1. What is the least number of operations necessary to sort an array of n arbitrary objects ?

В случае произвольных данных большинство самых быстрых алгоритмов требует минимум O(n log2 n) операций

2. What is the most efficient data structure to support appending to the end of the set, removing last element from the set, as well as accessing or updating *i*th value ? Provide explanation and complexities.

Наиболее эффективная структура для хранения набора данных называется список. В данной структуре элементы хранят ссылки на другие элементы вместе с данными, а так же ссылка на первый и последний элемент списка. В зависимости от функциональных требований к структуре он может односвязным (хранится информация только о следующем или предыдущем элементе) либо полносвязным (хранится информация о следующем и предыдущем элементе). В данном случае в задаче требуется иметь возможность вставки, удаления или изменения произвольного элемента. В случае с односвязным списком, в зависимости от того данные о каком элементе хранятся будет высокая эффективность либо вставки, либо удаления, такие структуры так же называют Стек и Очередь. Но стек и очередь не эффективны на операциях вставки внутрь или и не могут быть одновременно эффективными при операциях и удаления и вставки. Причиной тому недостаток информации о предущем или о следующем элементе. Поэтому в данной задаче наиболее эффективной структурой будет полносвязный список (двунаправленный связный список), в такой структуре удаление, вставка, изменение одинаково эффективны.

3. What is the most efficient data structure to support inserting into a set as well as selection and deletion of a value from random position ? Provide explanation and complexities.

В данном вопросе не идет речь о первом или последнем элементе из набора, поэтому видимо имеется в виду не сортированный набор данных. В таком случае наиболее эффективной структурой будет являться Хэш-таблица. Данная структура позволяет хранить пары ключ-значение и быстро выполнять перечисленные в вопросе операции (выбор, вставка и удаление элемента).

4. What is virtual memory ? What is it used for ? Why it is necessary ?

Виртуальная память это метод управления памятью, при котором для освобождения быстрой и ограниченной оперативной памяти используется перенос не требующихся в данный момент данных на медленную память, например жесткий диск. Данный метод используется в большинстве операционных систем для решения проблемы распределения оперативной памяти, например многие приложения, т.к. это позволяет предоставить приложению больше памяти чем доступно оперативной памяти. Это позволяет избежать сбоев в работе приложений из за недостатка оперативной памяти путем увеличения нагрузки на медленную память.

5. Write an efficient program which given two sorted arrays *A*[0*..n*] and *B*[0*..m*] finds all values that are present in both array

<https://github.com/vpuhoff/Test-Data-Science/blob/master/5.ipynb>

6. Describe an algorithm to find a vertex with a highest degree in an undirected graph. Describe the complexity. You do not have to write a program or a full algorithm. The description is sufficient.

Степень вершины - это инцидентность вершины и ребра, так как граф неориентированный, то задача облегчается.

Т.к. в условиях задачи ничего не сказано о входных данных приму, что изначально подается на вход матрица смежности:



И по этой матрице мы можем сразу легко дать ответ. Для каждой строки в матрице подсчитываем сумму элементов, строка с наибольшей суммой и будет представлять собой описание вершины с наибольшей степенью, например в матрице выше вершина V4 будет иметь наибольшую степень.

7. Describe what is Object Oriented Programming ? What are the key concepts and characteristics ? Where is it used ? Write a small program demonstrating this idea.

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Основные принципы:

* абстрагирование ( выделения важного)
* инкапсуляция (возможность описать команду «что делать», без одновременного уточнения как именно делать)
* наследование (организации родственных понятий: на каждом иерархическом шаге учитывать только изменения, не дублируя все остальное)
* полиморфизм (способность объекта использовать методы производного класса, который не существует на момент создания базового)

ООП реализовано в большинстве языков программирования и позволяет более точно описывать в рамках предметной области.

[Исходник](https://github.com/vpuhoff/Test-Data-Science/blob/master/7example/7example/7example/Program.cs):

|  |
| --- |
| class Человек |
|  | { |
|  | public int Вес; |
|  | public int Рост; |
|  | public string Пол; |
|  | public int Возраст; |
|  | public virtual void СделатьЧтото() |
|  | { |
|  | Console.WriteLine("Мне лень!"); |
|  | } |
|  | } |
|  | class Мужчина : Человек |
|  | { |
|  | new string Пол = "МУЖ"; |
|  | public override void СделатьЧтото() |
|  | { |
|  | Console.WriteLine(Пол+":Забил гвоздь!"); |
|  | } |
|  | } |
|  | class Женщина : Человек |
|  | { |
|  | new string Пол = "ЖЕН"; |
|  | public override void СделатьЧтото() |
|  | { |
|  | Console.WriteLine(Пол+":Полила цветы!"); |
|  | } |
|  | } |
|  |  |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | var Вася = new Мужчина(); |
|  | var Маша = new Женщина(); |
|  | var Чел = new Человек(); |
|  | Вася.СделатьЧтото(); |
|  | Маша.СделатьЧтото(); |
|  | Чел.СделатьЧтото(); |
|  | Console.ReadLine(); |
|  | //МУЖ: Забил гвоздь! |
|  | //ЖЕН:Полила цветы! |
|  | //Мне лень! |
|  | } |
|  | } |

8. Write a program that asks user’s name, records it in memory, prints the number of times it saw the name since it was last started and goes back to asking user’s name.

[Исходник](https://github.com/vpuhoff/Test-Data-Science/blob/master/8example/Prog/Program.cs):

static Dictionary<string, int> Memory = new Dictionary<string, int>();

static void Main(string[] args)

{

do

{

Console.WriteLine("Введите имя:");

string name = Console.ReadLine();

if (!Memory.ContainsKey(name))

{

Memory.Add(name, 1);

Console.WriteLine("Я впервые встретил данное имя.");

}

else

{

Memory[name]++;

Console.WriteLine("Я встречал данное имя {0} раз.", Memory[name]);

}

} while (true);

}

9.