Assignment on Algorithms, Data Structures for CS50L

Total points 9/10

Starting in 2021, all assignments in CS50L are out of 10 points. A score of 7 points or better (70%) is required to be considered to have "passed" an assignment in this course. Please do not resubmit an assignment if you have already obtained a passing score. You don't receive a final grade at the end of the course, so it will have no bearing on your certificate, and it will only slow down our graders!

Unlike CS50x, assignments in this course are graded on a set schedule, and depending on when you submitted, it may take up to three weeks for your work to be graded. Do be patient! Project scores and assignment status on cs50.me/cs50l (e.g. "Your submission has been received...") will likely change over time and are not final until the scores have been released.

Email *

maz786@outlook.com

Name *

Mazafer Ul-Raqib

edX Username *

Mazafer

What is your GitHub username?

You only need to tell us if you are concerned about checking your progress in the course and/or you want a free CS50 Certificate after you satisfy all of the requirements of the course. If you do not already have a GitHub account, you can sign up for one at https://github.com/join. You can then use this account to log in to cs50.me/cs50l to track your progress in the course (your progress will only show up after you have received at least one score release email from CS50 Bot, so do be patient!). Don't worry about seeing a 'No Submissions' message on submit.cs50.io, if you find that. The course collects submissions using Google Forms, and only the gradebook on cs50.me/cs50l is important! If you do decide to provide us with a GitHub username, BE CERTAIN IT IS CORRECT. If you provide the wrong username, you will not be able to see your scores.

https://github.com/maz786

This course is graded by human graders, and has a ZERO TOLERANCE plagiarism and collaboration policy. If *any* of your answers are copied and pasted from, or obviously based on (a) an online source or (b) another student's work in the course, in *any* of the course's ten assignments, you will be reported to edX and removed from the course immediately. There is no opportunity for appeal. There are no warnings or second chances. *

It is far better, we assure you, to leave an answer blank rather than risk it. This may be an online course, but it is offered by Harvard, and we're going to hold you to that standard.

- (●) I understand this policy and agree to its terms; I hereby affirm that I will not plagiarize any answers or collaborate with any other students in this course .

✓ In no more than one sentence: What does it mean for some algorithm to be in O(n)? 1/1

That is to say, in "big oh of n"?

Simply O(n) denotes that the algorithm's complexity or running time is directly correlated with the size of the input (n).

✓ Merge sort tends to be considered to run faster than many other sorting algorithms, but has a "space cost". What is meant by that? 1/1

The term "space cost" in computer science describes the extra memory space needed for an algorithm to complete its objective. Merge sort has a space cost of O(n), indicating that it requires additional memory space that is proportionate to the size of the input data set. This is because in order to combine the sorted subarrays back together, the merge sort algorithm needs to construct temporary arrays. As an alternative, some sorting algorithms, like quicksort, have a space cost of O(1), which means they just need the memory space necessary to hold the input data.

Merge sort is typically regarded as a more effective algorithm in terms of time complexity even if it has a larger space cost than certain other sorting algorithms. Merge sort is more effective than many other algorithms for large data sets since the time it takes to sort the data grows at a slower rate as the size of the data increases. Merge sort has a time complexity of O(n*log(n)), suggesting that it will take longer to sort a large data collection.

✓ Generally speaking, what's an array? 1/1

A collection of elements can be stored in an array, which is a type of data structure used in computer science. Each item can be accessed using its index, which is the place in the array where it is kept; this is how items are often stored in contiguous blocks of memory. A fundamental data structure that is utilised by numerous algorithms and programming languages is the array.

An array is a grouping of elements of the same data type, such as integers or floating point numbers, in the majority of programming languages. The index, which is an integer value that indicates the element's location in the array, is used to identify each element in the array. Usually, an array's first element has an index of 0, and its last element has an index that is one lower than the array's size.

Arrays are excellent for structuring data storage and organisation, and they are frequently used to implement other data structures like matrices and lists. They are also utilised by numerous algorithms to effectively store and handle enormous amounts of data.

✓ In the context of linked lists, what is a pointer (otherwise known as a reference)? 1/1

A pointer is a reference to a memory place in the context of linked lists that contains the address of another memory location. Each node in a linked list often contains some data as well as a pointer to the node after it in the list. The pointer enables the list's nodes to be connected to one another, enabling linear list traversal.

A pointer, also known as a reference, is a variable that stores the address of another variable in memory. In many programming languages, pointers are used to manipulate memory directly, allowing the programmer to create data structures such as linked lists and trees. Pointers can also be used to pass large blocks of data to functions more efficiently, as they allow the data to be accessed directly in memory rather than being copied.

In general, pointers are an important concept in computer science and are used in many different algorithms and data structures. They allow data to be organized and accessed in complex ways, and they are a fundamental building block of many computer systems.

✓ What advantages does a linked list offer over an array? 1/1

A linked list provides various benefits over an array, including the following:

Dynamic size: Unlike arrays, which have a fixed size after creation, linked lists can expand or contract as needed.

Insertion and deletion: Compared to an array, a linked list makes it simpler to add and remove members. With an array, adding or removing one member in the middle or at the start of the list necessitates moving every other element to create space, which takes time. By simply updating the pointers of the nearby nodes to point to the new node, additional entries can be added to a linked list.

Memory usage: As a linked list simply stores the data and a pointer to the next node rather than the whole array, it requires less memory than an array. If you're working with enormous data sets, this could be crucial.

Cache-friendly: Arrays can be less cache-friendly because they are kept in a single contiguous block of memory. Accessing linked list nodes is less likely to result in a cache miss since they are dispersed throughout memory. This can make linked lists faster than arrays for some operations, including traversal.

Additionally, there are several drawbacks to using linked lists. For instance, they typically require higher memory cost due to the pointers and are slower than arrays for index-based access. Linked lists are preferable for dynamic data sets where the size is unknown in advance and frequent insertion and deletion are necessary. Generally speaking, arrays are a solid choice when the size of the data is known in advance and when quick random access is required.

✓ What price(s) do you pay when using a linked list instead of an array? 1/1

When utilising a linked list as opposed to an array, there are a number of trade-offs to take into account:

Performance: Because you must follow n-1 pointers to access the nth element of a linked list, linked lists are typically slower than arrays for index-based access. The nth element of an array can be accessed in a constant amount of time because arrays can be directly indexed.

Memory overhead: Because each element in a linked list must hold a pointer to the next element in addition to the element's data, linked lists consume more memory than arrays. For huge data sets, this can be a considerable overhead.

Arrays are kept in a contiguous block of memory, which can make them more cache-friendly. This was already discussed. On the other hand, linked list nodes are dispersed across memory, which may make them less cache-friendly.

Implementation challenges: Because linked lists need to use pointers and dynamic memory allocation, they can be more challenging to implement than arrays. Because of this, they might not be as ideal for novices or circumstances where simplicity is important.

Linked lists are preferable for dynamic data sets where the size is unknown in advance and frequent insertion and deletion are necessary. Generally speaking, arrays are a solid choice when the size of the data is known in advance and when quick random access is required.

✓ Why is it problematic if a binary tree becomes unbalanced? 1/1

Certain operations on a binary tree may perform worse if the tree falls out of balance. For instance, the temporal complexity of tree operations like insert, delete, and search can decrease from O(log(n)) to O if the tree is strongly skewed to one side (n). This can greatly reduce the tree's efficiency and slow it down.

Because they may include more nodes than balanced trees, unbalanced trees may also require more memory. If the tree is very big and there isn't enough memory to store it, this could be an issue.

Finally, because imbalanced trees may not have the same predictable structure as balanced trees, they may be more challenging to utilise and comprehend. Due to this, designing algorithms that use the tree may become more challenging.

In order to maintain good speed and make a binary tree easier to use, it is crucial to keep it balanced. A binary tree can be balanced using a variety of methods, including self-balancing binary search trees like AVL trees and red-black trees.

✓ What problem(s) is/are solved when using a hash table? 1/1

To efficiently store and retrieve data from a collection, hash tables are employed as a solution. They are especially helpful when it is desired to access the data as rapidly as possible and no particular sequence in which it should be kept is required.

Data is stored in an array and is mapped to a specific index in the array by a hash function in hash tables. The data are input into the hash function, which then generates an integer that is used as the index where the data will be kept. Data is placed into the hash table at the index determined by the hash function. The hash function is used to determine the index at which the data is stored before the data is obtained straight from that location when data is requested from the hash table.

In comparison to alternative data structures like linked lists or binary search trees, this method allows data to be stored and retrieved on average in O(1) time. Therefore, hash tables are a useful data structure for swiftly storing and retrieving vast volumes of data.

Caches, in-memory data storage, and database indexing are just a few of the many uses for hash tables. They are a crucial tool in computer science and are applied to several systems and methods.

✗ Which of the following accurately characterizes the performance of linear search? 0/1

- ○ Linear search is in O(1)
- ○ Linear search is in O(n log n)
- ○ Linear search is in Ω(1)
- ○ Linear search is in Ω(n)
- ○ Linear search is in Θ(1)
- ◉ Linear search is in Θ(n) ✗

No correct answers

✓ Suppose that the names in a phone book with n names are not alphabetized but randomly ordered instead. What's an upper bound on the running time of searching that phone book for some name? 1/1

- ○ Constant time - O(1)
- ○ Factorial time - O(n!)
- ◉ Linear time - O(n) ✗
- ○ Loglinear time - O(n log n)
- ○ Logarithmic time - O(log n)
- ○ Quadratic time - O(n²)

No correct answers

Debrief

About how many MINUTES would you say you spent on this assignment? *

Just to set expectations for future students.

666