 (объем и стоимость) 5
- 2 – 3 и 21 7
- 3 – 4 и 16 4
- 4 – 5 и 40 8
- 5 – 6 и 18 3
- 6 – 7 и 14 2

$V=18$

Как определить, получено ли лучшее решение?

Если есть предметы с одинаковой ценой, то какова стратегия выбора?

Если несколько рюкзаков, то как поступать?

3) Простой точный алгоритм неполиномиальной сложности

1. Перенумеруем все предметы
2. Установим максимум достигнутой стоимости $C=0$
3. Составим двоичное число с n разрядами, в котором единица в i -том разряде будет означать, что i -тый предмет выбран для укладки в рюкзак
4. Рассмотрим все комбинации, начиная от 000...000 до 111...111
Для каждой из них подсчитаем значение суммарного объёма V_k
 - (a) Если суммарный объём расстановки V_k не превосходит объёма рюкзака V , то подсчитывается суммарная стоимость C_k и сравнивается с достигнутым ранее максимумом стоимости C
 - (b) Если вычисленная суммарная стоимость превосходит максимум C , то максимум устанавливается в вычисленную стоимость C_k и запоминается текущая конфигурация

Упражнение Оцените время работы при поиске одного решения 1 нс

- N=10, 20, 50 (1 компьютер)
- N=20, 40, 80 (10 компьютеров)
- N=80, 100, 128
- (1 000 000 000 000 компьютеров)

секунды

10^2 1.7 минуты

10^4 2.8 часа

10^5 1.1 дня

10^6 1.6 недели

10^7 3.8 месяца

10^8 3.1 года

10^9 3.1 десятилетия

10^{10} 3.1 столетия

3 600 сек – 1 ч

86 400 сек – 1 сутки

31 536 000 сек – 1 год

Время нахождения точного решения полным перебором вариантов

$$\frac{2^{128} \times 10^{-9}}{10^{12}} \text{ секунд} \approx 10.8 \times 10^9 \text{ лет.}$$

NP-полная задача!!!

Статус таких задач до сих пор не
известен со времени постановки
проблемы в 1971 году

Парадигма NP-полной задачи

- 1. Не найден эффективный алгоритм, но и не доказано его отсутствие
- 2. Если существует эффективный алгоритм хотя бы для одной NPC-задачи, то и для других он может быть найден
- Некоторые NPC-задачи похожи на задачи с эффективными алгоритмами. Но небольшое изменение формулировки может значительно ухудшить эффективность известного алгоритма

2 основных класса алгоритмов

Итеративные

- В основе лежат циклы и условные операторы
- При анализе требуется оценить число операций внутри цикла и число итераций цикла
(операции суммирования)

Рекурсивные

- Разбивают большую задачу на фрагменты и применяются к каждому фрагменту в отдельности
- При анализе подсчитывается число операций, необходимых для разбиения задачи на части, выполнения алгоритма на каждой из частей и объединения отдельных результатов
(рекуррентные соотношения)

Установление корректности алгоритмов

- Метод математической индукции – доказательство тождества на множестве **всех** натуральных чисел



- Индуктивная функция и инвариант цикла

Индуктивная функция

Пусть аргументами функции f являются последовательности элементов множества M , а значениями — элементы множества N . Тогда, если значение функции f на последовательности x_1, x_2, \dots, x_n можно восстановить по её значению на последовательности x_1, x_2, \dots, x_{n-1} и по элементу x_n , то такая функция называется индуктивной.

$$\text{maximum}(x_1, x_2, \dots, x_n) = \max(\text{maximum}(x_1, x_2, \dots, x_{n-1}), x_n)$$

Инвариант — предикат (логическое выражение с переменными), сохраняющий своё значение после исполнения заданных шагов алгоритма

- **Инвариант цикла** – предикат, принимающий истинное значение перед началом цикла и после его прохождения

Доказательство корректности алгоритмов

1. Выбираем предикат (или группу предикатов), значение которого истинно до начала исполнения фрагмента
2. Исполняем фрагмент, наблюдая за поведением предиката
3. Если после исполнения предикат остался истинным при любых путях прохождения фрагмента, алгоритм корректен относительно значения этого предиката

Инвариант цикла и ММИ

Свойства инварианта цикла

- 1. Инициализация
- 2. Сохранение
- 3. Завершение
- Частые ошибки –
- 1-го шага
- Последнего шага

ММИ

- 1. База индукции
- 2. Предположение индукции
- 3. Индуктивный переход

Главное отличие - в **конечности** цикла

Примеры инварианта цикла

Поиск максимального элемента числового массива

- Соответствующая переменная на i -той итерации содержит наибольшее значение из всех «перебранных» элементов

Суммирование чисел массива

- Значение суммы для подмножества элементов, состоящих из первых i элементов массива на соответствующей итерации

Формула суммы первых n натуральных чисел

▸ $n := 9000 : s := 0 :$

for i **from** 1 **to** n

do

$s := s + i$

od:

$s : is \left(s = \frac{1}{2} \cdot (n^2 + n) \right)$

А если записать вместо n i ?



Что «лучше»:

1) использовать цикл или формулу?

2) Итеративную или рекурсивную процедуру?

▸ $n := 'n' : is \left(sum(k, k = 1 .. n) = \frac{1}{2} \cdot (n^2 + n) \right)$

true

true

$$\frac{1}{2} (n + 1)^2 - \frac{1}{2} n - \frac{1}{2} = \frac{1}{2} n^2 + \frac{1}{2} n$$

▸ $factor(sum(k, k = 1 .. n))$

$$\frac{1}{2} n (n + 1)$$

Вычисление процессорного времени

Для $n=9 \cdot 10^3, 10^4, 10^5, 10^6, 10^7$

Цикл. проц.: 0.001, 0.016, 0.187, 1.703, 17.016

Рек. Проц.: 0.047, 0.094, 0.407, «выс.ур. рек»

- При увеличении порядка циклическая процедура «затормозила» с $9 \cdot 10^8$

Вопрос: Что можно сделать для быстрого получения ответа при больших n ?

Этапы каждого уровня рекурсии

- 1. Разделение
- 2. Властвование
- 3. Комбинирование

- Различать прямую и косвенную рекурсии!

Вычисление факториала

```
> Fct := proc(n :: posint)
    if n = 1 then return 1
    else n · Fct(n - 1)
    fi
end proc;
```

Составьте итерационную процедуру и сравните

Первый шаг цикла и последний шаг цикла для вычисления 100!

```
> Fct(100); 100!;
```

93326215443944152681699238856266700490715968264381621468592963895217599993229915
6089414639761565182862536979208272237582511852109168640000000000000000000000
00

93326215443944152681699238856266700490715968264381621468592963895217599993229915
6089414639761565182862536979208272237582511852109168640000000000000000000000
00

(5)

Составьте рекурсивную процедуру для нахождения суммы n первых натуральных чисел

ВОПРОС Нахождение max из 4-х элементов

```
largest = a
if b > largest then
    largest = b
end if
return a
if c > largest then
    largest = c
end if
if d > largest then
    largest = d
endif
return largest
```

Какой из алгоритмов
«лучше»?

```
if a > b then
    if a > c then
        if a > d then
            return a
        else
            return d
        end if
    else
        if c > d then
            return c
        else
            return d
        end if
    end if
else
    if b > c then
        if b > d then
            return b
        else
            return d
        end if
    else
        if c > d then
            return c
        else
            return d
        end if
    end if
end if
```

```

if a > b then
  if a > c then
    if a > d then
      return a
    else
      return d
    end if
  else
    if c > d then
      return c
    else
      return d
    end if
  end if
else
  if b > c then
    if b > d then
      return b
    else
      return d
    end if
  else
    if c > d then
      return c
    else
      return d
    end if
  end if
end if

```

- Построение в виде двоичного дерева

