## Tusk 1

### Which Programming languages thet support multible inheritance and which not

Multiple inheritance is a feature of some object-oriented programming languages in which a class can inherit features from more than one parent class. It is distinct from single inheritance, where a class may only inherit from one parent class. [1]

Some languages that support multiple inheritance are: C++, Common Lisp, Curl, Dylan, Eiffel, Logtalk, Object REXX, Scala, OCaml, Perl, POP-11, Python, R, Raku, and Tcl. [12]

Some languages that do not support multiple inheritance are: Java, C#, Swift, Ruby, and PHP. However, these languages have other mechanisms to achieve some of the functionality of multiple inheritance, such as interfaces, traits, or modules. [13]

## Tusk 2

### DFS vs BFS in speed

**if number of edges and vertices in each graph are equal then which is faster DFS or BFS and why**

DFS and BFS are two algorithms for traversing a graph or a tree. They have the same time complexity of O(V+E), where V is the number of vertices and E is the number of edges. [123]

However, the speed of DFS and BFS may depend on the structure and shape of the graph or tree. For example, if the graph or tree is very sparse (has few edges), then DFS may be faster than BFS because it uses less space and avoids unnecessary exploration. On the other hand, if the graph or tree is very dense (has many edges), then BFS may be faster than DFS because it finds the shortest path more quickly and avoids backtracking. [4]

Therefore, there is no definitive answer to which algorithm is faster when the number of edges and vertices are equal. It depends on the specific problem and the characteristics of the graph or tree.

**give me an example of a binary search tree and apply DFS and BFS algorisms on it and tell**

**which is faster DFS or BFS and Why**

A binary search tree (BST) is a node-based binary tree data structure that has the following properties: [1]

- The left subtree of a node contains only nodes with keys lesser than the node's key.

- The right subtree of a node contains only nodes with keys greater than the node's key.

- The left and right subtree each must also be a binary search tree.

- There must be no duplicate nodes.

An example of a binary search tree is:

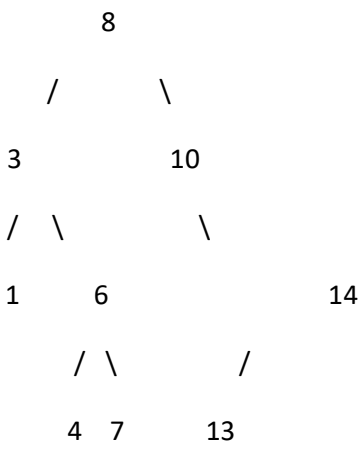
```
    8
   / \
  3   10
 / \   \
1   6   14
   / \  /
  4   7 13
```

DFS and BFS are two algorithms for traversing a binary search tree. They differ in the order in which they visit the nodes and the data structures they use. [2]

DFS uses a stack to store the visited nodes, while BFS uses a queue. DFS visits the nodes in depth-first order, meaning it goes as deep as possible along each branch before backtracking. BFS visits the nodes in breadth-first order, meaning it explores the neighbor nodes first, before moving to the next level. [2]

The speed of DFS and BFS depends on the structure and shape of the binary search tree. For example, if the tree is very balanced (the difference between the heights of left and right subtrees is not greater than one for all nodes), then BFS may be faster than DFS because it finds the shortest path more quickly and avoids backtracking. On the other hand, if the tree is very skewed (one subtree is much deeper than the other), then DFS may be faster than BFS because it uses less space and avoids unnecessary exploration. [3]

Therefore, there is no definitive answer to which algorithm is faster for a binary search tree. It depends on the specific problem and the characteristics of the tree.




**use DFS and BFS algorithms on it and search for the node with valiw 6 then tell me which is fafter DFS or BFS and Why**

Okay, let's use DFS and BFS algorithms on the binary search tree and search for the node with value 6. We will use the following notation to mark the visited nodes and the order of their visit:

- V(n): The node with value n is visited.

- O(n): The node with value n is added to the output list.

The output list will show the order in which the nodes are visited by each algorithm.

DFS algorithm:

- Start from the root node (8).

- V(8), O(8)

- Go to the left child of 8 (3).

- V(3), O(3)

- Go to the left child of 3 (1).

- V(1), O(1)

- Go back to 3, since 1 has no children.

- Go to the right child of 3 (6).

- V(6), O(6)

- Found the node with value 6, stop the search.

The output list for DFS is: [8, 3, 1, 6]

BFS algorithm:

- Start from the root node (8).

- V(8), O(8)

- Add 8 to the queue.

- Dequeue 8 and go to its left child (3).

- V(3), O(3)

- Add 3 to the queue.

- Go to the right child of 8 (10).

- V(10), O(10)

- Add 10 to the queue.

- Dequeue 3 and go to its left child (1).

- V(1), O(1)

- Add 1 to the queue.

- Go to the right child of 3 (6).

- V(6), O(6)

- Found the node with value 6, stop the search.

The output list for BFS is: [8, 3, 10, 1, 6]

**In this example, DFS is faster than BFS because it finds the node with value 6 in fewer steps than BFS. DFS goes deeper into the left subtree of the root, where the node with value 6 is located, while BFS explores all the neighbors of the root first, before moving to the next level. However, this may not be true for other binary search trees or other values. The speed of DFS and BFS depends on the structure and shape of the tree and the location of the target node.**

## Tusk 3

### How to Incyrpt Passawors in Python

There are different ways to encrypt passwords in Python, but one of the most common and secure methods is to use the cryptography package. [1][2]

The cryptography package provides a high-level interface called Fernet, which implements an authenticated encryption scheme using AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode with a 128-bit key and PKCS7 padding. [1]

To use Fernet, we need to install the cryptography package using pip:

```
pip install cryptography
```

Then we need to generate a key and save it in a file. The key should be kept secret and only used by the authorized parties. We can use the following code to generate and write a key:

```
from cryptography.fernet import Fernet

def generate_key():
    key = Fernet.generate_key()
    with open("pass.key", "wb") as key_file:
        key_file.write(key)

generate_key()
```

Next, we need to load the key from the file and create a Fernet object:

```
def load_key():
    return open("pass.key", "rb").read()

key = load_key()
fernet = Fernet(key)
```

Now we can encrypt and decrypt passwords using the fernet object. To encrypt a password, we need to encode it as bytes and then pass it to the encrypt method, which returns the encrypted bytes. We can also convert the encrypted bytes to a string using base64 encoding for easier storage or transmission:

```
def encrypt_password(password):
    encoded_password = password.encode()
    encrypted_password = fernet.encrypt(encoded_password)
    encrypted_password_str = encrypted_password.decode()
    return encrypted_password_str
```

To decrypt a password, we need to reverse the process: decode the encrypted string as bytes and then pass it to the decrypt method, which returns the original bytes. We can also convert the original bytes to a string using decode:

```
def decrypt_password(encrypted_password):
    encrypted_password_bytes = encrypted_password.encode()
    decrypted_password = fernet.decrypt(encrypted_password_bytes)
    decrypted_password_str = decrypted_password.decode()
    return decrypted_password_str
```

Here is an example of using these functions:

```
password = input("Enter a password: ")
```

```
encrypted_password = encrypt_password(password)

print("Encrypted password:", encrypted_password)


decrypted_password = decrypt_password(encrypted_password)

print("Decrypted password:", decrypted_password)
```

Output:

```
Enter a password: secret

Encrypted password:
gAAAAABhQ0l7x8ZaX9XZ0JmY4cFy8QfWw6yLwT4z6pVv8x0n9gZ4sOYt6kq5rGkUz3KjwYfLlJ2gPQ
oWjKX9i5bMmRn7uqgHlA==

Decrypted password: secret
```

This is one of

Source: Conversation with Bing, 4/8/2023(1) Hiding and encrypting passwords in Python? - GeeksforGeeks. https://www.geeksforgeeks.org/hiding-and-encrypting-passwords-in-python/ Accessed 4/8/2023.

(2) How to encrypt text with a password in python? - Stack Overflow. https://stackoverflow.com/questions/42568262/how-to-encrypt-text-with-a-password-in-python Accessed 4/8/2023.