Основы анализа алгоритмов.

Оценка сложности алгоритма. Лекция 1.

Доцент КВТ Лапина О.Н.

Понятие алгоритма.

- *Алгоритм представляет собой описание способа решения некоторой задачи.*
- ***Математическая энциклопедия (М.: Советская энциклопедия, 1977)*** так определяет понятие алгоритма:
- **Алгоритм** – *точное предписание, которое задает вычислительный процесс (называемый в этом случае алгоритмическим), начинающийся с произвольного исходного данного (из некоторой совокупности возможных для данного алгоритма исходных данных) и направленный на получение полностью определяемого этим исходным данным результатом.*
- *Примеры: Алгоритм Евклида НОД, алгоритм поиска кратчайшего пути в графе и т.п.*

Основные требования к алгоритму.

- *1.**Каждый алгоритм имеет дело с данными***: входными, промежуточными, выходными. Для уточнения понятия данных вводится алфавит исходных символов (цифры, буквы и т.д.) и указываются правила построения данных (числа, слова, формулы и т.п.)
- *2. **Дискретность.*** Алгоритм состоит из конечного числа описаний шагов и эти шаги выполняются в дискретном времени
- **3. *Элементарность шагов.*** *Объем работ на каждом шаге не зависит от данных.* Например, элементарными шагами могут быть сложение, умножение, сравнение, пересылка чисел. Сравнение файлов не является элементарным шагом.
- **4. *Детерминированность (определенность)*** *алгоритма и его шагов.* Результат не зависит от случайных факторов.
- **5. *Конечность (финитность).*** *Алгоритм останавливается в некоторый момент времени.*
- ***6. Массовость.*** *Данные не должны быть уникальными.*
- *7. Результативность алгоритма* ***(сходимость).***

Понятие алгоритма.

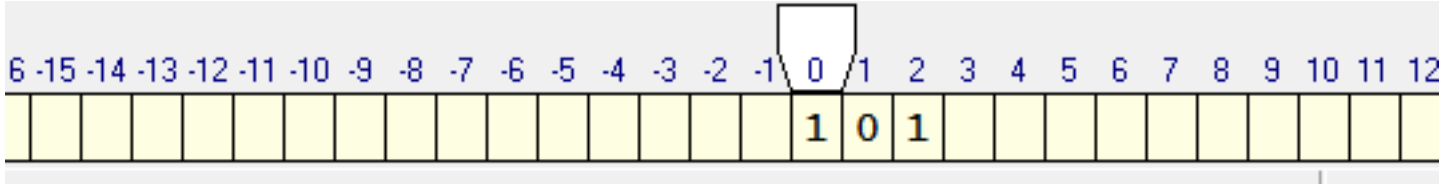
- **Пример 1.** Рассмотрим алгоритм с одним входным аргументом – натуральным числом k:
- *если* ***k=1,*** *то остановиться;*
- *если k четное, то* ***k=k/2;***
- *если k нечетное, то положить* ***k=3k+1;***
- *повторить, используя новое значение k.*
- *Требуется рассмотреть свойство конечности алгоритма.*
- В общем случае процесс не монотонный. Например, для k=40:
- k=40, 20, 10, 5, 16, 8, 4, 2, 1.
- В настоящий момент конечность алгоритма не доказана!
- Ю. Нивергельт, Дж. Фаррар, Э. Рейнгольд «Машинный подход к решению математических задач» (М.: Мир, 1977. С. 298).

Формальное определение алгоритма.

- ***Э. Дейкстра:*** *«Преимущество формального способа записи состоит в том, что он дает нам возможность изучать алгоритмы как математические объекты. Благодаря этому сумеем доказать теоремы о классах алгоритмов.»*
- В 20-30-е годы XX столетия предпринимались попытки формализовать понятие алгоритма.
- Для получения ***формального определения алгоритма*** необходимо уточнить понятия алфавита данных и формы их представления, понятия элементарных шагов и т.п.
- В теории алгоритмов для ведения точного понятия алгоритма принят подход, основанный на введении конкретной алгоритмической модели, в которой все требования выполняются естественным образом. Эти алгоритмические модели являются универсальными. Они и являются формальными понятиями алгоритма:
- ***Рекурсивные функции*** *(структурное программирование).*
- ***Машина Тьюринга*** *(микропрограммирование (прошивки))*
- ***Алгоритмы Маркова*** *(Языки символьной обработки информации: Пролог)*
- *Являются математически эквивалентными с точки зрения понятия алгоритма*, но порождают разные направления в программировании.

Формальное определение алгоритма.

- ***Машина Тьюринга .*** В 1937 году американский ученый Алан Тьюринг (A. Turing) предложил следующий способ описания алгоритма.



- ***Формат команды имеет следующий вид****а:* ***a q b r D,***
- где a – читаемый символ,
- b – символ, записываемый вместо символа a,
- q-текущее состояние,
- r - новое состояние,
- D – направление движения головки.

Формальное определение алгоритма.

- *Существуют различные модели машин Тьюринга.*
- ***Тьюринг*** Высказал тезис:
- ***Тезис Тьюринга.*** *Любой алгоритм может быть реализован на подходящей машине Тьюринга.*

- *Несколько раньше в 1936 г. Аналогичный тезис высказал американский логик Алонзо Черч.*
- ***Тезис Черча.*** *Класс задач, решаемых в любой формальной алгоритмической модели, совпадает с классом задач, которые могут быть решены интуитивно алгоритмическими методами.*

Формальное определение алгоритма.

- **Нормальные алгорифмы Маркова.**
- В 1951 году математик Марков Андрей Андреевич (1903-1979 гг.) в Трудах Математического института АН СССР им. В.А. Стеклова опубликовал свою формализацию понятия алгоритма, которую он назвал нормальным алгорифмом. *Он рассматривал алгоритмы, работающие со словами.*
- ***Нормальный алгорифм задается указанием 3-х объектов: алфавита А, алфавита*** $\{\rightarrow, \rightarrow ., |\}$ ***и схемы Z.***
- Пример. Задан алгорифм:
- Алфавит А= {а, б, в, г}.
- Схема $Z = | a \rightarrow \Lambda | б \rightarrow \Lambda | в \rightarrow \Lambda | г \rightarrow \Lambda |$
- Такой алгорифм переводит любое слово в пустое: $\alpha : P \Rightarrow \Lambda$

Разрешимые и неразрешимые задачи.

- ***Основным содержанием Теории алгоритмов является классификация задач по признаку алгоритмической разрешимости: является ли задача алгоритмически разрешимой, либо данная задача является алгоритмически неразрешимой.***
- В данном направлении получен ряд фундаментальных результатов. Например: ***Десятая проблема Гильберта,*** сформулированная в 1900 году, состоит в нахождении алгоритма решения произвольных алгебраических диофантовых уравнений. В 1970 году Юрий Матиясевич доказал ***алгоритмическую неразрешимость*** этой проблемы.
- ***Диофантовы уравнения*** (по имени древнегреческого математика Диофанта), алгебраические уравнения (x+y=1) или системы алгебраических уравнений с целыми коэффициентами, имеющие число неизвестных, превосходящее число уравнений, и у которых разыскиваются целые или рациональные решения.

Понятие сложности алгоритмов.

- В программировании *понятие сложности алгоритма* связано с использованием ресурсов компьютера.
- ***Временную сложность*** будем подсчитывать в исполняемых командах (элементарных шагов): количество арифметических операций, сравнений, пересылок чисел.
- ***Емкостная сложность*** будет определяться количеством скалярных переменных, элементов массивов, элементов записей или просто байт.
- В общем случае количество операций и требуемая память зависят от исходных данных, т.е. являются функциями исходных данных. *Такие функции называются функциями сложности.*
- $T_\alpha(V)$ - временная сложность алгоритма
- $S_\alpha(V)$ - емкостная сложность

Понятие сложности алгоритмов.

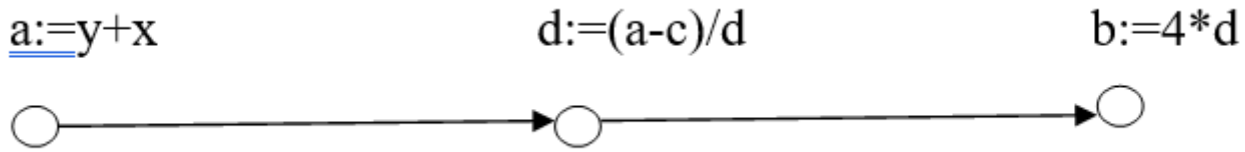
- **Пример 1.** Вычисление факториала числа x
- (k – счетчик операций)
- ***function*** *Factorial(x: integer): integer;*
- *var m, i: integer;*
- ***begin*** *m:=1; k=1;*
- *for i:=2 to x do*
- *m:=m*i; k=k+2*
- *Factorial:=m; k=k+1*
- *end.*
- *Подсчет количества операций:*
- Если сложность 1-й операции равна 1, то общая сложность:
- T(x)= 1+2(x-1)+1=2x=O(x). – функция сложности линейная

Основные методы и приемы анализа сложности.

- Отыскание функции сложности производится на основе анализа текста алгоритма.
- Если встречаются ***вызовы процедур***, то вызов не рассматривается как одна операция, подсчитывается количество операций в процедуре плюс сам оператор вызова (с единичной сложностью).
- Рассмотрим сначала ***нерекурсивные алгоритмы*.**
- С целью анализа алгоритм или программу удобно представлять ***управляющим графом (схема алгоритма).***
- Управляющий граф строится следующим образом. Каждому оператору присваивания или вызову процедуры ставится в соответствие вершина графа. Два последовательных оператора (вершины) соединяются стрелкой, указывающей порядок исполнения.

Основные методы и приемы анализа сложности.

- **Построение функции сложности:**
- ***Управляющий граф – линейный участок***. Сложность равна сумме весов вершин, принадлежащих линейному участку, если на участке нет вершин – вызов процедуры. Если такие вершины есть, то их вес равен функциям сложности соответствующих процед



- ***Управляющий граф содержит разветвления, но не содержит циклов***. Это означает, что вычислительный процесс в зависимости от исходных данных может направиться по одной из веток. При вычислении функции сложности мы можем рассматривать ***либо худший случай, либо средний случай.***
- Для худшего случая нужно вычислить вес каждой ветви и найти максимальный из весов. Это и будет сложностью процедуры.

Основные методы и приемы анализа сложности.

- **Построение функции сложности:**
- ***Управляющий граф содержит цикл, порожденный оператором for***. Если выражения, определяющие начальное и конечное значение параметра цикла - константы, мы можем подсчитать, сколько раз выполняется цикл, и вес цикла умножить на этот коэффициент. Если выражения зависят от исходных данных, то мы можем оценить количество повторений либо в худшем случае, либо в среднем.
- ***Управляющий граф содержит цикл, порожденный операторами while или repeat.*** В этом случае анализ вызовет больше затруднений, так как условия повторения цикла вычисляются в теле самого цикла (необходимо провести анализ самого алгоритма).

Понятие сложности алгоритмов.

- **Задачи для практики:**
- *Написать процедуру и найти функцию сложности для следующих алгоритмов:*
- **1. Скалярное умножение векторов;**
- **2. Перемножение матриц (nxn);**
- **3. Сортировка массива (вставкой; выбором; пузырьковая).**
- В данной задаче необходимо определить максимальную сложность алгоритмов (сложность в худшем случае) и минимальную сложность (сложность в лучшем случае), а также оценить среднюю сложность алгоритмов. Сравнить сложности для различных алгоритмов.

Список литературы.

- **Дж. Клейнберг, Е. Тардос** Алгоритмы. Разработка и применение. – СПб. Питер, 2018 г., 800 с.
- **Королев Л.Н., Миков А.И.** Информатика. Введение в компьютерные науки. – М.: Абрис, 2012.
- **Китаев А., Шень А., Вялый М.** Классические и квантовые вычисления. М., 1999.
- **Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн.** Алгоритмы. Построение и анализ. М.: Вильямс, 2005, 1296 с.
- **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М., 1983 г.

Список литературы.

- **Гашков, С. Б.** Теория алгоритмов и вычислений / С. Б. Гашков. — Санкт-Петербург : Лань, 2023. — 168 с. — ISBN 978-5-507-46897-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/352274
- **Тюкачев Н.А..** C#. Алгоритмы и структуры данных: Учебное пособие для вузов [Электронный ресурс] / Н.А. Тюкачев, В.Г. Хлебостроев — Электрон. дан. — Санкт-Петербург : Лань, 2023. — 232 с. — Режим доступа: https://e.lanbook.com/book/346067?category=1540
- **Миков А.И., Лапина О.Н.** Вычислимость и сложность алгоритмов: учеб. пособие. – Краснодар: КубГУ, 2013 г., 79 с.
- **Миков А.И., Лапина О.Н.** Сложность алгоритмов и задач: учеб. пособие. – Ростов-на-Дону: ЮФУ, 2014 г., 103 с.