**Covid19 Helpline Webapp**

An Industrial Internship Report

*submitted by*

**Mihir Srivastava**

**18BCI0214**

*in partial fulfilment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE ENGINEERING WITH SPECIALIZATION IN INFORMATION SECURITY**



**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

MAY 2021

**DECLARATION BY THE CANDIDATE**

I hereby declare that the Industrial Internship report entitled **“Covid 19 Helpline Webapp”** submitted by me to Vellore Institute of Technology, Vellore in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in **Computer science and engineering with specialization in information security** is a record of bonafide industrial training undertaken by me under the supervision of **HCL Technologies Ltd**. I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Signature of the student

Name: Mihir Srivastava

Reg. Number: 18BCI0214



**School of Computer Science and Engineering**

**BONAFIDE CERTIFICATE**

This is to certify that the Industrial Internship report entitled **COVID19 HELPLINE WEBAPP** submitted by **Mihir Srivastava(18BCI0214)** to Vellore Institute of Technology, Vellore in partial fulfilment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer science and engineering with specialization in information security** is a record of bonafide Industrial Internship undertaken by him/her under my supervision. The training fulfils the requirements as per the regulations of this Institute and in my opinion, meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**SUPERVISOR**

Date: Date:

**Internal Examiner (s) External Examiner (s)**

**CERTIFICATE**

**ACKNOWLEDGEMENT**

I would like to thank Alok Bhatt(Data Scientist) at HCL Technologies for guiding and training me for my project. His expertise on Django and NLP was fruitful in my learning experience and helped me building my project on a much better level. The internship surely helped me to have a better understanding for my future coursework and career.

Place : Vellore Mihir Srivastava

**(Name of the Student)**

Date :

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
|  | Table of Figures |  |
| 1 | Synopsis | 1 |
| 2 | About the Organization | 1 |
| 3 | The skillset of the student before the training | 2 |
| 4 | Knowledge acquired from the training | 2 |
| 5 | TOPIC wise Descriptions | 5 |
|  | 5.1 Introduction to DJANGO | 5 |
|  | 5.1.1 Django Documentation | 6 |
|  | 5.1.2 DBMS test app | 6 |
|  | 5.2 PROJECT INITIALIZATION | 7 |
|  | 5.2.1 Resource API | 7 |
|  | 5.2.2 Home page and news display | 8 |
|  | 5.2.3 Weather API | 8 |
|  | 5.3 Covid Tracking | 9 |
|  | 5.3.1 Government of India Github data collection | 9 |
|  | 5.4 Twitter Data Collection | 9 |
|  | 5.4.1 Tweepy | 9 |
|  | 5.4.2 Twitter data collection | 10 |
|  | 5.5 SENTIMENT ANALYSIS OF THE TWITTER DATA | 10 |
|  | 5.5.1 Representation of polarity using chart js | 11 |
|  | 5.5.2 Using NLP determining polarities of tweets | 11 |
|  | 5.6 KNN CLASSIFIER TO CHECK ZONE OF A PERSON | 13 |
|  | 5.6.1 KNN | 13 |
|  | 5.6.2 Histogram to represent KNN output | 13 |
|  | 5.7 RESTAPIS | 13 |
| 6 | Comparison of competency levels before and after the training | 15 |
| 7 | Appendices | 16 |
|  | Appendix 1 (Code snippets) | 16 |

**TABLE OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **Page No.** |
| 1.1 | Django Documentation | 6 |
| 1.2 | Django DBMS test app | 7 |
| 2.1 | Resources | 7 |
| 2.2 | News related to covid | 8 |
| 2.3 | API Practice | 8 |
| 3.1 | Covid Tracking Problem | 9 |
| 4.1 | TWITTER Forms | 10 |
| 5.1 | Sentiment analysis using chart js and nlp | 11 |
| 6.1 | KNN implementation and Histogram visualization | 13 |
| 7.1 | RESTAPI of city | 14 |
| 7.2 | RESTAPI of state | 14 |

**SYNOPSIS**

My Internship was based on web development and data science. My initial objective was to create a webapp that would help to cater the needs of people and provide them with functionalities to tackle covid. My mentor recommended me that some of my work will include machine learning activities and application and hence Django was the backend framework used in my project.

My Tasks included the following : -

1. COVID info and other news using news api
2. Display resources available for covid using Government of India Github API
3. Display a covid tracker using API
4. Creating a Visualization for the zone of location using KNN algorithm.
5. Scraped twitter data for COVID related topics using Tweepy.
6. Used that twitter data and created an NLP model which would tell the polarity of those tweets.
7. Created RESTAPI’s for others to use which would give them the data related to a particular state or city.

**ABOUT THE ORGANISATION**

HCL Technologies is an Indian multinational information technology (IT) services and consulting company, headquartered in Noida, Uttar Pradesh, India. It is a subsidiary of HCL Enterprise. Originally a research and development division of HCL, it emerged as an independent company in 1991 when HCL entered into the software services business. The company has offices in 32 countries including United Kingdom, United States, France, and Germany with a worldwide network of R&D, "innovation labs" and "delivery centers", over 159,000 employees and its customers include 250 of the Fortune 500 and 650 of the Global 2,000 companies.

It operates across sectors including aerospace and defense, automotive, banking, capital markets, chemical and process industries, energy and utilities, healthcare, hi-tech, industrial manufacturing, consumer goods, insurance, life sciences, manufacturing, media and entertainment, mining and natural resources, oil and gas, retail, telecom, and travel, transportation, logistics & hospitality.

HCL Technologies is on the Forbes Global 2000 list. It is among the top 20 largest publicly traded companies in India with a market capitalization of $21.5 billion as of May 2019. As of July 2020, the company, along with its subsidiaries, had a consolidated annual revenue of ₹71,265 crore (US$10 billion).

Enterprises across industries stand at an inflection point today. In order to thrive in the digital age, technologies such as analytics, cloud, IoT, and automation occupy center stage. In order to offer enterprises the maximum benefit of these technologies to further their business objectives, HCL offers an integrated portfolio of products and services through three business units. These are IT and Business Services (ITBS), Engineering and R&D Services (ERS), and Products and Platforms (P&P).

ITBS enables global enterprises to transform their businesses via Digital Foundation, our modernized infrastructure stack built around hybrid cloud, software-defined networks, the digital workplace, and other elements; Digital Business, a combination of our application services and consulting capabilities; and Digital Operations, a three-pronged setup for modernized and efficient operations at enterprise level.

ERS offers engineering services and solutions in all aspects of product development and platform engineering.

Under P&P, HCL provides modernized software products to global clients for their technological and industry-specific requirements.

Their holistic Mode 1-2-3 strategy forms the backbone of these three business units to help enterprises navigate the digital age with ease. It is the core aspect of our ‘Digital Enterprise 4.0’ focus – aimed at offering holistic services to our clients to meet the technology needs of their present while readying them to be future-ready.

The company’s DNA of grassroots innovation, its ingrained culture of co-innovation, and its tradition of going far beyond what is expected, to create customer value, clearly differentiates it and gives it a distinct advantage in creating value for businesses in the digital and connected world.

The tech stacks of HCL web based technologies are abundant like :-

* Microsoft Azure
* Azure SQL
* LogMeIn Rescue
* Datadog
* Jira Service Desk
* IBM HACP
* JIRA Software
* Greenplum DB
* ManageEngine ServiceDesk Plus
* SAS DataFlux
* Bandwidth
* AWS Certificate Manager
* SysAid
* IBM POWER Systems

Whereas they work with ample of other departments like :-

* IT, Server & Network Monitoring
* Devops and Development
* Business Intelligence & Analytics
* Platform and Storage
* Communications
* Computer networks
* IT Security
* Operations Software
* Testing and QA
* Business Intelligence & Analytics
* Health tech
* IT Security
* Product and Design
* Ecommerce Technology

**THE SKILLSET OF THE STUDENT**

In the second semester of my college I was taught Problem Solving and Object-Oriented Programming in the CSE1002 course in C,C++ and Data Structures and Algorithms in the CSE2003 course. In the CSE1002 course I was given a brief introduction of programming and object oriented programming. From there I was keen on exploring more and more and starting with Web Development. Due to the Course Internet and Web Programming in the CSE3002, I was able to start coding in html, css and js. Also due to the exposure of PHP, I decided to make various webbased projects. Other than that I also dived deep into data science and learnt about automation, data scraping and collection, Exploratory Data Analysis etc.

Using these skill sets I’ve tried making few personal projects which also helped me in my internship.

**KNOWLEDGE ACQUIRED FROM THE TRAINING**

I learnt about different tech stacks like Django, RESTAPIs, Machine Learning Classifier, Git,

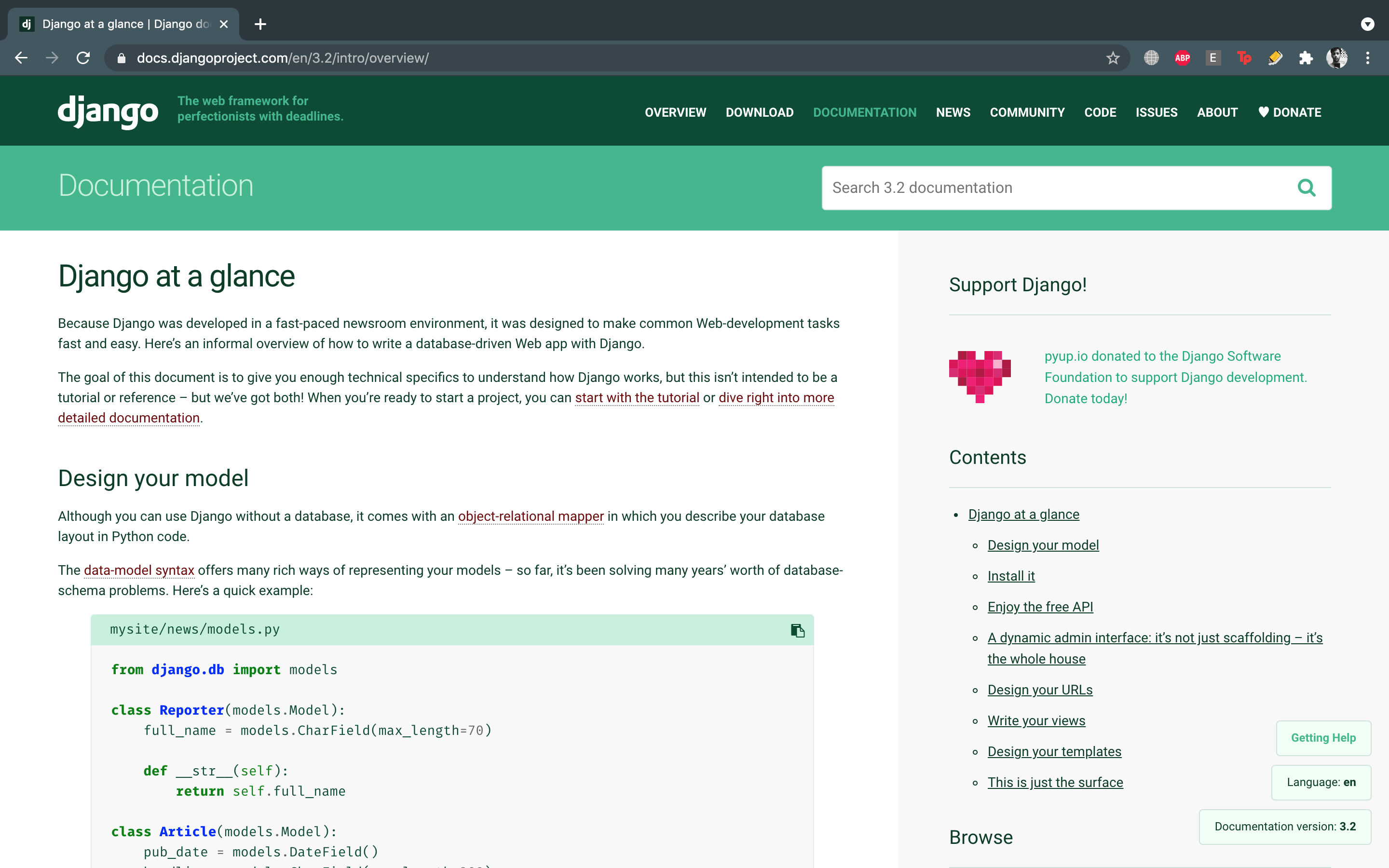
Heroku deployment, using sqlite and postgres databases, Data Visualization and many other technologies.

**DESCRIPTIONS**

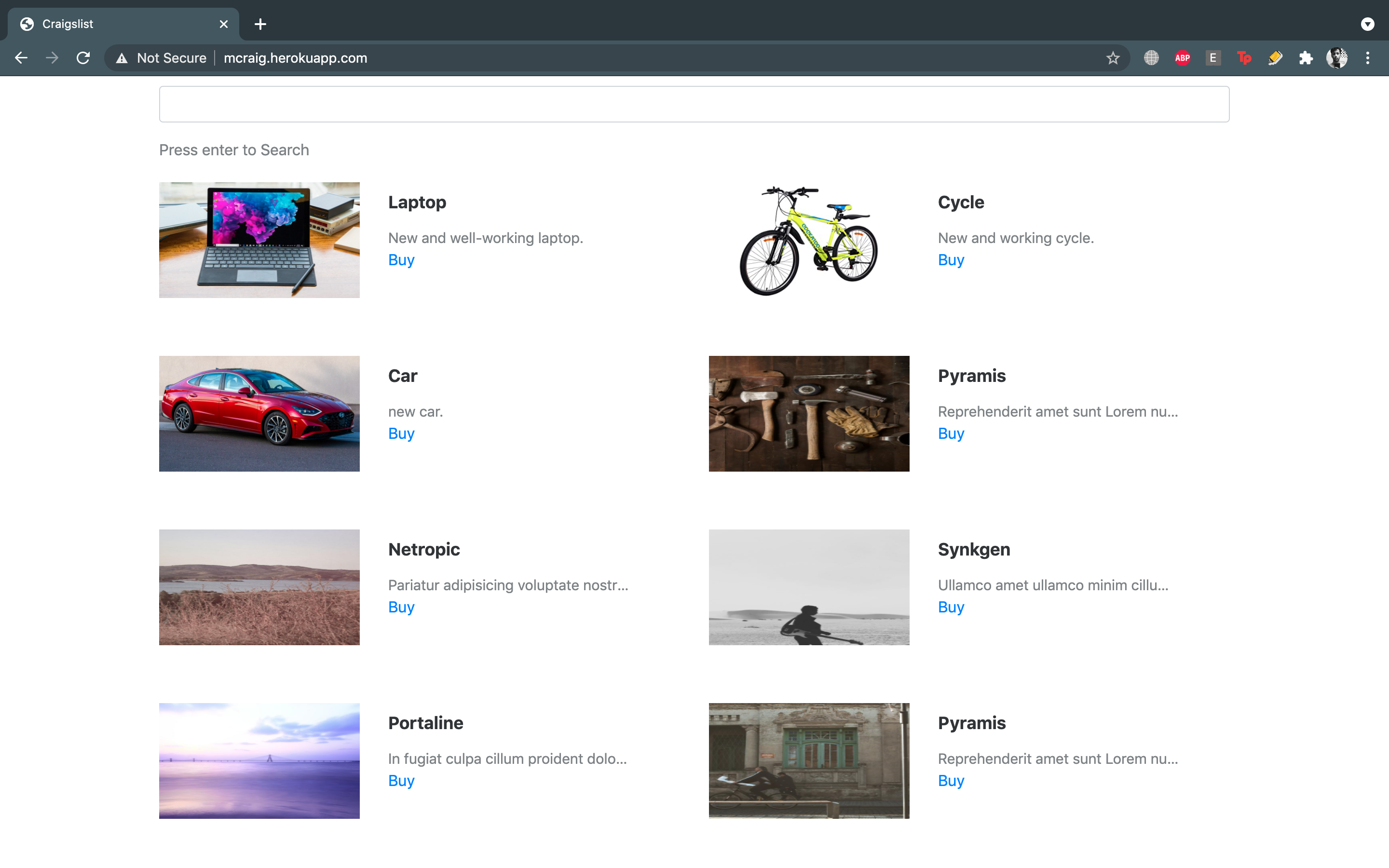
This section contains details about all the topics that I studied and implemented in this internship.

**INTRODUCTION TO DJANGO**

1. Here are few things my mentor asked me to do in order to learn Django properly.
2. Read the Mozilla guide to html form, and the official Django tutorial does not give this important topic enough emphasis.
3. Did the official Django tutorial, twice. The first time I struggled, the second time the key concepts will become more clear https://docs.djangoproject.com/en/2.1/intro/tutorial01/
4. Mozilla use Django in production Complete the Mozilla Django tutorial https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django This is doable in 2-3 days, faster if you really push it
5. Read about HTTP: https://developer.mozilla.org/en-US/docs/Web/HTTP
6. Mozilla is the best site for all things web. Some people love videos, the problem is with a video it is easy to miss something, and with Django a single mistake can be a showstopper and the cause may not be clear for a newcomer. The video series Coding for Entrepreneurs, may be helpful too. https://www.codingforentrepreneurs.com .



Other than doing the documentation applications, I also created an ecommerce webapp which used Django and SQLite3 and PostGresSQL.



**WEEK 2: ALGORITHMIC WARMUP**

This week, we learned that programs based on efficient algorithms can solve the same problem billions of times faster than programs based on naïve algorithms. We also learned how to estimate the running time and memory of an algorithm without even implementing it. Armed with this knowledge, we will be able to compare various algorithms, select the most efficient ones, and finally implement them as our programming challenges.

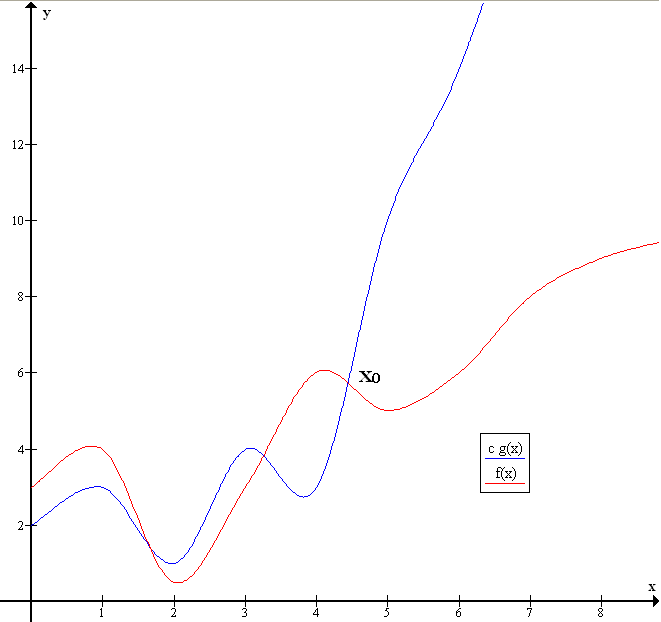
**Big O Notation**

We use big-Θ notation to asymptotically bound the growth of a running time to within constant factors above and below. Sometimes we want to bound from only above.

For example, although the worst-case running time of binary search is Θ(log⁡2n), it would be incorrect to say that binary search runs in Θ(log⁡2n) time in all cases. What if we find the target value upon the first guess? Then it runs in Θ(1) time. The running time of binary search is never worse than Θ(log⁡2n) but it's sometimes better.

It would be convenient to have a form of asymptotic notation that means "the running time grows at most this much, but it could grow more slowly." We use "big-O" notation for just such occasions.

If a running time is O(f(n)), then for large enough n, the running time is at most k⋅f(n) for some constant k. Here's how to think of a running time that is O(f(n)).



**Figure 2.1:** Graph of Big O

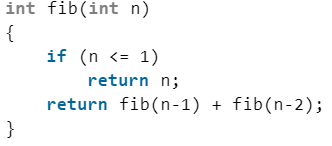
**Fibonacci number**

The Fibonacci numbers are the numbers in the following integer sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144…

In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation

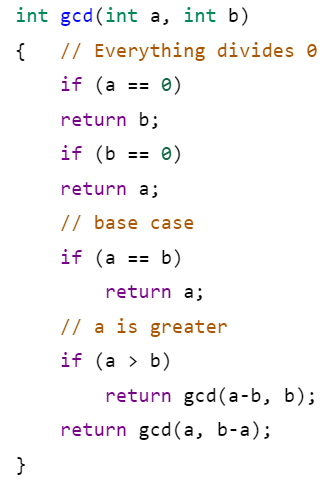
Fn = Fn-1 + Fn-2



**Figure 2.2:** Fibonacci Algorithm

**Greatest Common Divisor**

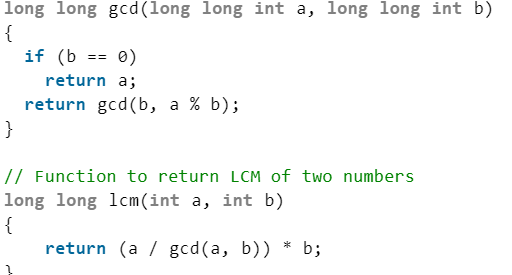
GCD (Greatest Common Divisor) or HCF (Highest Common Factor) of two numbers is the largest number that divides both of them.



**Figure 2.3:** GCD Algorithm

**Least Common Multiple**

LCM (Least Common Multiple) of two numbers is the smallest number which can be divided by both numbers. For example, LCM of 15 and 20 is 60, and LCM of 5 and 7 is 35.



**Figure 2.4:** LCM Algorithm

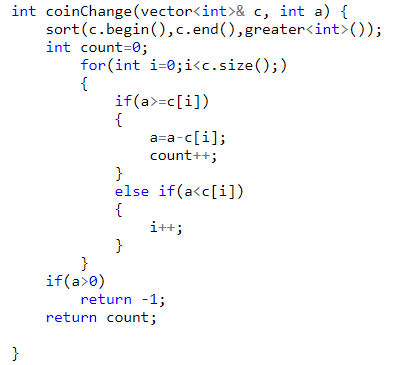
Hence we warmed up our concepts of a few of the popular problems and algorithms in this module.

**WEEK 3: GREEDY ALGORITHMS**

A greedy algorithm is a simple, intuitive algorithm that is used in optimization problems. The algorithm makes the optimal choice at each step as it attempts to find the overall optimal way to solve the entire problem. Greedy algorithms are quite successful in some problems, such as Huffman encoding which is used to compress data, or Dijkstra's algorithm, which is used to find the shortest path through a graph. However, in many problems, a greedy strategy does not produce an optimal solution.

**Money Change**

A common intuition would be to take coins with greater value first. This can reduce the total number of coins needed. Start from the largest possible denomination and keep adding denominations while the remaining value is greater than 0.

****

**Figure 3.1:** Money Change Algorithm

**Majority Element**

Given an array nums of size n, return the majority element.

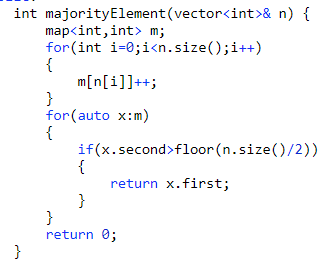
The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

This method is somewhat similar to Moore voting algorithm in terms of time complexity, but in this case, there is no need for the second step of Moore voting algorithm. But as usual, here space complexity becomes O(n).

In Hashmap(key-value pair), at value, maintain a count for each element(key) and whenever the count is greater than half of the array length, return that key(majority element).

Algorithm:

1. Create a hashmap to store a key-value pair, i.e. element-frequency pair.
2. Traverse the array from start to end.
3. For every element in the array, insert the element in the hashmap if the element does not exist as key, else fetch the value of the key ( array[i] ), and increase the value by 1
4. If the count is greater than half then print the majority element and break.
5. If no majority element is found print “No Majority element”



**Figure 3.2:** Majority Element Algorithm

Hence in this module we studied all about greedy algorithms and their applications.

**WEEK 4: DIVIDE AND CONQUER**

Divide and Conquer technique is helpful and how we can solve the problem with the DAC technique approach. This technique can be divided into the following three parts.

Divide: This involves dividing the problem into some sub problem.

Conquer: Sub problem by calling recursively until sub problem solved.

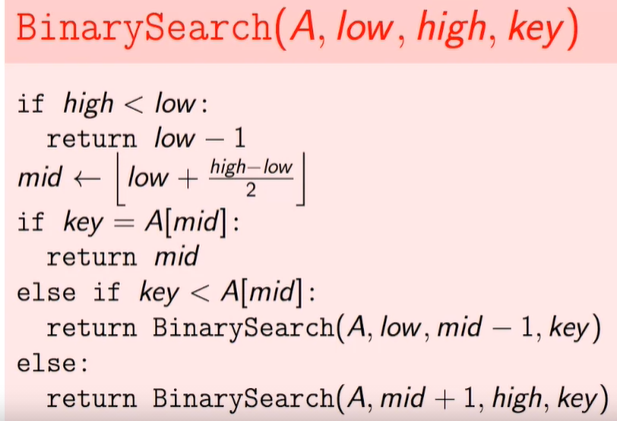
Combine: The Sub problem Solved so that we will find a problem solution.

This week I have learnt about a powerful algorithmic technique called Divide and Conquer. Based on this technique, we will see how to search huge databases millions of times faster than using naïve linear search. We will then apply the divide-and-conquer technique to design two efficient algorithms (merge sort and quicksort) for sorting huge lists, a problem that finds many applications in practice. Finally, we will show that these two algorithms are optimal, that is, no algorithm can sort faster.

**Binary Search**

Binary search is a fast search algorithm with run-time complexity of Ο(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

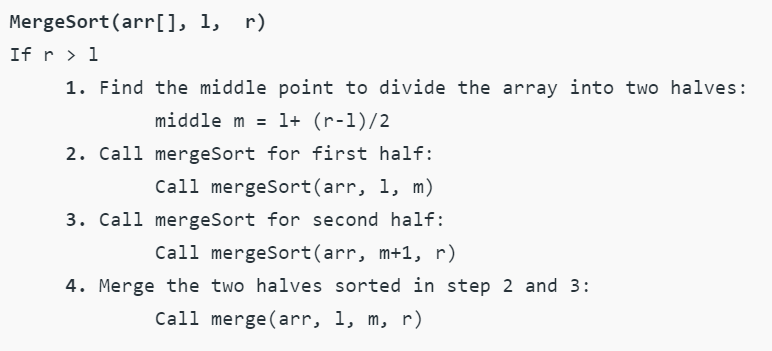
Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of the item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduces to zero.



**Figure 4.1:** Binary Search Algorithm

**Merge Sort**

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is a key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted subarrays into one.



**Figure 4.2:** Merge Sort Algorithm

**Quick Sort**

QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

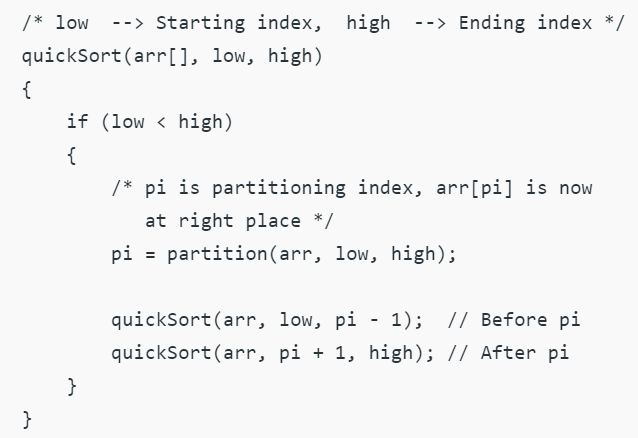
->Always pick the first element as pivot.

->Always pick last element as pivot (implemented below)

->Pick a random element as pivot.

->Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time.



**Figure 4.3:** Quick Sort Algorithm

Hence this module taught us all about divide and conquer algorithms and its applications and problems.

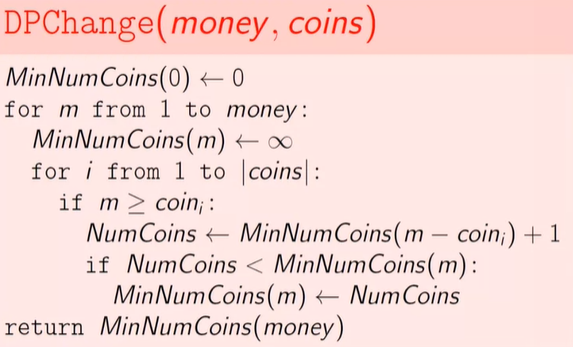
**WEEK 5: DYNAMIC PROGRAMMING - 1**

In this week I have learnt about the powerful algorithmic technique for solving many optimization problems called Dynamic Programming. It turned out that dynamic programming can solve many problems that evade all attempts to solve them using greedy or divide-and-conquer strategies.

**Coin Change**

Given an integer array of coins representing coins of different denominations and an integer amount representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

****

**Figure 5.1:** Coin change algorithm using Dynamic Programming

**Edit Distance**

The idea is to process all characters one by one starting from either the left or right sides of both strings. Let us traverse from the right corner, there are two possibilities for every pair of characters being traversed.

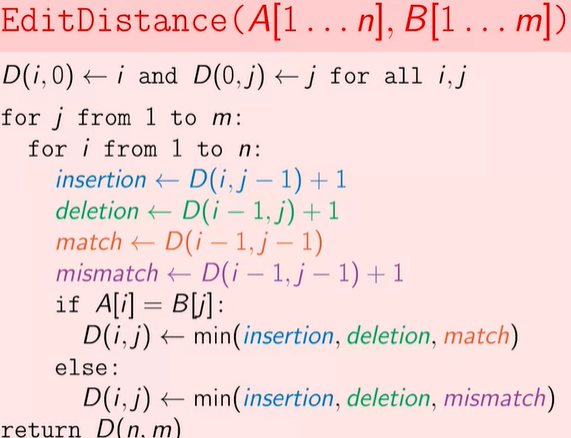
If the last characters of two strings are the same, there isn't much to do. Ignore last characters and count for remaining strings. So we recur for lengths m-1 and n-1.

Else (If the last characters are not the same), we consider all operations on ‘str1’, consider all three operations on the last character of the first string, recursively compute the minimum cost for all three operations and take a minimum of three values.

Insert: Recur for m and n-1

Remove: Recur for m-1 and n

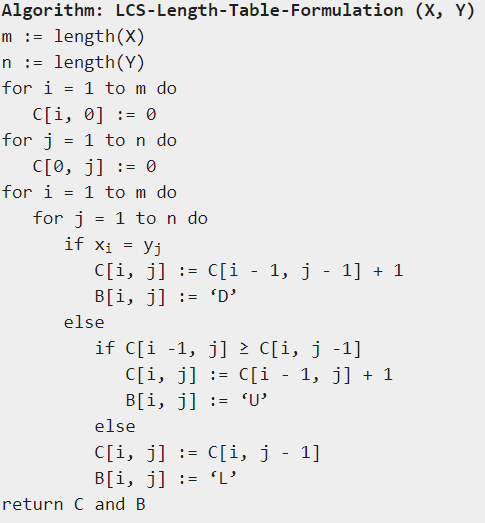
Replace: Recur for m-1 and n-1



**Figure 5.2:** Edit Distance Algorithm

**Longest Common Subsequence(LCS)**

Given two sequences, find the length of the longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”.



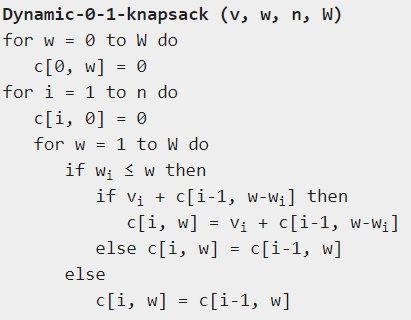
**Figure 5.3:** LCS Algorithm

**WEEK 6: DYNAMIC PROGRAMMING - 2**

This week, we continue to practice dynamic programming and solve more problems using it.

**0/1 Knapsack Problem**

Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. In other words, given two integer arrays val[0..n-1] and wt[0..n-1] which represent values and weights associated with n items respectively. Also given an integer W which represents knapsack capacity, find out the maximum value subset of val[] such that sum of the weights of this subset is smaller than or equal to W. You cannot break an item, either pick the complete item or don’t pick it (0-1 property).



**Figure 6.1:** Knapsack Algorithm

**Partition Equal Subset Sum**

Partition problem is to determine whether a given set can be partitioned into two subsets such that the sum of elements in both subsets is the same.

Examples:

arr[] = {1, 5, 11, 5}

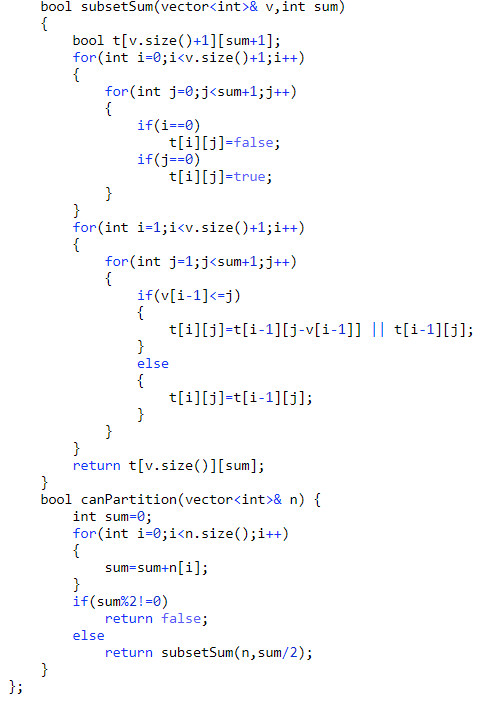
Output: true

The array can be partitioned as {1, 5, 5} and {11}

arr[] = {1, 5, 3}

Output: false

The array cannot be partitioned into equal sum sets.



**Figure 6.2:** Partition Algorithm

Thus from week 5 and week 6 we learnt about dynamic programming and solved many challenging problems using it.

**COMPARISON OF COMPETENCY LEVELS BEFORE AND AFTER THE TRAINING**

Before the training, I had studied algorithms and how, when and where to apply them. I also learned C++ programming and had practiced a few questions theoretically. But never got a chance to practically implement it. This course gave me that chance. Not only this it cleared many doubts regarding some extremely important algorithms like Divide & conquer, Greedy algorithm, Dynamic Programming, etc. This course gave me an opportunity to find optimal solutions to many challenging problems and compare with other solutions to find the best of them. This training provided me with hands-on experience in coding all major algorithms. Now after the training is completed, I am very confident about my algorithm skills and solving problems effectively. This training would be very helpful for my placement tests and interviews. In short, this training helped me alot in terms of coding skills, algorithm concepts and self-confidence.

**Appendix 1**

**My Code Snippets**

**1)Maximum Pairwise Product**

#include<bits/stdc++.h>

using namespace std;

int main()

{

long long int n;

cin>>n;

vector<long long int>v(n);

for(int i=0;i<n;i++)

{

cin>>v[i];

}

sort(v.begin(),v.end(),greater<int>());

cout<<v[0]\*v[1];

}

**2)COIN CHANGE Greedy**

#include<bits/stdc++.h>

using namespace std;

int main()

{

int m;

cin>>m;

vector<int>v;

v.push\_back(10);

v.push\_back(5);

v.push\_back(1);

int count=0;

for(int i=0;i<v.size();)

{

if(m>=v[i]){

m=m-v[i];

count++;

}

else{

i++;

}

}

cout<<count;

}

**3)Binary Search**

#include<bits/stdc++.h>

using namespace std;

int binary\_search(vector<int>&a,int key,int n)

{

int s=0;

int e=n-1;

while(s<=e)

{

int mid=(s+(e-s)/2);

if(a[mid]==key)

{

return mid;

}

else if(a[mid]>key)

{

e=mid-1;

}

else

{

s=mid+1;

}

}

return -1;

}

int main()

{

int n;

vector<int>a;

cin>>n;

for(int i=0;i<n;i++)

{

int input;

cin>>input;

a.push\_back(input);

}

int k;

cin>>k;

vector<int>v;

for(int i=0;i<k;i++)

{

int input;

cin>>input;

v.push\_back(input);

}

for(int i=0;i<k;i++)

{

cout<<binary\_search(a,v[i],n)<<" ";

}

}

**4)Majority Element**

#include<bits/stdc++.h>

using namespace std;

int majority\_element(vector<int>&v)

{

map<int,int>m;

for(int i=0;i<v.size();i++)

{

m[v[i]]++;

}

for(auto x:m)

{

if(x.second>ceil(v.size()/2))

{

return 1;

}

}

return 0;

}

int main() {

int n;

cin>>n;

vector<int>v;

for(int i=0;i<n;i++)

{

int input;

cin>>input;

v.push\_back(input);

}

cout<<majority\_element(v);

}

**5)Coin Change DP**

#include<bits/stdc++.h>

using namespace std;

int fun(vector<int>& c,int a,int t[])

{

if(a==0)

{

return 0;

}

if(t[a]!=0)

{

return t[a];

}

int ans=INT\_MAX;

for(int i=0;i<c.size();i++)

{

if(a-c[i]>=0)

{

int subprob=fun(c,a-c[i],t);

ans=min(ans,subprob+1);

}

}

t[a]=ans;

return t[a];

}

int main() {

int n;

cin>>n;

vector<int>v;

v.push\_back(1);

v.push\_back(3);

v.push\_back(4);

int t[1001]={0};

cout<<fun(v,n,t);

}

**6)Longest Common Subsequence**

#include<bits/stdc++.h>

using namespace std;

int lcs(string s,string l,int n,int m)

{

int t[n+1][m+1];

for(int i=0;i<n+1;i++)

{

for(int j=0;j<m+1;j++)

{

if(i==0 || j==0)

{

t[i][j]=0;

}

}

}

for(int i=1;i<n+1;i++)

{

for(int j=1;j<m+1;j++)

{

if(s[i-1]==l[j-1])

{

t[i][j]=1+t[i-1][j-1];

}

else

{

t[i][j]=max(t[i-1][j],t[i][j-1]);

}

}

}

return t[n][m];

}

int main()

{

int n;

cin>>n;

string s;

for(int i=0;i<n;i++)

{

int input;

cin>>input;

s.push\_back(input+'0');

}

int m;

cin>>m;

string l;

for(int i=0;i<m;i++)

{

int input;

cin>>input;

l.push\_back(input+'0');

}

cout<<lcs(s,l,n,m);

}