Due Wednesday, April 22, 2020

Data Structures HW7

Do as much of this homework as makes sense for you! If only Level 1 makes sense, that's totally fine!

Level 1 - Heap

Implement a **min** heap! You can bulid this on an array/vector, or make your own tree structure.

Swift	**C++**
Heap <T>	**Heap**
func insert(element : T)	void insert(T)
func extractMin() -> T	T extractMin()
var isEmpty : Bool	bool isEmpty()

Level 2 - Priority Queue

Implement a priority queue. Feel free to use the Min Heap! Let 1 be the highest priority item (should bubble up to the top/front of the queue), the higher the value, the lower the priority.

Swift	**C++**
PriorityQueue <T>	**PriorityQueue**
func insert(element : T, withPrority : Int)	void insert(T, int)
func pullHighestPriority() -> T	T pullHighestPriority()
var isEmpty : Bool	bool isEmpty()

Level 3 - Huffman encoding

Do any of the sublevels here! Don't worry about doing more than one.

The Huffman tree is a tree with nodes that hold both a character (char/Character) and a number of occurrences (int/Int). A Huffman encoded string is a sequence of bits. Going bit by bit, and starting at the root, go left for each 0 and right for each 1 until a leaf node is hit. This leaf node holds the character for the string.

Level 3a:
Write a function that takes a string of 0s and 1s and a Huffman tree. This function should return the unencoded message as a string.

HuffmanTree - Swift	HuffmanTree - C++
static decode(message : String, withTree : HuffmanTree) -> String	static std::string decode(const std::string, const HuffmanTree* const)

Level 3b:
Similar to Level 1, but instead of a string of 0s and 1s, take an int which holds the binary representation of the encoded string. Hint: Use Bitwise Operators

HuffmanTree - Swift	HuffmanTree - C++
static decode(binaryMessage : [UInt8], withTree : HuffmanTree) -> String	static std::string decode(const vector<char>, const HuffmanTree* const)

Level 3c:
Write a function that takes a string and makes a Huffman tree for it, and encodes it. Write a second function that takes the tree and the encoded message and decodes it. EncodedString is a struct (C++), or a tuple (Swift) that holds a tree and a message stored in std::vector<char> for C++, or [UInt8] for Swift.

HuffmanTree - Swift	HuffmanTree - C++
static encode(message : String) -> EncodedString	static EncodedString encode(const std::string message)
static decode(encodedMessage : EncodedString) -> String	static std::string decode(EncodedString encodedMessage)