

Topic	Encryption - Algorithms	
Class Description	Students will face a decryption challenge where they will have to decode the secret message in real time.	
Class	C-229	
Class time	45 mins	
Goal	<ul style="list-style-type: none"> Enhancing the algorithms Finding the message in real time 	
Resources Required	<ul style="list-style-type: none"> Teacher Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Visual Studio Code Student Resources: <ul style="list-style-type: none"> Laptop with internet connectivity Earphones with mic Notebook and pen Visual Studio Code 	
Class structure	Warm-Up Student - led Activity 1 Wrap-Up	10 mins 30 mins 5 mins
WARM UP SESSION - 10mins		
Teacher Action		Student Action
<i>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</i>		ESR: Hi, thanks, yes, I am excited about it!

Q&A Session	
Question	Answer
<p>In computer technology, what do we say if we hide data in files/audio files?</p> <p>A. Cryptography B. Steganography C. Encrypt D. Decrypt</p>	B
<p>What do you mean by pixels?</p> <p>A. Small Dots that draws Picture B. Computer memory to save pictures C. Smallest part of picture D. None of the above</p>	A
STUDENT-LED ACTIVITY - 30mins	
Student Initiates Screen Share	
<p align="center"><u>ACTIVITY</u></p> <ul style="list-style-type: none"> • Understanding about WebRTC and it's functions • Fetching the audio and the video for the chat app from the user's browser 	
Teacher Action	Student Action
<p><i>This class includes a challenge we want the students to solve with minimum help. Try to give them appropriate time to think and help them come to the solution themselves.</i></p>	
<p>In the last few classes, we have learnt about encryption, what it is and how it is done. We learnt about different techniques of encryption.</p>	

<p>In today's class, we will be doing something new. We will understand the importance of algorithm building while trying to decode/decrypt a message.</p> <p>We will be having the following 3 things -</p> <ol style="list-style-type: none"> 1. A word list - with a total of 7,779 words from a dictionary. 2. An anagram 3. An MD5 encrypted message that we have to decode. <p>Before we begin, let's understand what an Anagram is!</p>	
<p>An anagram is a word or a phrase formed by rearranging the letters of a different word or phrase, typically using the original letters.</p> <p>It's basically like a jumbled sentence where the characters from one sentence are rearranged to form another.</p> <p>Let's look at an example -</p> <p>The sentence/phrase - "THE OIL PINKY" - if rearranged, will make - "I LIKE PYTHON"</p> <p>We all have solved jumbled letters in our childhood! Did you like it?</p>	<p>ESR: Varied!</p>
<p>Now here, you can also observe that since there were 2 spaces used in the anagram, we knew that the original phrase was 3 words long.</p> <p>Similar to this, we are given an anagram - "who outlay thieves" and we are given an encrypted message which we have to decode.</p> <p>From the anagram "who outlay thieves", can you tell me how many words would be there in the original phrase?</p>	<p>ESR:</p>

<p>Great! Now, how do you think we should solve this. Remember that we also have a word list with us.</p> <p>We have some boilerplate code with the logic written for us already! Let's clone it.</p> <p><i>Teacher refers to Teacher Activity 1</i></p>	<p>3!</p> <p>ESR: We can try different combinations of 3 words one by one, and encrypt them to check if it matches and we will find the original phrase.</p> <p><i>Student refers to Student Activity 1</i></p>
<p>Let's go through this code once, before running it.</p> <p>First, we have imported some libraries -</p>	
<pre>import hashlib from itertools import permutations</pre>	
<p>Trying out different combinations of n number of words is called permutations in Math!</p> <p>There is a function already that will do the thing for us, so let's not worry about that.</p> <p>Next, we have some initial code at the end of the file -</p>	
<pre>hash = '13b382e1a2f8e22535b4730d78bc8591' answer = find_hash(hash) print(f"Collision! The word corresponding to the given hash is '{answer}'")</pre>	

Here, we have written down the md5 encrypted message given to us. We are then passing this into a function called **find_hash()** and printing the answer.

Inside the function **find_hash()** -

```
def find_hash(original_hash):  
    word_file = open("words.txt", "r")  
    word_file = list(word_file)  
  
    anagram = "who outlay thieves"  
    words = anagram.count(' ')  
    words += 1  
  
    char_list = list(set(anagram))  
  
    if ' ' in char_list:  
        char_list.remove(' ')  
  
    final_words = []  
  
    #Student Activity
```

Here, we first load the word file - **words.txt** and then we define the anagram that we have.

We count the number of spaces in it “ ” and deduce the total words our original phrase should have.

Next, we use the **set()** function to take out the unique alphabets used in our anagram, and convert it into a **list()**.

Finally, we remove the space from our **char_list** and have an empty list call **final_words**

After this -

```
for elem in permutations(final_words, words):
    hash_elem = " ".join(elem)

    #Student Activity

    m = hashlib.md5()
    m.update(hash_elem.encode('utf-8'))
    word_hash = m.hexdigest()

    if word_hash == original_hash:
        return hash_elem
```

We have a for loop, where we are using the **permutations()** function that will give us a list of 3 words.

We join them with a space " " with the **join()** function, and encrypt it with **md5**.

Finally, we are checking if the current's phase md5 hash is the same as the hash we received in the problem statement, and return the answer if it matches.

Looks good, doesn't it?

ESR:
Varied!

Now let's think about it.

The total number of possible 3 word combinations in a word list consisting of 7,779 words is **470,547,867,174**, which is a

<p>huge number even to read!</p> <p>It's almost 470.5 billion combinations we are looking at. If let's say, our computer can perform 1 million calculations per second, even then our computer will take close to 1 year to go through all the 470.5 billion possibilities!</p> <p>It's not ethical to wait that long, is it?</p> <p>Here comes the challenge then. You need to think of an algorithm that can solve this within seconds!</p> <p>It's possible, you only need to think of a way!</p>	<p>ESR: It's not!</p>
<p>What do you think should be our first step?</p> <p>I think the first step should be to reduce the number of words we are dealing with. Can you think of a way on how we can remove words?</p> <p><i>Give student time to think for a couple of minutes</i></p> <p>We can check all the words in our word list, and eliminate any word that has a character that we don't have in our anagram.</p> <p>For example, why do we need any of the words which use an "R". We don't have it in our anagram, so we don't have it in the original phrase too!</p> <p>Therefore, the final list of words - final_words - should be a list of words that does not contain any alphabet our anagram doesn't have.</p> <p>Let's write the code for that -</p>	<p>ESR: Varied!</p> <p>ESR: Varied!</p>

Teacher helps the student in writing the code	Student writes the code
 <pre> for i in word_file: flag = False temp_word = i.replace('\n', '') temp_char = list(set(temp_word)) for i in temp_char: if i not in char_list: flag = True break if flag == False: final_words.append(temp_word) print(len(final_words)) </pre>	
<p>In this for loop, we are first iterating over all the words in word_file.</p> <p>Inside the loop, we have a flag False, which denotes that this word initially does not have any character that is not in our anagram.</p> <p>Next, we take a list of all the unique alphabets used in this word with the set() and list() functions and save it into temp_char.</p> <p>Lastly, we are iterating over all the characters in the word, and we are checking if they are in the anagram or not.</p> <p>If they are not in the anagram, we are changing the flag to True and finally we are deciding based on the flag, if this word should go into the final_words list or not.</p>	

<p>This will significantly help us reduce the total number of words we are dealing with. As a result, out of 7,779 words, we are only left with 194!</p> <p>This will still have about 7,188,864 combinations in total.</p> <p>With this, now we can be certain that our program does not run for more than 10 seconds, but we can reduce this time even further. Any ideas?</p> <p>What is that one more similarity between our anagram and the original phrase?</p> <p>The length of both the anagram and the phrase should be the same.</p> <p>Let's try to enhance our algorithm further by comparing the lengths of the anagram and the phrase, before encrypting it with md5 and comparing.</p> <p><i>Teacher helps the student in writing the code</i></p>	<p>ESR: Varied!</p> <p>ESR: Varied!</p> <p><i>Student writes the code</i></p>
---	---

```
for elem in permutations(final_words, words):
    hash_elem = " ".join(elem)

    if len(hash_elem) != len(anagram):
        continue

    m = hashlib.md5()
    m.update(hash_elem.encode('utf-8'))
    word_hash = m.hexdigest()

    if word_hash == original_hash:
        return hash_elem
```

With this extra line of code, we are able to ensure that we only encrypt those phrases that could potentially be the answer. This will reduce the computation time of this code even further.

Now, let's run the program and see if we find the message!

We can use the following command on terminal/command prompt -

python main.py





Teacher helps the student in running the program

Student runs the program

Collision! The word corresponding to the given hash is 'whitehat loves you'

With this, we have found the answer. You can see how powerful an algorithm could be, and how we managed to reduce a year's work to less than 10 seconds with just simple logic and a few lines of code.

<p>The message is “whitehat loves you”</p> <p>Different hackers / software engineers keep devising such algorithms to decrypt/decipher such encrypted messages to gain personal information!</p> <p>In the next class, we will be learning about viruses, how they work and can be created!</p>	
Teacher Guides Student to Stop Screen Share	
WRAP UP SESSION - 5 Mins	
Quiz time - Click on in-class quiz	
Question	Answer
<p>Which function is used to remove an element from a list?</p> <ol style="list-style-type: none"> 1. pop() 2. append() 3. remove() 4. delete() 	C
<p>If you want to get a list of unique elements from a list (arr), which of the following series of functions can be used?</p> <ol style="list-style-type: none"> 1. list(set(arr)) 2. set(list(arr)) 3. list(list(arr)) 4. set(set(arr)) 	A
<p>If you wanted to get permutations of <i>n</i> words from a list <i>arr</i>, how can you do that?</p> <ol style="list-style-type: none"> 1. permutations(arr, n) 2. permutations(arr(n)) 3. permutations(n, arr) 4. permutations(n(arr)) 	C

End the quiz panel	
FEEDBACK <ul style="list-style-type: none"> Appreciate the students for their efforts in the class. Ask the student to make notes for the reflection journal along with the code they wrote in today's class. 	
Teacher Action	Student Action
<p>You get Hats off for your excellent work!</p> <p>In the next class</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div> <div>Creatively Solved Activities  +10</div> <div>Great Question  +10</div> <div>Strong Concentration  +10</div> </div>
Project Discussion	
<p>Teacher Clicks</p> <p></p>	
ADDITIONAL ACTIVITIES	
<p>Additional Activities</p> <p><i>Encourage the student to write reflection notes in their reflection journal using markdown.</i></p> <p>Use these as guiding questions:</p>	<p><i>The student uses the markdown editor to write her/his reflections in the reflection journal.</i></p>

<ul style="list-style-type: none"> • What happened today? <ul style="list-style-type: none"> ○ Describe what happened. ○ The code I wrote. • How did I feel after the class? • What have I learned about programming and developing games? • What aspects of the class helped me? What did I find difficult? 	
---	--

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-C229-Teacher-Boilerplate-Code
Teacher Activity 2	Reference Code	https://github.com/pro-whitehatjr/PRO-C229-Reference-Code
Student Activity 1	Boilerplate Code	https://github.com/pro-whitehatjr/PRO-C229-Teacher-Boilerplate-Code