Hack & Puzzle (HP)

Grégor JOUET

Assignement 3

Data Structures and language paradigms

gregor.jouet@devinci.fr

1 Requirements

1. You must have a working Ubuntu (or other Linux based) system installed **PRIOR** to this class. Failure to do so will prevent you from doing this assignment.
2. It is recommended, although not required, to work on this assignment in advance and to use the actual assignment day to ask questions and engage in discussion on the topic.

Contents

2 Foreword

Data Structures are a corner stone of modern computer science. They are a way to format and organize data in a specific way for a certain usage and are independent of the language used. Data structures can be used to access data efficiently and are often pre-programmed in the language you use. In this workshop you will discover and re-implement different data structures commonly used such as the linked list or hash map. If you wish to learn more about data structures please check out [1].
Language paradigms are the way programming languages are designed and thought to answer a specific need. Each language has its own characteristics and is suited for a category of use. For example, it is easier to write a web server in NodeJS than in C. But NodeJS might not be adapted for baremetal projects or to drive a motor. In this assignment, you will practice with different programming languages and discover their particularities.

3 Data Structures

In this first section, we will implement different data structures in C and Python. Though there are many, we will focus on Arrays, Linked Lists and Hash Maps.

3.1 Array

The most common data structure is the Array.

1. Make an integer array in C.
2. Print the memory address of each element in the array.
3. What do you notice about the addresses?
4. What does that imply if you want to remove an element from the array?
5. What is the rate of change of the **access time** of an array element in function of the number of elements in the array?
6. In C, how would you store items of different natures (types) in an array?
7. What is the common point between an array and pointer in C?
8. What is a string in C?
9. How do we know the length of a string in C? Implement that function.
10. With array of other types, how do we know their length in C?

3.2 Linked List

The Linked List is a data structure similar to an array but with significant differences.

1. What is the driving principle of a linked list?
2. How do you implement it in C?
3. Implement functions to create, add, remove and get the size of a linked list in C.
4. What is the rate of change of the **access time** of a linked list?
5. In Python, how is the Linked List implemented?

3.3 Hash Map

1. What is a *dictionary* in Python?
2. How is it implemented?
3. What is a hash function?
4. How is it used in a hash map?
5. What is a hash collision and how to avoid them?
6. In C, use the hashing code below to make a Hash Map of strings. Implement a function to init, add, remove and count the number of element in a hash map.

```c
// From http://www.cse.yorku.ca/~oz/hash.html
MAX_ELEMENT = 100 // Max items in your Hash Map
unsigned long hash(unsigned char *str)
{
    unsigned long hash = 5381;
    int c;

    while (c = *str++)
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */

    hash = hash % MAX_ELEMENT;
    return hash;
}
```

3.4 Tree

1. Find and explain the working principle of a tree.
2. What is a leaf? A node? The root node?
3. What is the depth of a tree?
4. What is a binary search tree? What are its uses?
5. Implement a binary search tree in python with methods to add new elements.
6. Implement a balancing algorithm for your BSTs.
7. In the case of a database, how would you use a BST to accelerate the search in a table? How is it called?
8. Give a toy implementation of this index.

3.5 Complexity

Algorithm complexity is an expression of the rate of change of the computing time necessary depending on the size of the input to be processed. This complexity is often represented with the *big-O* [2] notation $O(n)$.
Formally, Big-O notation mean that if, for two functions f and g, $f(x) = O(g(x))$ when $x \to +\infty$ is equivalent to $\exists M \in \mathcal{R}+,\ \exists x_0, x \geq x_0\ |f(x)| \leq Mg(x)$.
In the specific context of algorithm complexity, f is the algorithm execution time in function of the input size and g is the corresponding complexity.
For example, an algorithm executing a simple loop would be in $O(n)$ if n is the number of elements to iterate on. This would be a linear complexity.

1. What is the complexity of a simple lookup algorithm to find a value in a list?
2. What is the time complexity of an algorithm with 2 imbricated loops?
3. What is the time complexity of a dichotomy search algorithm?
4. What are the time complexities of the different access/lookup/change operations of the different data structures you have used so far (arrays, linked lists, hash maps)

4 Language Paradigms

Language paradigms are concepts and features embedded into a language that characterize it. Programming languages can be classified based on their paradigms. A well known paradigm is the ”object-oriented” programming languages. It is important to understand that paradigms are more important than the language syntax. Although you will learn both, the syntax is often the main barrier to writing code but the paradigm is the core of your work. **In this section, you will try out different languages**, learn their syntax and understand the different paradigms they involve.

4.1 Object Oriented Programming

To showcase the Object Oriented Programming paradigm, you will implement a Matrix class with a *add, sub, mult, pow, div, det* methods to respectively add, subtract, multiply, put to an integer power, divide and compute the determinant of your matrix in multiple languages.
Alternatively, you can implement a mini textual RPG with different characters and actions that can be controlled via text inputs on the console.

1. Implement the Matrix class or the mini rpg in Python
2. Implement the Matrix class or the mini rpg in C++
3. Implement the Matrix class or the mini rpg in Java
4. Point out the differences in and similarities between these languages

4.2 Event driven programming

Event driven programming is the paradigm of calling a routine when a certain event occurs.

1. In Javascript, read from and write to a file
2. In Java, use the **awt** and **swing** packages to show a window with a button a write to the console when that button is pressed.

5 To Go further

5.1 Specific Data structures

1. Explain how you would implement a program that read an audio file and sends its content to the audio hardware. What data structure would you use to store the audio? Give a toy implementation of this data structure. (In C :D)
2. When implementing a game engine, you need to check collisions between your different object’s hitboxes, what data structure would you use? Give a toy implementation of this use case in Python.

References

[1] John Bullinaria. *Data Structures and Algorithms*. School of Computer Science University of Birmingham, 2019. URL: https://www.cs.bham.ac.uk/~jxb/DSA/dsa.pdf.

[2] Wikipedia. *Big O notation*. URL: https://en.wikipedia.org/wiki/Big_O_notation.