# Лекции первого полугодия

1. От проблемы к решению. Информация, данные и знания. Сложность и борьба с ней. Инкапсуляция и модуляризация. Основы эффективной алгоритмизации.  
   Сущности и системы. Принцип абстракции. Процесс абстрагирования . Уровни абстракции. Интерфейсы и протоколы. Различие абстракции процессов и данных, объектная абстракция.
2. Абстрактные типы данных (АТД). Интерфейс и реализация АТД. Классификация АТД. Объектно-ориентированная парадигма и АТД.  
   Организация памяти в современных цифровых компьютерах. Структуры данных. Атомарные и комплексные данные, ссылки и их применение для организации комплексных структур данных. Индексы и указатели. Курсоры. Массивы и списки как базовые СД.
3. Коллекции и контейнеры. Структура контейнера. Операции извлечения, замены, добавления и удаления элементов. Классификация контейнеров. Значимость контейнеров. Особенности времени жизни элементов контейнеров. Библиотеки контейнеров.
4. Базовые линейные контейнеры – вектора, стеки, очереди и деки. Реализация базовых линейных контейнеров. Списочные, индексные и смешанные реализации.
5. Методы проектирования алгоритмов решения задач (1).  
   Информационный поиск и упорядочение. Порядковые статистики. Оглавление и антиоглавление, их построение. Систематизация алгоритмов сортировки. Базовые алгоритмы поиска в векторе и упорядочения вектора.
6. Множества и отображения (словари). Ключи. Реализации быстрого доступа по ключу. Хэш-таблицы. Требования к хэш-функциям. Хэш-функции для строк. Классификация способов разрешения хэш-коллизий. **Внешние цепи**, последовательное зондирование, двойное хэширование.
7. Эффективная реализация упорядоченных контейнеров. Деревья поиска. Бинарные деревья. Высота дерева поиска и её уменьшение, балансировка. Самобалансирующиеся деревья. **AVL-дерево**, RB-дерево. Рандомизированные деревья. B-дерево.
8. Другие СД для реализации АТД с логарифмическим временем выполнения всех базовых операций.  
   Очереди с приоритетами. Варианты реализации.  
   Методы проектирования алгоритмов решения задач (2).
9. Внешняя память и работа с ней. Операционная система и файловые системы. Базовые понятия иерархических файловых систем: файл, папка, том, путь, атрибуты файла. Универсализация доступа к линейным контейнерам в оперативной и внешней памяти. АТД «поток». Базовые операции: позиционирование, чтение, запись. Прямой и последовательный доступ.
10. Текстовые файлы. Кодировка символов, таблицы символов, Unicode. Строки и их разделители. Основные приёмы работы с файлами. Сериализация структур данных.
11. Структурная информация . Анализ структур систем в виде графовых моделей. Значимость структурной информации и задач её обработки. Представление структурной информации в памяти компьютера. Универсальные структурные контейнеры. АТД «обыкновенный граф». Основные операции АТД.
12. Варианты представления структурной информации в оперативной и внешней памяти. Матричные и списковые представления. Зависимость сложности выполнения операций АТД от структуры данных. Работа с весами вершин и рёбер.
13. Системы исследования (обходы) графов и базовые алгоритмы. Обходы графа в глубину и в ширину.
14. Основные классы задач анализа структурной информации и методы их решения. Методы проектирования алгоритмов решения *переборных* задач.

# Задачи

## Линейные структуры данных

Для каждой операции контейнера необходимо эмпирически установить вычислительную сложность как функцию времени выполнения операции от числа элементов контейнера и построить соответствующие графики.

Сравните характеристики своей реализации с имеющейся в библиотеке используемого языка/среды программирования (*STL*, *.NET* и т.п.).

1. Реализация АТД «Стек» (для любого типа данных, с операциями *IsEmpty*, *Clear*, *Pop*, *Push*) на основе:
   1. массива;
   2. односвязного списка.
2. Реализация АТД «Очередь» (для любого типа данных, с операциями *IsEmpty*, *Clear*, *Enqueue*, *Dequeue*) на основе:
   1. циклического массива;
   2. односвязного списка.

## Сортировка

Сравните характеристики своей реализации с имеющейся в библиотеке используемого языка/среды программирования (*STL*, *.NET* и т.п.).

1. Реализация сортировки вектора сравнимых объектов методом:
   1. сортировки выбором (*Select Sort*);
   2. сортировки вставками (*Insert Sort*);
   3. быстрой сортировки (*Quick Sort*);
   4. быстрой сортировки с выбором разделяющего элемента как медианы;
   5. быстрой сортировки с ограничением использования стека;
   6. сортировки слиянием (*Merge Sort*);
   7. сортировкой кучей (*Heap Sort*);
   8. гибридным методом.

## Информационный поиск 1

Реализация и тестирование алгоритмов:

1. Бинарного поиска по совпадению в векторе.
2. Бинарного поиска по близости в векторе.

## Информационный поиск 2

Суть задания заключается в получении практических навыков работы со сложными критериями поиска в линейных контейнерах. Элементами контейнеров являются *информационные записи* (далее – записи), состоящие из набора *полей*.

Приведём примеры состава записей.

|  |  |
| --- | --- |
| Информация о студенте   1. Фамилия 2. Имя 3. Отчество 4. Пол. 5. Дата рождения 6. Телефон 7. Группа 8. Рейтинг 9. Факультет | Информация о компании   1. Название 2. Организационно-правовая форма 3. Год основания 4. Юридический адрес 5. Фактический адрес 6. Число сотрудников |

### Задание

1. Опишите свой вариант состава информационной записи, то есть имена и типы полей. Число компонентов – не менее 6.
2. Создайте систему автоматической генерации массива исходных данных для тестирования алгоритмов поиска (см. пункт 4 для уточнения требований к генератору).
3. Предложите сложный критерий поиска (требования см. ниже).
4. Разработайте наиболее очевидную процедуру поиска по предложенному критерию на основе применения фильтра к каждой исходной записи. Для одного из условий придётся использовать дополнительный проход по массиву. Рассмотрите три случая тестовых исходных данных:
   1. результаты поиска содержат число записей, ненамного отличающееся от общего числа исходных записей;
   2. результаты поиска содержат число записей, много меньшее от общего числа исходных записей;
   3. результат поиска – пустое множество.
5. Разработайте также не менее двух более «интересных» (многоэтапных) процедур поиска.
6. Опишите количественные характеристики исходных данных и результатов поиска.
7. Продемонстрируйте работу разработанных программных средств.
8. Ответьте на следующие вопросы и объясните ответы.
   1. На сколько уменьшилось число записей после каждого этапа при многоэтапном поиске?
   2. Какое число операций сравнения полей пришлось совершить в каждом случае?
   3. Какое общее время работы процедуры поиска в каждом случае?
   4. Как изначально должен быть отсортирован массив записей для оптимизации поиска по Вашему критерию?
   5. Как изменится сложность поиска и процедура поиска, если из критерия исключить условия, не сводящиеся к проверке поля на абсолютное значение/множество/диапазон?

### Требования к критерию поиска

Формулировка задача информационного поиска: найти в контейнере все записи, удовлетворяющие сложному критерию (не менее 4 простых условий). Хотя бы одно условие должно формулироваться в виде, отличном от равенства абсолютным значениям, их множествам или диапазонам.

Пример критерия:

«Студентки 1984 года рождения, групп ПМ-09-01 и ПМ-09-02, входящие в верхние 10% по рейтингу среди всех студентов»

Здесь четыре условия:

1. пол – женский (равенство абсолютному значению);
2. дата рождения в диапазоне от 01.01.1984 до 31.12.1984 (принадлежность диапазону);
3. группа принадлежит множеству {ПМ-09-01, ПМ-09-02} (принадлежность множеству);
4. рейтинг таков, что при сортировке по возрастанию рейтинга запись оказывается в последних 10% записей (более сложное условие, требующее отдельной обработки).

## Более сложные контейнеры (Множества)

При реализации более сложных контейнеров (множеств, мультимножеств, словарей) необходимо как минимум обеспечить корректное выполнение базовых операций. Реализация дополнительных операций (взятых в квадратные скобки – []) повышает оценку. Задания повышенной сложности обозначены звёздочкой – \*.

Для каждой операции необходимо эмпирически установить вычислительную сложность как функцию времени выполнения операции от числа элементов контейнера и построить соответствующие графики.

Сравните характеристики своей реализации с имеющейся в библиотеке используемого языка/среды программирования (*STL*, *.NET* и т.п.).

1. Реализуйте АТД «Множество» (для чисел от 0 до *n*, где *n* изменяемо, с операциями *Clear*, *IsMember*, *Include*, *Exclude*, [*Union*, *Intersection*, *Difference*]) на основе **битового массива**.
2. Реализуйте АТД «Множество» (для любого типа данных, с операциями *Clear*, *IsMember*, *Include*, *Exclude*, [*Union*, *Intersection*, *Difference*]) на основе
3. неупорядоченного массива;
4. упорядоченного массива;
5. неупорядоченного связного списка;
6. упорядоченного связного списка.
7. Реализуйте АТД "Мультимножество" (для любого типа данных, с операциями *Clear*, *CountOf*, *Include*, *Exclude*, [*Union*, *Intersection*, *Difference*]) на основе:
8. неупорядоченного массива;
9. упорядоченного массива
10. неупорядоченного связного списка;
11. упорядоченного связного списка.

## Более сложные контейнеры с логарифмической сложностью выполнения всех базовых операций с одним элементом

1. Реализуйте АТД «Множество» (для любого типа данных, с операциями *Clear*, *IsMember*, *Include*, *Exclude*, [*Union*, *Intersection*, *Difference*]) на основе:
2. хэш-таблицы с внешними цепями;
3. хэш-таблицы с линейным зондированием;
4. *AVL*-дерева;
5. *RB*-дерева;
6. \* хэш-таблицы с двойным хэшированием;
7. \* рандомизированного дерева;
8. \* *B*-дерева;
9. \* списка с пропусками (*skip list*);
10. \* другой эффективной структуры данных по Вашему выбору.
11. Реализуйте АТД «Мультимножество» (для любого типа данных, с операциями *Clear*, *CountOf*, *Include*, *Exclude*, [*Union*, *Intersection*, *Difference*]) на основе:
12. хэш-таблицы с внешними цепями;
13. хэш-таблицы с линейным зондированием;
14. *AVL*-дерева;
15. *RB*-дерева;
16. \* хэш-таблицы с двойным хэшированием;
17. \* *B*-дерева;
18. \* рандомизированного дерева;
19. \* списка с пропусками (*skip list*);
20. \* другой эффективной структуры данных по Вашему выбору.
21. Реализуйте АТД «Словарь» (для пар <ключ, значение> любого типа, с операциями *Clear*, *HasKey*, *Value*, *Add*, *Delete*, [*Keys*, *Values*]) на основе:
22. хэш-таблицы с внешними цепями;
23. хэш-таблицы с линейным зондированием;
24. *AVL*-дерева;
25. *RB*-дерева;
26. \* хэш-таблицы с двойным хэшированием;
27. \* *B*-дерева;
28. \* рандомизированного дерева;
29. \* списка с пропусками (*skip list*);
30. \* другой эффективной структуры данных по Вашему выбору.
31. \* Реализуйте АТД «Упорядоченный словарь» (для пар <ключ, значение> любого типа, с операциями *Clear*, *HasKey*, *Value*, *Add*, *Delete*, *MinKey*, *MaxKey*, *Keys*, *ElementsByKeys*, [*Values*]).
32. \* Реализуйте АТД «Дважды упорядоченный словарь» (для пар <ключ, значение> любого типа, с операциями *Clear*, *HasKey*, *Value*, *Add*, *Delete, MinKey*, *MaxKey*, *MinValue*, *MaxValue*, *Keys*, *Values*, *ElementsByKeys*, *ElementsByValues*).

## Использование контейнеров

Вначале работы программа загружает исходные данные из файла. Наличие режима редактирования исходных данных даёт бонус.

### Работа с множествами и словарями

1. Для заданного текста составить частотный словарь, то есть определить, сколько раз каждое слово встречается в тексте, список пар <слово, число вхождений в текст> упорядочить по алфавиту (\* интересен вопрос учёта заглавных букв и других особенностей текста).
2. Используя записи, с которыми Вы работали в задаче о поиске, осуществите классификацию Ваших записей по некоторому критерию и найдите мощности классов.
3. Для *k* последовательностей с наименованиями объектов определите уникальные объекты для каждой из них (то есть объекты, присутствующие в первой последовательности и отсутствующие в остальных, и т.д.).
4. Для *k* стран заданы название страны и список названий городов в ней. Для введённого названия города определить страну, в которой он находится.
5. Для *k* сотрудников агентства переводов заданы ФИО и список упорядоченных пар языков (с какого на какой он сотрудник может перевести текст). Для введённого языка определить возможные языки перевода и отсортировать их по невозрастанию числа сотрудников, которые могут осуществить перевод на данный язык.

## Задачи анализа графов

Исходными данными является один или пара графов в файле. Формат – матрица смежности или *FO*(*FI*)-представление.

Большинство заданий предполагают, что результат – одно число. Некоторые задания требуют более сложных форматов выходных данных (векторов и матриц).

### Несложные задания

1. Радиус.
2. Диаметр.
3. Число центральных вершин графа.
4. Средний диаметр.
5. Упорядоченный по возрастанию набор степеней (полустепеней исхода и захода) вершин графа (орграфа).
6. Число компонент связности.
7. Число мостов.
8. Число точек сочленения.
9. Число блоков.
10. Длина нижнего обхвата.
11. Число периферийных вершин.
12. Число компонент связности, содержащих циклы длины 4.
13. Число компонент связности, содержащих не менее 3 различных цепей длины 4.
14. Число вершин с максимальной степенью (суммой полустепеней исхода и захода).
15. Число циклов (контуров) длины 3 и 4.
16. Количество вершин, смежных с парой вершин по выходящим дугам (для каждой пары вершин).
17. Количество вершин, смежных с парой вершин по входящим дугам (для каждой пары вершин).
18. Количество вершин, смежных с тройкой вершин по входящим дугам (для каждой тройки вершин).
19. Количество вершин, смежных с тройкой вершин по выходящим дугам (для каждой тройки вершин).
20. Число циклов (контуров) длины 3.
21. Число циклов (контуров) длины 4.
22. Число вершин и рёбер в окрестностях глубины 1 для каждой вершины.
23. Число вершин и рёбер замкнутой окрестности каждой вершины.
24. Число цепей (путей) наибольшей длины.
25. Упорядоченный по возрастанию значений вектор эксцентриситетов каждой вершины.
26. Число медиан.
27. Число изолированных вершин.
28. Число висячих вершин.
29. Число компонент сильной связности.
30. Признак «является ли граф слабо связной».
31. Число эйлеровых цепей в графе.
32. Число мостов в графе, дополнительном к заданному.

### Задания средней сложности

1. Радиус графа, полученный после гомеоморфного сжатия графа.
2. Диаметр графа, полученный после гомеоморфного сжатия графа.
3. Признак «является ли граф 2-связным».
4. Число вершин и рёбер в окрестностях глубины 2 для каждой вершины.
5. Число вершин, удалённых от заданной по кратчайшим маршрутам каждой из следующих длин: 1,2,3 и т.д., для каждой вершины.
6. Число вершин, удалённых от каждой пары вершин по кратчайшим маршрутам каждой из следующих длин: 1,2,3 и т.д.
7. Число вершин, удалённых от заданного ребра (дуги) по кратчайшим маршрутам каждой из следующих длин: 1,2,3 и т.д., для каждого ребра (дуги).
8. Цикломатическое число графа после удаления в нём мультирёбер (весов рёбер).
9. Число висячих вершин и сумма попарных кратчайших расстояний между ними.
10. Наибольшее число рёбер (дуг), удаление которых оставляет граф связным.
11. Диаметр графа после удаления в нём висячих вершин.
12. Число центральных вершин графа после удаления в нём висячих вершин.
13. Набор количества рёбер (дуг) подграфов, получаемых поочерёдным удалением пар вершин исходного графа.
14. Набор количества рёбер (дуг), получаемых поочерёдным удалением троек вершин.
15. Число компонент сильной связности орграфа.
16. Радиус графа после удаления в нём висячих вершин.
17. Средний диаметр графа после удаления в нём висячих вершин.
18. Число вершин и рёбер (дуг) окрестности каждой пары вершин.
19. Число вершин и рёбер (дуг) в замкнутой окрестности каждой пары вершин в графе.
20. Число рёбер (дуг) замкнутой окрестности каждой тройки вершин.
21. Длины циклов, входящих в фундаментальное множество циклов графа.
22. Длины циклов, входящих в фундаментальное множество циклов наименьшей суммарной длины в графе.
23. Число полных подграфов порядка 4 в заданном графе.
24. Число цепей (путей) кратчайшей длины между каждой парой вершин.
25. Число циклов длины 3, 4 и 5 в графе.
26. Радиус части графа, полученной удалением замкнутой окрестности вершины, для каждой вершины графа.
27. Диаметр части графа, полученной удалением замкнутой окрестности пары вершин, для каждой пары вершин графа.
28. Вектор степеней вершин в замкнутой окрестности вершины для каждой вершины исходного графа.
29. Вектор степеней вершин в замкнутой окрестности пары вершин, для каждой пары вершин исходного графа.
30. Набор степеней (полустепеней исхода и захода) в подграфе, образованном вершинами, удалёнными от заданной маршрутами длины 2 и 3, для каждой вершины исходного графа.
31. Минимальное число добавляемых к 1-связному графу рёбер для построения 2-связного графа.
32. Минимальное число добавляемых к 2-связному графу рёбер для построения 3-связного графа.
33. Число циклов графа, не имеющих попарно общих ребер.

### Задания повышенной сложности

1. Упорядоченный по возрастанию набор значений радиусов, определённых для подграфов замкнутой окрестности каждой вершины исходного графа.
2. Вершинная связность графа.
3. Рёберная (дуговая) связность графа (орграфа).
4. Живучесть графа, то есть минимальное число вершин, удаление которых увеличивает диаметр графа или приводит к тривиальному графу (графу без рёбер).
5. Рёберная (дуговая) живучесть графа (орграфа), то есть минимальное число рёбер (дуг), удаление которых увеличивает диаметр графа или приводит к тривиальному графу (орграфу).
6. Минимальное число пересечений рёбер (дуг) при расположении графа (орграфа) на плоскости.
7. Вектор количества полных подграфов порядка 2, 3, 4 и т.д., для каждой вершины.
8. Количество циклов (контуров) каждой длины, проходящих через каждую вершину.
9. Максимальное число изоморфных компонент связности.
10. Принадлежность графа к гамильтоновым графам.
11. Планарность графа (орграфа), то есть возможность его расположения на плоскости без пересечения рёбер (дуг).
12. Хроматическое число графа, то есть наименьшее число цветов вершин графа, необходимых для такой их раскраски, при которой никакие две вершины одного цвета не смежны.
13. Число вершин в полном подграфе максимального порядка в графе.
14. Число внешней устойчивости графа, то есть минимальное число вершин, в которые можно попасть из любой оставшейся вершины за один шаг.
15. Наибольшее число рёбер (дуг), удаление которых оставляет граф (орграф) 2-связным.
16. Наибольшее число рёбер (дуг), удаление которых оставляет граф (орграф) 2-рёберно-связным.
17. Число гамильтоновых циклов (контуров) в графе (орграфе).
18. Имеется ли гамильтонова цепь (путь) в графе (орграфе).
19. Число гамильтоновых цепей (путей) в графа (орграфе).
20. Число простых цепей (путей) всех длин в графе (орграфе).
21. Число циклов (контуров) для каждой из длин 3, 4, 5, 6.
22. Число циклов (контуров) в графе (орграфе).
23. Индекс цепной сложности графа.
24. Число каркасов в графе.
25. Число различных маршрутов для каждой пары вершин.
26. Число каркасов с наименьшим суммарным весом входящих в каркас рёбер.
27. Число верхних обхватов графа.
28. Число изоморфных вложений одного графа в другой.
29. Число вершин в наибольшей общего фрагмента (или пографа) всех компонент связности графа (компоненты связности графа рассматриваются как отдельные графы, вычисляются все возможные попарные максимальные изоморфные пересечения).
30. Количество цепей длины 2, 3, 4 и 5 в графе.
31. Число каркасов наименьшего суммарного веса по рёбрам.
32. Число изоморфных вложений K4 в граф.
33. Число изоморфных вложений всех деревьев на шести вершинах в граф.
34. Число неизоморфных графов, которые изоморфно вкладываются в заданный граф.
35. Число различных деревьев, которые изоморфно вкладываются в заданный граф.
36. Наименьшее число пересечений рёбер, необходимое для прорисовки диаграммы графа на плоскости.

### Прорисовка диаграмм графов (орграфов)

1. Универсальные методы.
2. Циркулярная.
3. Ортогональная.
4. Иерархическая.
5. Методом физических аналогий.
6. Специальные методы.
7. Деревьев и лесов.
8. С учётом симметрии.
9. Прорисовка ациклических орграфов.
10. Других классов графов.

### Некоторые прикладные задачи

1. Нахождение максимального потока.
2. Нахождение максимального потока минимальной стоимости.
3. Размещение центров обслуживания.
4. Поиск пути минимального веса, проходящего через несколько вершин в произвольном порядке.
5. Поиск *k* кратчайших путей между парой вершин.

### Уникальные задачи (требуют обсуждения)

1. Игры.
2. Генераторы классов и семейств графов.