InterPlanetary File System

PREPARED FOR

Data Structures (CS2001) Semester Project Fall2023

PREPARED BY

Abdullah Zubair (22i1239 F)- Group Leader

Safii Muhammad Hashir (22i1293 G)

Tauha Imran (22i-1239 G)

SUMMARY

The IPFS software offers a user-friendly solution akin to BitTorrent, simplifying the publication of content and ensuring accessibility anytime, anywhere. While this project primarily focuses on file sharing as the sole application of IPFS, it's crucial to recognize that IPFS extends its capabilities beyond this realm. The implementation will specifically address key properties of IPFS, showcasing its potential for more diverse functionalities.

The project emphasises the inherent strengths of IPFS, such as decentralised and distributed storage, enabling users to publish content with unparalleled ease. This decentralised nature ensures robust accessibility, breaking free from traditional centralised models.

The Project was led and Managed By Abullah Zubair.

The Technical components of the project consist of Data Structures

- B-tree
- Circular Doubly-Linked Lists
- SIngly Linked Lists
- Arrays, BigInteger
- String

The Algorithms comprise of

- Routing Mechanisms
- Hashing Mechanism
- SHA-1 encryptions
- FIle Handling
- Windows API
- Routing History mechanisms
- User History Mechanisms

Function Overview

Name	Description	Parameter Arguments	Return Arguments
B-Tree.h			
1 - FileNode			
Stores hashed key in 'Hkey' and stores File Path in 'value'			
Constructor	Creates a FileNode with hashed key 'h' and file path 'va'	(string h = "", string val = "")	none
2 - BNode			
Stores hashed keys in an array of FileNodes, stores children in an array of Pointers to FileNode. Keeps count of number of keys for array 'keys' and keycount+1 for array 'children'. Stores a boolean 'isLeaf' for if this BNode is a Leaf Node.			
Constructor	Makes a Bnode with an Array of FileNodes as keys of max size order-1 and an array of Bnode pointers as children with a size of 'order'	(int order)	none
insertInNode	Simple inserter used inside the insert function in Btree	(string val, string rv)	int
3 - BTree			
Stores Pointer to BNode 'root' and minimum and maximum values for both keys and children as 'minkey', 'maxkey', 'minchild', and 'maxchild' respectively.			
Constructor	Assigns 'root' a Bnode from heap with the order of 'order1' and calculate and assigns all the values for *'minkey', 'maxkey', 'minchild', and 'maxchild'*	(int order1)	none
search	Requires an integer declared in another scope to update size of array, it searches for the key val and adds all the same key FileNodes into an array which it returns. It is an array of FileNode pointers.	(string val, int& sizereturn)	FileNode**
searchrecurs	Helper function for search function by returning an array of files with the same key done recursively.	(BNode* n, FileNode**& f, int& s, string val)	void

searchall	Adds all FileNodes into an array and Returns it to deletionmachineB	(BNode* n, FileNode**& f, int& s, string val)	void
searchrecursless	Helper function for additionmachineB function by returning an array of files with the same or smaller keys done recursively.	(BNode* n, FileNode**& f, int& s, string val)	void
findparent	Returns pointer to parent of a BNode n, and returns NULL if n is root	(BNode* n)	BNode*
insert	Inserts a FileNode into a BNode in the Btree with the key val and path rv	(string val, string rv)	void
deleteNode	1. Deletes the first found FileNode in the keys of a Bnode with the key val. Will not do anything if the key is not found. Prints if the key is found or not. 2. Can also specify a path for deletion in the case of duplicates.	(string val, path=" ")	void
additionmachineB	Used for adding machines so that respective Btree files are removed from one tree and shifted to the other.	(BTree* prevB, string comparitor, string home)	void
deletionmachineB	Used for deleting machines so that respective Btree files are removed from one tree and shifted to the other.	(BTree* prevB, string comparitor, string home)	void
Name	**Description**	**Parameter Arguments**	**Return Arguments**
DHT_RING.h			
1- DHT_ring			
A doubly-Circular linked list, made to access machines in IPFS, also keeps track of capacity of DHT ring during addition and removal of any machine, It provides functionality of adding and removing files from the machine by coordinating with B-trees using SHA1 generated keys. Member Variables: (start , end (Node type pointers to maintain the ring),Bitcount(biginteger to hold maximum machines that DHT can support),bits(user specified system size) ,treeorder(user specified b-tree order))			
DHT_ring	Default constructor	None	None
DHT_ring	Customized constructor	int bit_val int BtreeOrder	None
addmachine	Create directory for the machine, creates machine and modify fingertables for the DHT ring.	string val	void
pass_limits	Newly created node is passed to this and updated ring node start and end are passed	Node* create	void

	to machine for the sake of creating fingertables.		
insert	Adds machine along with checking if the machine exists in DHT ring	string val	bool
print_fwd	Ascending printing of DHT	void	void
print_rev	Descending printing of DHT	void	void
print_fingertable	Create fingertable and print them according to the user selection whether user wants to view the specific machine fingertable or whole DHT ring's Fingertable	void	void
createfingertable	Traverses the DHT ing and creates Fingertables for each machine in the DHT ring.	void	void
automatic_activation	Automatically checks the ring and inserts a random machine to the system	int	void
inputfile	Gets fila path and get the hash of the file content and inserts it into that particular machine	string Machinei, string filepath bool input = true	void
removemachine	Deletes machine and transfer its file contents stored in b-tree of it to the next machine in flyer	string val	void
display_node_Btree	Takes in machine id as string and search and displays the b-tree	string machineId	void
display_node_files	Gets string input of node id and displays its file.	string machineId	void
display_node_files_to_delete	Displays file of the machine along with the menu to specify the file that should be deleted	string machineId	void
Name	**Description**	**Parameter Arguments**	**Return Arguments**
dhtNodes.h			
1-FingerTable			
This class is a node that is used to provide functionality for creating fingertable. When fingertable connects they form a singly linked list. *int index;* *BigInteger machineId;* *Node* pointerToNode;* *FingerTable* next;*			
FingerTable()	Default constructor	None	None

setIndex	This is to set index of fingertable	Int val	void
getIndex	This is to get index value	none	int
setMachineId	This is to set machine id to fingertable node	BigInteger val	void
getMachineId	This is to get value of Machine id	None	BigInteger

2-Node

This class creates nodes that are used in doubly-circular linked list and are considered as machines of DHT ring

```
BigInteger id, modulus_value;
bool active;
std::string filename;
FingerTable* finger_table;
BigInteger base;
int bitcount;
Node* next;
Node* prev;
Node* DHT_start, * DHT_end;
BTree* filesystem;
```

Node(int order = 5)	Parameterized constructor to pass order of b-tree.	Int val	none
Node(BigInteger id_val, int bit_val, int order)	Parameterized Customised constructor that sets system bit value along with order of b-tree in each node to maintain proper functionality of Finger table and b-tree	BigInteger id_val int bit_val, int order	None
searchNextActiveNode	This function is to find and store the pointers to machine in Fingertable node	BigInteger val FingerTable*& obj	void
getId()	Returns the Machine id	None	BigInteger
setId	Sets the Machine id	BigInteger id_val	void
add_remove_file	Inserts file into b-tree along with maintaining the files in directories	string key string filepath bool insert	void
createFingerTable()	Creates Finger table for that paricular node according to system bit size	None	void
addFingerTable()	Creates linked list of Finger table according to system bit size	None	void
deleteFingerTableList()	Deletes the linked list of finger tables	None	void
drawFingerTable()	Prints finger table along with indexes for that particular node	None	void
splitaddmachine	This function is a reroute function for 'additionmachineB' in Btree, and sends the	(Node* added, Node* nextadd)	void

	respective Btrees and hashed keys only when the Btree is not empty.		
splitdeletemachine	This function is a reroute function for ‘deletionmachineB’ in Btree, and sends the respective Btrees and hashed keys only when the Btree is not empty.	(Node* added, Node* nextadd)	void

Name	Description	Parameter Arguments	Return Arguments
BigInteger.h			
1-BigInteger			
This class is capable of dealing with huge numbers that are beyond the scope of int, long long int and double. The purpose of this class is to free the IPFS system from the restriction of size of values. *string value*			
BigInteger()	Default constructor	None	None
BigInteger	Parameterized constructor	String str	None
operator+	Calls ‘add’ function with operator overloading.	const BigInteger& other	BigInteger
operator-	Calls ‘subtract’ function with operator overloading.	const BigInteger& other	BigInteger
operator*	Calls ‘multiply’ function with operator overloading.	const BigInteger& other	BigInteger
operator/	Calls ‘divide’ function with operator overloading.	const BigInteger& other	BigInteger
operator%	Calls ‘/’ function and uses it to extract remainder.	const BigInteger& other	BigInteger
power	Calls ‘*’ function and uses that to achieve power.	Int exponent	BigInteger
add	Adds two BigIntegers (strings) using simple carry addition.	const string& a const string& b	string
subtract	Subtracts two BigIntegers (strings) using simple borrow Subtraction.	const string& a const string& b	string
multiply	Multiplies two BigIntegers (strings) using simple carry functionality.	const string& a const string& b	string
reverse	Reverse the BigInteger string	string& str	string
divide	Divides two BigIntegers (strings) using simple borrow functionality.	const BigInteger& numerator	string

		const BigInteger& denominator BigInteger& quotient BigInteger& remainder	
operator<	Operator overloading of 'smaller than'	const BigInteger& other	bool
operator==	Operator overloading of 'is equal to'	const BigInteger& other	bool
operator!=	Operator overloading of 'is not equal to'	const BigInteger& other	bool
operator>=	Operator overloading of 'greater than equal to'	const BigInteger& other	bool
operator>	Operator overloading of 'greater than'	const BigInteger& other	bool
operator=	Operator overloading of 'equal assigning'of a string		BigInteger&
operator=	Operator overloading of 'equal assigning'of a BigInteger	const BigInteger& other	BigInteger&
generateRandomBigInteger	Generates a random string for BigInteger by assigning each digit as a random number.	Int numdigits	BigInteger
display()	Prints the BigInteger 'value' (string)	None	void

Name	Description	Parameter Arguments	Return Arguments
router.h			
1- router			
Stores a ***std::string traversed_path;*** to store the path to be sent into history_mech, and a ***bool startup;*** to check if a routing is in session. The class hold the following functions to route and aid in routing			
router	Default Constructor	None	None
route_next_node	Makes a single route and starts a routing session until manually terminated. Only routes once to the next routing location. Changes values arguments passed by reference.	Node*& current, std::string ID BigInteger& srch_va	Void

terminate_routing	Manual routing session termination, ends routing. Pass argument True for successful routing and false for failed routing	bool result	void
display_routing_track	Display the routing track taken in the routing session	None	Void
route_to_node	Automatically routes the node to the desired location,starts and ends the routing session, and returns a pointer to that location set . Requires you to pass the start pointer of the ringDHT data structure (pointer to first set)and an ID value of desired location .	Node* start = nullptr BigInteger endID = BigInteger("0")	Node* curr
is_in_correct_set	Checks whether your routing is correct and your pointers lie in the correct sets with respect to the properties of the IPFS	Node* start_of_ring Node* curr BigInteger Id BigInteger MAX	bool
Name	**Description**	**Parameter Arguments**	**Return Arguments**
hashing.h			
1- hashing			
This class contains ***BigInteger modulous;*** to be able to generate hash key			
hashing(...)	Parameterized Constructor taking in the number of bits of the identifier space	int bits = 0	none
displaymodval()	Displayed the value of modulus data member	None	void
generate_key(...)	Takes an argument of the sha1x encryption hex string (lowercase ascii) and generates a hash key according to the limit determined by the modulous data member	Std::string sha1x_value	BigInteger
SHR(..)	Takes a binary string and does a simple shift right and sets MSB to zero/'0'	std::string& bits	void
convert_to_binary(...)	Takes a hex string and returns it's binary conversion	std::string hex_str	std::string
convert_to_decimal(...)	Takes a Binary string and returns it's decimal string value in the form of BigInteger data type	std::string binary_str	BigInteger
Name	**Description**	**Parameter Arguments**	**Return Arguments**
sha1x.h			
1- Macros & Global functions			

sha1x(...)	**Major SHA-1 function that takes in string of content of the file and returns an encryption for SHA-1's 160 bit hex string (in lowercase ascii for hex values a,b,c,d,e,& f)**	**const std::string& string**	**std::string**
#define sha1x_ops_rotateWord(value, bits) (((value) << (bits)) | (((value) & 0xffffffff) >> (32 - (bits))))			Rotates a single block.
#define sha1x_block(i) (block[i&15] = sha1x_ops_rotateWord(block[(i+13)&15] ^ block[(i+8)&15] ^ block[(i+2)&15] ^ block[i&15],1))			
#define MAX(i1, i2) (i1 > i2 ? i1 : i2)			max value returner
#define MIN(i1, i2) (i1 < i2 ? i1 : i2)			min value returner

//operation0 - a constant operation
// t>=16 :: W[s] = sha1_rotateWord(W[(s + 13) & 0xf] ^ W[(s + 8) & 0xf] ^ W[(s + 2) & 0xf] ^ W[s], 1);

#define
sha1x_operation0(v,w,x,y,z,i) z += ((w&(x^y))^y) + block[i] + 0x5a827999 + sha1x_ops_rotateWord(v,5); w=sha1x_ops_rotateWord(w,30) ;

//operation1 a constant operation
// 0>= t<20 :: W[s] = sha1_k[0] + ((B & C) | (~B & D));

#define
sha1x_operation1(v,w,x,y,z,i) z += ((w&(x^y))^y) + sha1x_block(i) + 0x5a827999 + sha1x_ops_rotateWord(v,5); w=sha1x_ops_rotateWord(w,30);

//operation2 - a constant operation
// t<40 :: W[s] = sha1_k[1] + (B ^ C ^ D);

#define
sha1x_operation2(v,w,x,y,z,i) z += (w^x^y) + sha1x_block(i) + 0x6ed9eba1 + sha1x_ops_rotateWord(v,5); w=sha1x_ops_rotateWord(w,30);

//operation3 - a constant operation
// t<60 :: W[s] = sha1_k[2] + ((B & C) | (B & D) | (C & D));

#define
sha1x_operation3(v,w,x,y,z,i) z += (((w|x)&y)|(w&x)) + sha1x_block(i) + 0x8f1bbcdc + sha1x_ops_rotateWord(v,5); w=sha1x_ops_rotateWord(w,30);

// operation4 - a constant operation
// t<80 :: W[s] = sha1_k[3] + (B ^ C ^ D);

#define
sha1x_operation4(v,w,x,y,z,i) z += (w^x^y) + sha1x_block(i) + 0xca62c1d6 + sha1x_ops_rotateWord(v,5); w=sha1x_ops_rotateWord(w,30);

2- SHA1x

```
//  Using library standard data members for SHA-1

  //using macros manipulate memory correctly
    typedef unsigned long int int32;   /* just needs to be at least 32bit */
    typedef unsigned long long int64;  /* just needs to be at least 64bit */

    //contents...
    static const unsigned int DIGEST_h = 5;  /*sha1_h[5] integers per SHA1 digest  */
    static const unsigned int BLOCK_INTS = 16;  /* number of 32bit integers per SHA1 block */
    static const unsigned int BLOCK_BYTES = BLOCK_INTS * 4;

    //makes 32 bit integers
    int32 digest[DIGEST_h];
    std::string buffer; //our output string
    int64 numTansforms;//encryptions stored in here

 //private functions
     void reset(); //resets constant values
     void sha1x_function(int32 block[BLOCK_BYTES]); //makes blocks
  //makes buffer to block
    static void buffer_to_block(const std::string& buffer, int32 block[BLOCK_BYTES]);
     static void readXtract(std::istream& is, std::string& s, int max);//extractor
```

Name	Description	Parameter Arguments	Return Arguments
SHA1x();	empty constructor , resets the sha1 mechanism	None	None
sha1x_update(...);	function to turn string into istringstream	const std::string& s)	void
sha1x_update(...);	function to update values as istringstream	std::istream& is	void
sha1x_final(...);	Runs encryption rounds and operations	None	std::string
from_file(...);	Makes sha1x object and calls sha1x_fnal()	const std::string& filename	std::string

fileHandler.h

1-Global functions

```
// Log file paths
string filename_log_IPFS = "C:\\IPFS data\\IPFSlog.txt";
string filename_log_ROUTER = "C:\\IPFS data\\IPFSRouterlog.txt";
```

filename_extractions(...)	From full file path this function is used to extract the filename	std::string path	std::string
append_duplicates(...)	This function takes in a file, appends the duplicate id at the end and return the string to handle duplicates	std::string path int id	std::string

mergepath(...)	This function is used to merge a filename with some path and return a proper path	std::string path, std::string newname	std::string
copyFileToDirectory(...)	Function to copy a file to a specified directory	const string& sourceFilePath const string& targetDirectoryPath	bool
copyFileToDuplicateDirectory(...)	This function is used to take in a file path and destination and renames the file and copy the file to the destination .	const string& sourceFilePath const string& targetDirectoryPath const string& newFileName	bool
	bool deleteFile(const string& filePath)		
createDirectory(...)	Function to create a directory	const string& directoryPath	bool
removeDirectory(...)	Function to remove a directory and its contents recursively	const std::string& directoryPath	bool
stringToLPCWSTR(...)	Function to convert std::string to	const std::string& str	LPCWSTR
searchfile(...)	Function to open a file dialog for file selection	const std::string& initialPath	std::string
read_file_from_path(...)	Function to read content from a file	string filepath = "?"	std::string
writeToFile(...)	Function to write content to a file with optional timestamp	const string& content const string& filename bool append = true bool time = true	void

Project Pictures

(the following are a few snapshots showing the building,running and testing of the IPFS)

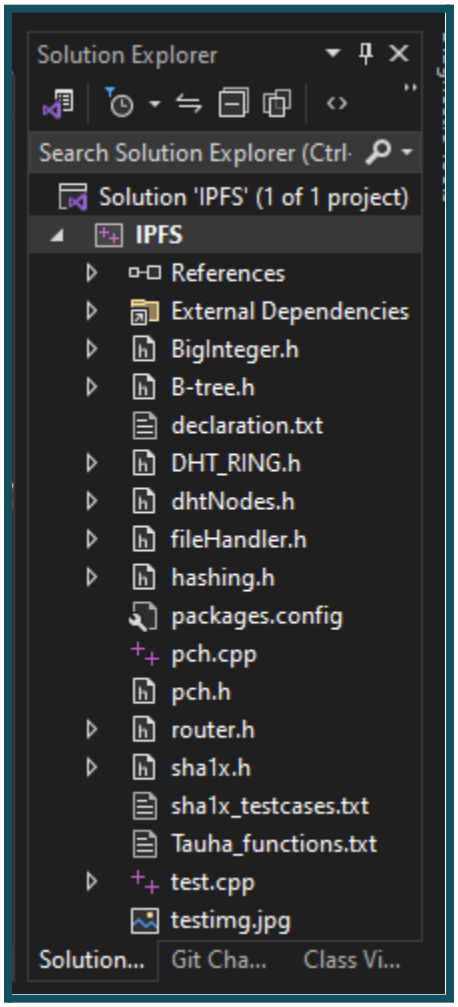

Fig1: Files in the Visual Studio Community 2022 c/c++ gtest project

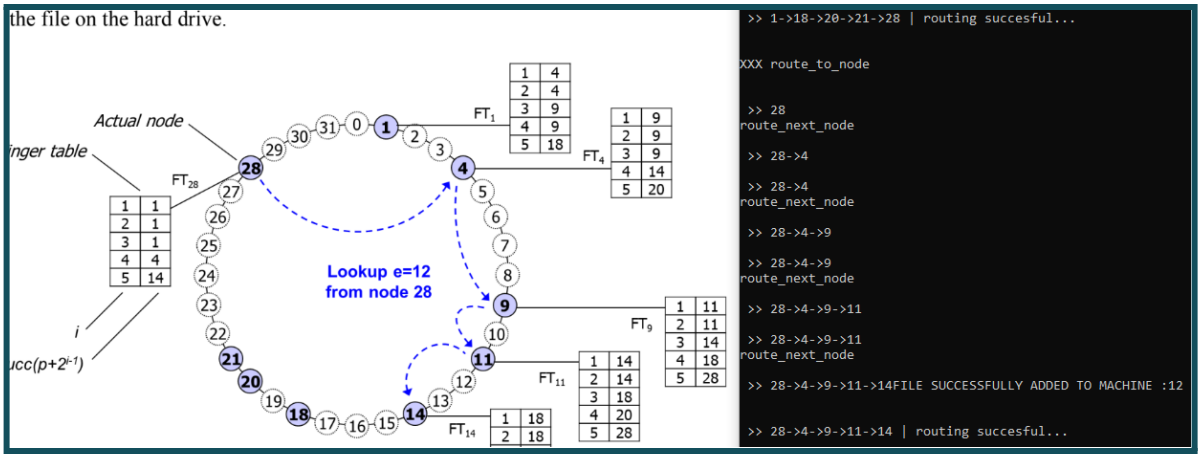

Fig2: Running Given routing scenario (fig5 of FinalProject_final.pdf)

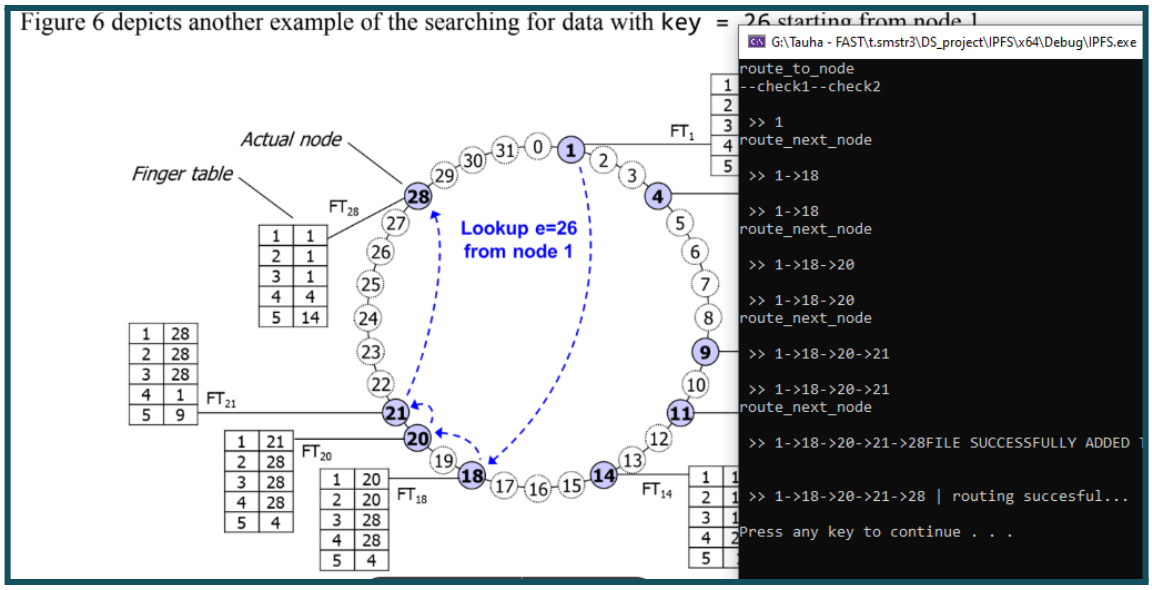

Fig3: Running Given routing scenario (fig6 of FinalProject_final.pdf)

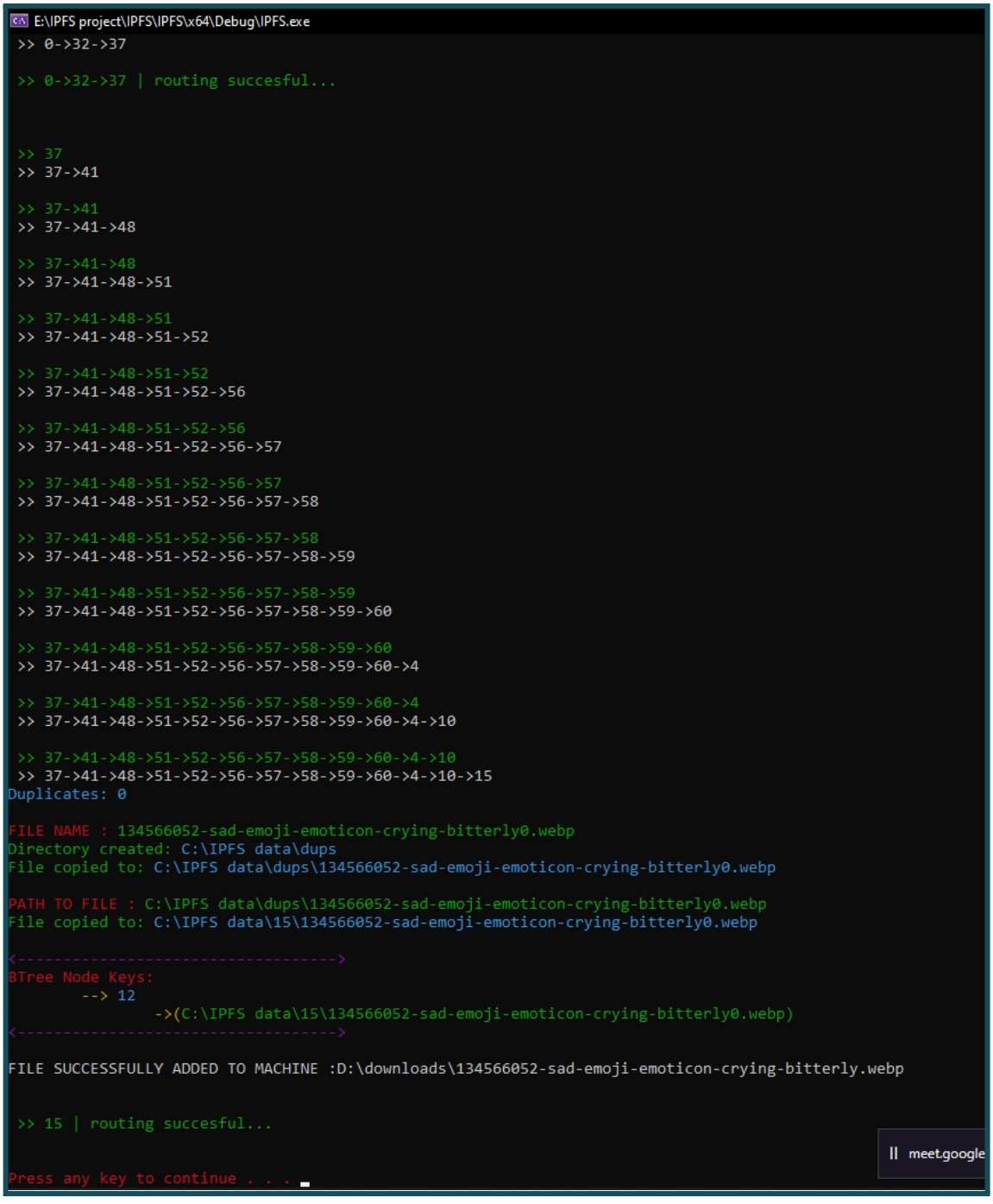

Fig4: Running a routing file insertion

```
 >> 37
 >> 37->41

 >> 37->41
 >> 37->41->48

 >> 37->41->48
 >> 37->41->48->51

 >> 37->41->48->51
 >> 37->41->48->51->52

 >> 37->41->48->51->52
 >> 37->41->48->51->52->56

 >> 37->41->48->51->52->56
 >> 37->41->48->51->52->56->57

 >> 37->41->48->51->52->56->57
 >> 37->41->48->51->52->56->57->58

 >> 37->41->48->51->52->56->57->58
 >> 37->41->48->51->52->56->57->58->59

 >> 37->41->48->51->52->56->57->58->59
 >> 37->41->48->51->52->56->57->58->59->60

 >> 37->41->48->51->52->56->57->58->59->60
 >> 37->41->48->51->52->56->57->58->59->60->4

 >> 37->41->48->51->52->56->57->58->59->60->4
 >> 37->41->48->51->52->56->57->58->59->60->4->10
Duplicates: 0

FILE NAME : 15162444127840.jpeg
Directory created: C:\IPFS data\dups
File copied to: C:\IPFS data\dups\15162444127840.jpeg

PATH TO FILE : C:\IPFS data\dups\15162444127840.jpeg
File copied to: C:\IPFS data\10\15162444127840.jpeg

<------------------------------------>
BTree Node Keys:
        --> 5
                ->(C:\IPFS data\10\0001-03190.mp4)
        --> 6
                ->(C:\IPFS data\10\15162444127840.jpeg)
<------------------------------------>

FILE SUCCESSFULLY ADDED TO MACHINE :D:\downloads\1516244412784.jpeg


 >> 10 | routing succesful...


Press any key to continue . . .
```

Fig5: Running a routing file insertion

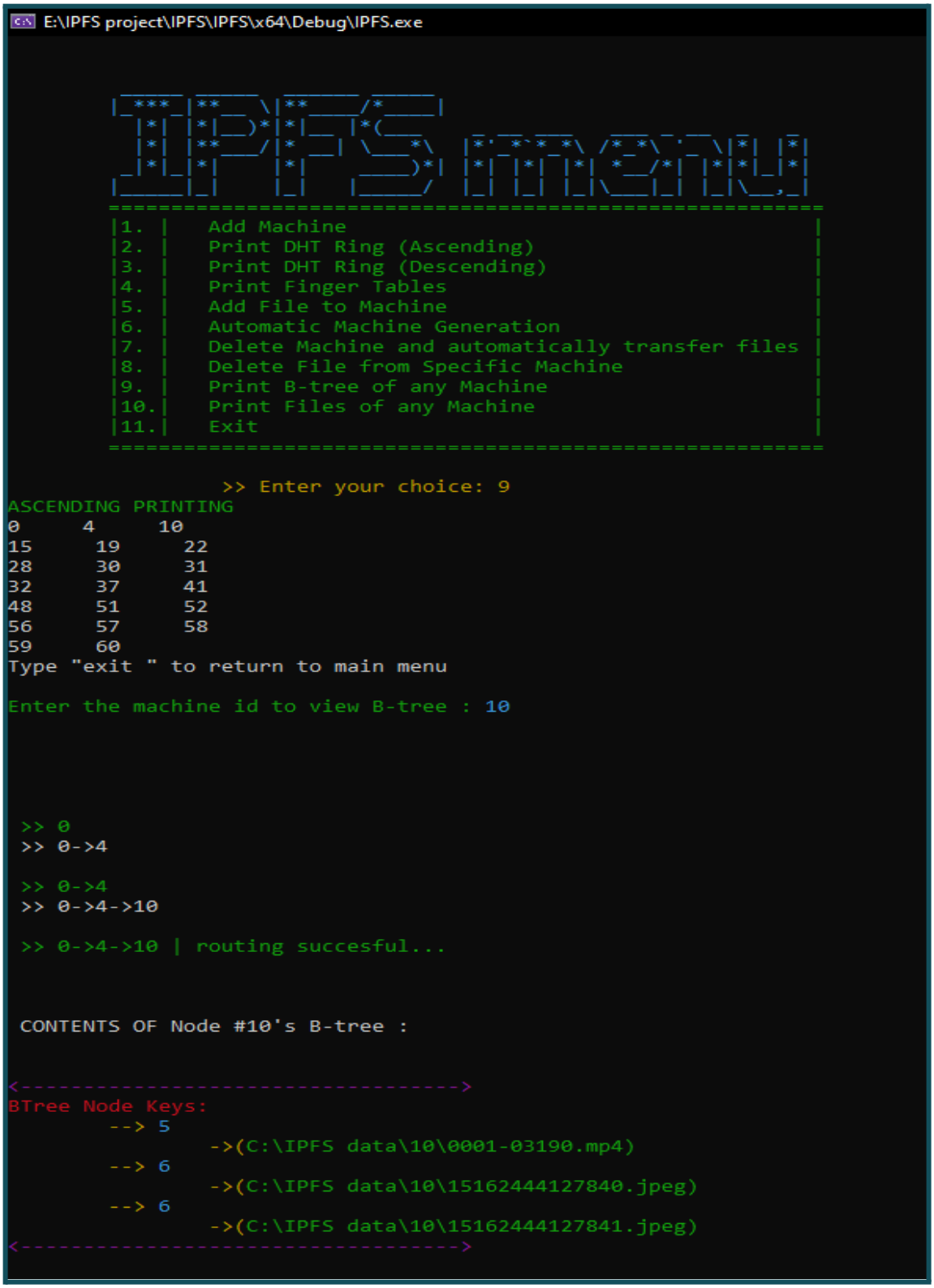

Fig6: Running a routing file insertion

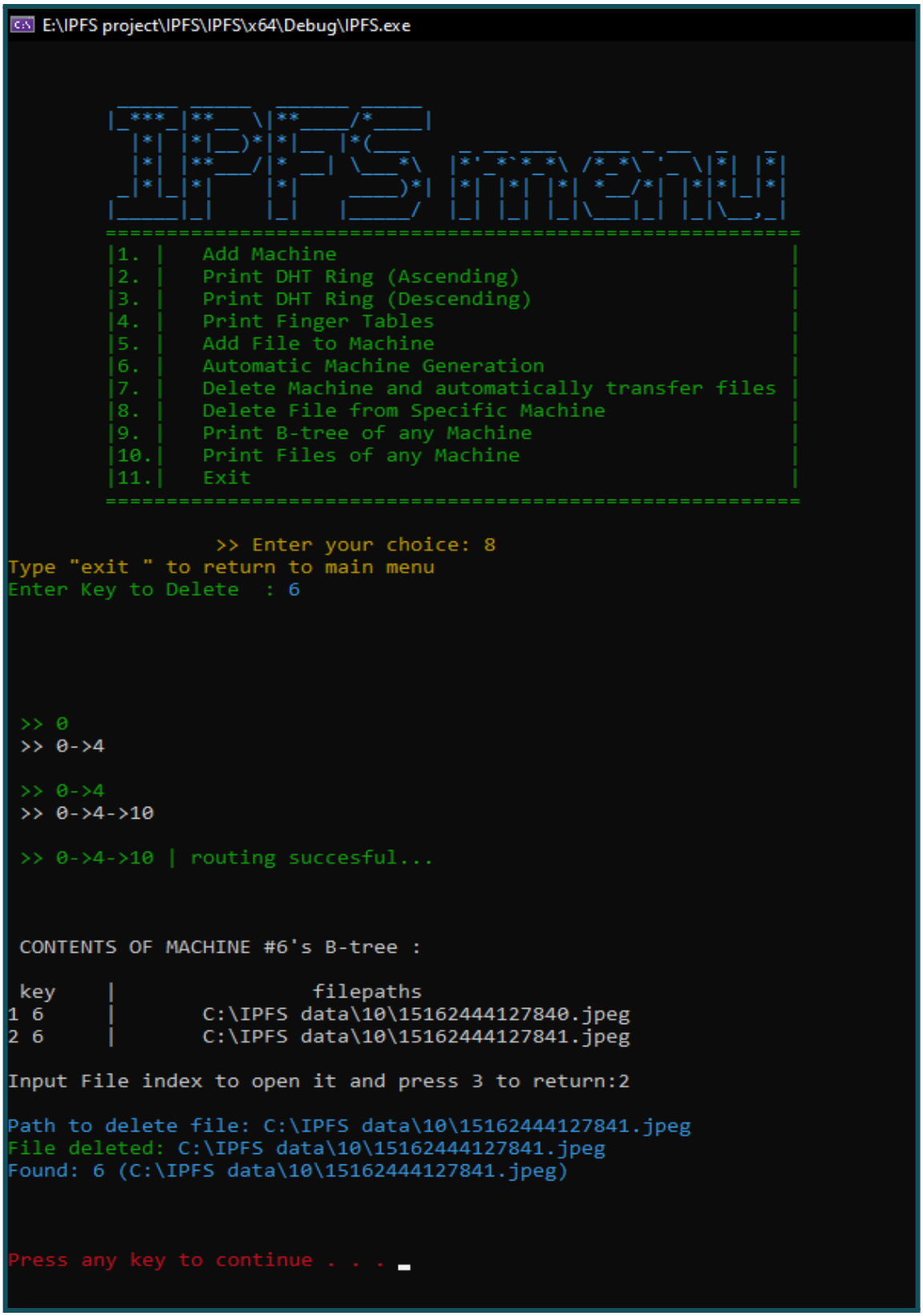

Fig7: Running a file deletion

Contributions

Abdullah Zubair	Saffi Muhammad Hashir	Tauha Imran
Constructed B-Tree.h: Applied universal algorithms for addition and deletion for Keys in their respective nodes and their depth-first-balancing. Entirely Iterative. **Splitting of B-trees:** Created functions which are called when adding a machine to transfer file content from one machine to the next. **Merging of B-trees:** Function calls to merge respective B-trees of Nodes when Machines are deleted from the system. **System Integrations of B-Trees:** Collected and Combined locations were to call Btree functions in the DHT_ring and Dht_nodes files. **Project Direction:** Helped debug and direct my colleagues' work under the mandate of the Project explanation file. **User Interface design:** Design for User interface, colours , sound and aesthetics **Making Documentation:** Design and making Project documentation, and picture formatting	**DHT_ring.h:** Implementation of a Distributed Hash Table (DHT) using a ring topology. **Dht_nodes.h:** Definitions for nodes in the DHT, crucial for decentralised storage and retrieval. **Biginteger.h:** Handling large integer arithmetic, likely for cryptographic operations or unique identifiers. **File and Folder Management:** Functions for decentralised file and folder operations in IPFS. **IPFS Logs:** Logging mechanisms for transparency and troubleshooting. **IPFS Router Logs:** Specific logging for IPFS router activities **Other Functions**: Various enhancements, contributing to the overall efficiency and functionality of IPFS.i.e Various functions in filehandler.h **User Interface design:** Design for User interface, colours , sound and aesthetics **Making Documentation:** Design and making Project documentation	**Router.h:** Made the routing class m the routing Algorithm and implemented it , in cooperating all the data structures in the the DHT_ring **Hashing.h:** Implemented all the hashing mechanisms incorporating the sha1x.h and file reading functions to generate accurate and usable hash values **sha1x.h:** Implemented the encryption algorithm according to the standards specified by the official SHA-1 documentation and library. **Linking Project files in VS:** Link routing with DHT_ring and Btree objects (combined header files and integrated them into objects of other classes/headers.) **Header file Management:** Managed header files, OOP aspects, and class constructors in the project to ensure smooth running of project **User Interface design:** Design for User interface, colours , sound and aesthetics **Additional Functions:** Made functions) *display_node_files(..)* & *display_node_Btree(..)* from DHT_RING.h Made *function read_file_from_path(...)* from filehandler.h **Making Documentation:** Design and making Project documentation, and picture formatting