

Assignment No: - HPC-Group1-1

TITLE	Parallel Searching Algorithms
PROBLEM STATEMENT /DEFINITION	Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .
OBJECTIVE	<ul style="list-style-type: none">• To understand concept of Breadth First Search and Depth First Search based on sequential algorithm.• To understand concept of parallel algorithm.• To compare performance by varying number of processors used and also with sequential algorithm.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems 1. Open source Linux or its derivative 2. Master slave parallel computation model
REFERENCES	<ul style="list-style-type: none">• Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, "Introduction to Parallel Computing", 2nd edition, Addison-Wesley, 2003, ISBN: 0-201-64865-2.
INSTRUCTIONS FOR WRITING JOURNAL	<ol style="list-style-type: none">1. Date2. Assignment no.3. Problem definition4. Learning objective5. Learning Outcome6. Concepts related Theory7. Related Mathematics8. Algorithm.9. Test Cases10. Conclusion and applications

Assignment No: - HPC-Group1-1

Problem statement: Design and implement Parallel Breadth First Search and Depth First Search based on existing algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .

- **Aim**
Write a program to design and implement parallel Breadth First Search and Depth First Search algorithm
- **Prerequisites**
 - Concept of existing sequential algorithms.
 - Concept of High Performance Computing.
- **Learning Objectives**
 - To understand concept of Breadth First Search and Depth First Search based on sequential algorithm.
 - To understand concept of parallel algorithm.
 - To compare performance by varying number of processors used and also with sequential algorithm.
- **Learning Outcomes**
After successfully completing this assignment, you should be able to
 - Display result for parallel Breadth First Search and Depth First Search
 - Analyze performance by varying number of processors used and also with sequential algorithm.
 - Calculate speedup, efficiency, throughput
- **Concepts related Theory :-**

Breadth First Search (BFS)

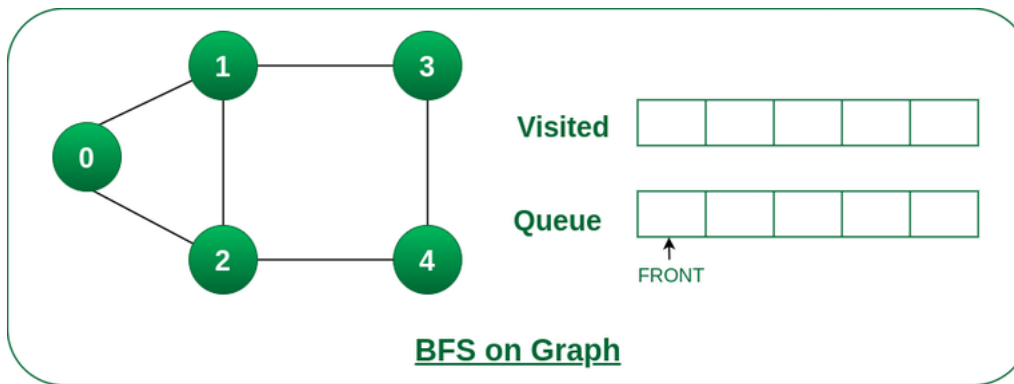
The breadth-first search (BFS) algorithm is used to search a tree or graph data structure for a node that meets a set of criteria. It starts at the tree's root or graph and searches/visits all nodes at the current depth level before moving on to the nodes at the next depth level.

Follow the below method to implement BFS traversal.

- Declare a queue and insert the starting vertex.
- Initialize a **visited** array and mark the starting vertex as visited.
- Follow the below process till the queue becomes empty:
 - Remove the first vertex of the queue.
 - Mark that vertex as visited.
 - Insert all the unvisited neighbours of the vertex into the queue.

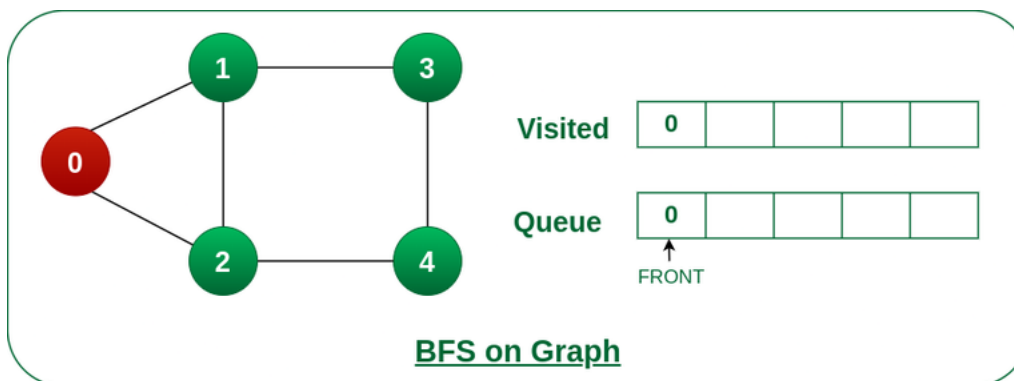
Illustration:

Step1: Initially queue and visited arrays are empty.



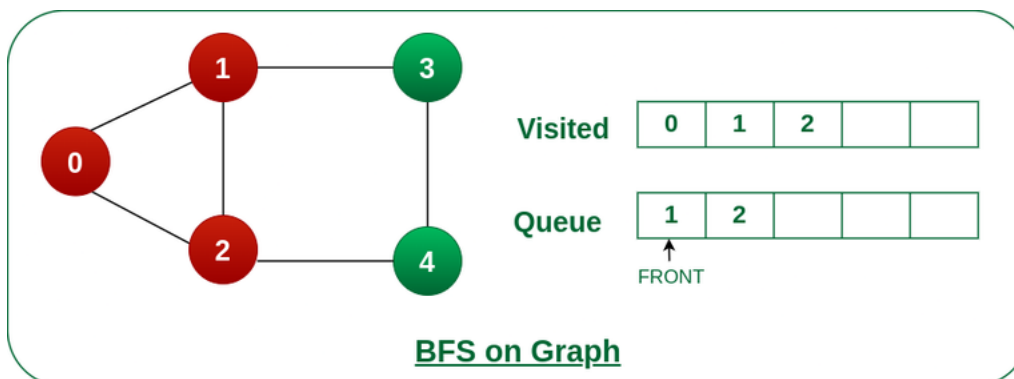
Queue and visited arrays are empty initially.

Step2: Push node 0 into queue and mark it visited.



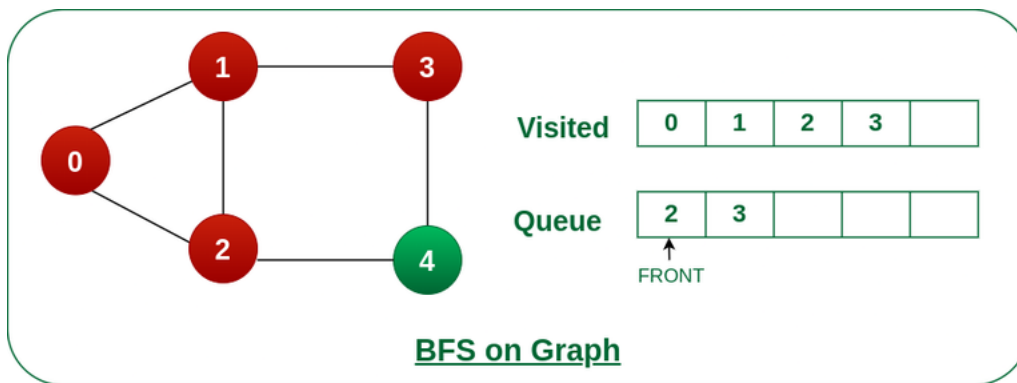
Push node 0 into queue and mark it visited.

Step 3: Remove node 0 from the front of queue and visit the unvisited neighbours and push them into queue.



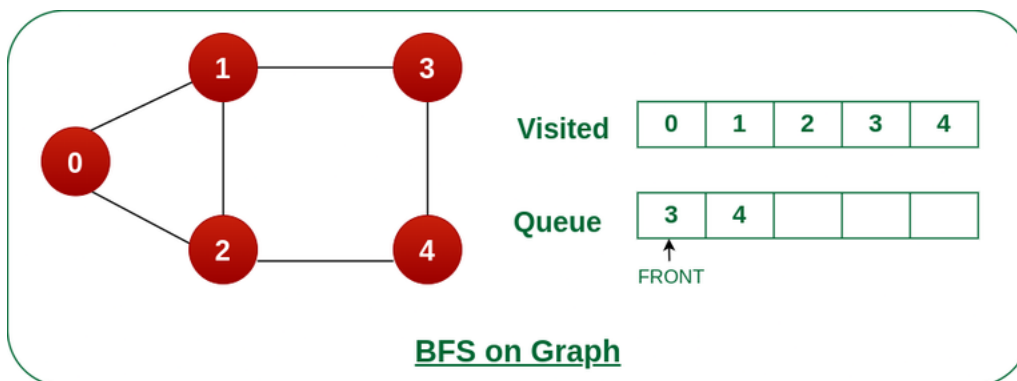
Remove node 0 from the front of queue and visited the unvisited neighbours and push into queue.

Step 4: Remove node 1 from the front of queue and visit the unvisited neighbours and push them into queue.



Remove node 1 from the front of queue and visited the unvisited neighbours and push

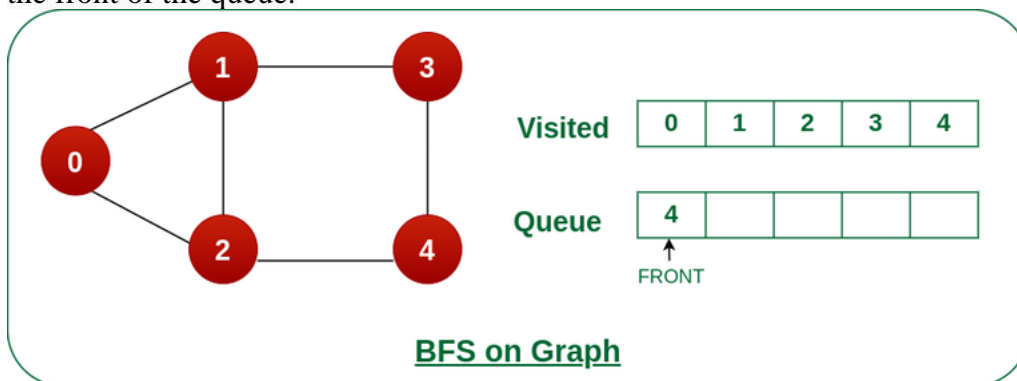
Step 5: Remove node 2 from the front of queue and visit the unvisited neighbours and push them into queue.



Remove node 2 from the front of queue and visit the unvisited neighbours and push them into queue.

Step 6: Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.

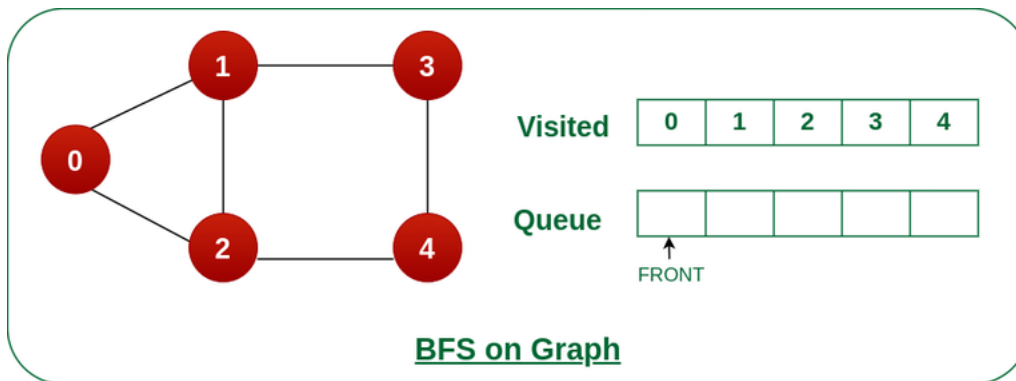
As we can see that every neighbours of node 3 is visited, so move to the next node that are in the front of the queue.



Remove node 3 from the front of queue and visit the unvisited neighbours and push them into queue.

Steps 7: Remove node 4 from the front of queue and visit the unvisited neighbours and push them into queue.

As we can see that every neighbours of node 4 are visited, so move to the next node that is in the front of the queue.



Remove node 4 from the front of queue and visit the unvisited neighbours and push them into queue. Now, Queue becomes empty, So, terminate these process of iteration.

Depth First Search Algorithm

A standard DFS implementation puts each vertex of the graph into one of two categories:

1. Visited
2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

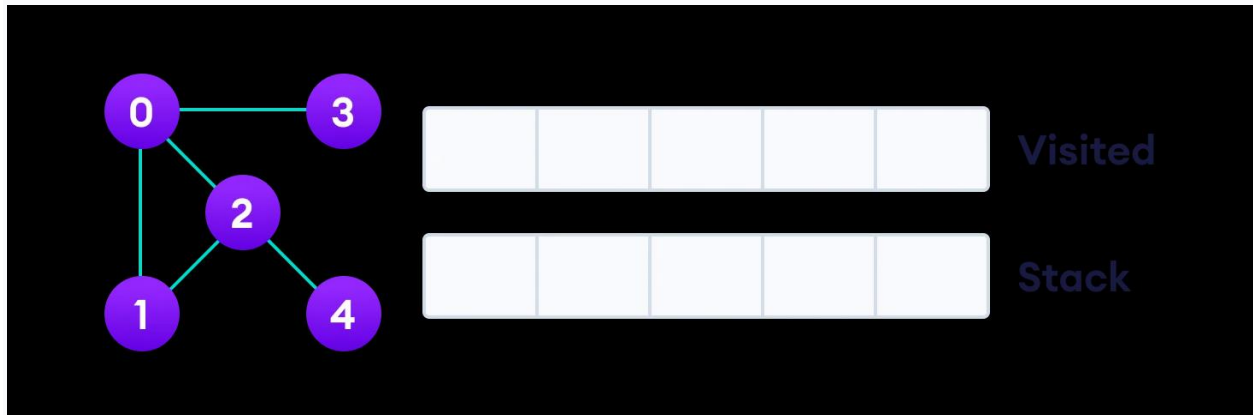
The DFS algorithm works as follows:

1. Start by putting any one of the graph's vertices on top of a stack.
2. Take the top item of the stack and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the stack.

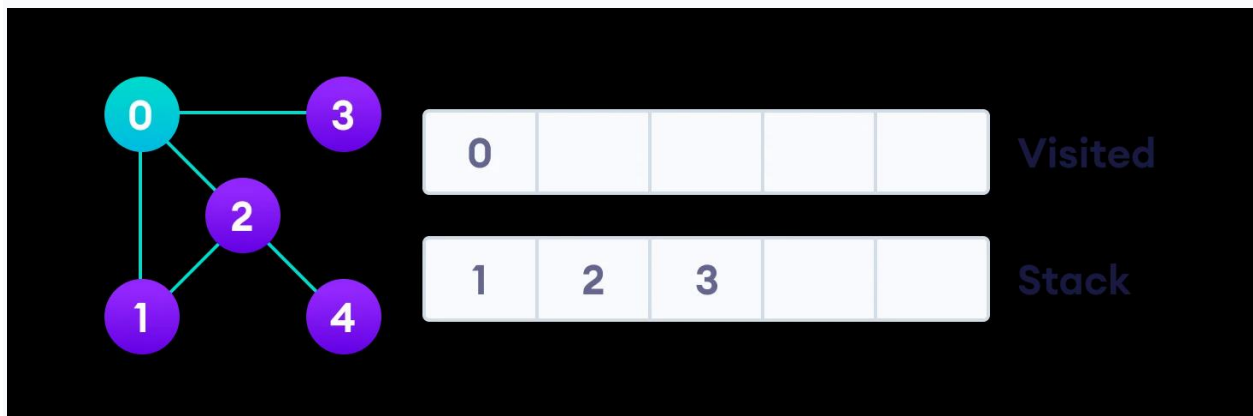
- Keep repeating steps 2 and 3 until the stack is empty.

Depth First Search Example

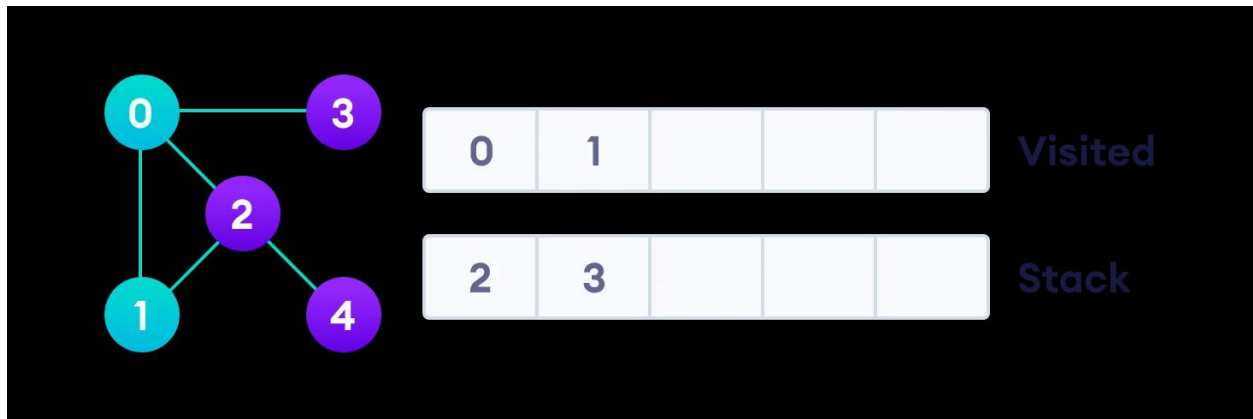
Let's see how the Depth First Search algorithm works with an example. We use an undirected graph with 5 vertices.



We start from vertex 0, the DFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.



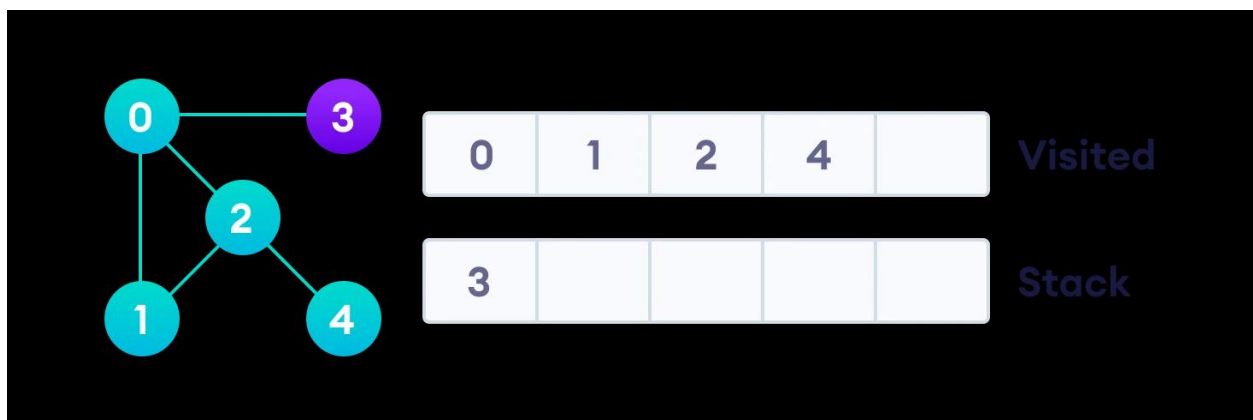
Next, we visit the element at the top of stack i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

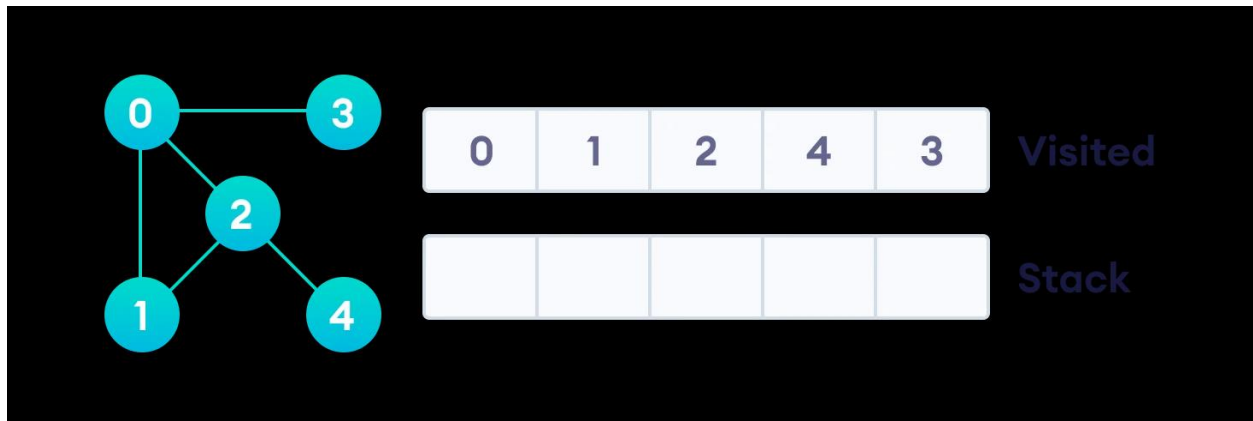


Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.



Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the top of the stack and visit it.

After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.



After we visit the last element 3, it doesn't have any unvisited adjacent nodes, so we have completed the Depth First Traversal of the graph.

Parallel DFS and BFS: Extract parallelism from existing sequential algorithms and implement using OPENMP

Conclusion :

After successfully completing this assignment, student should be able to understand and implement parallel BFS and DFS in OpenMP.

Assignment No: - HPC-Group1-2

TITLE	Parallel Sorting Algorithms
PROBLEM STATEMENT /DEFINITION	Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.
OBJECTIVE	<ul style="list-style-type: none">• To understand concept of Bubble Sort and Merge Sort based on sequential algorithm.• To understand concept of parallel algorithm.• To compare performance by varying number of processors used and also with sequential algorithm.
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating Systems 1. Open source Linux or its derivative 2. Master slave parallel computation model
REFERENCES	<ul style="list-style-type: none">• Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, "Introduction to Parallel Computing", 2nd edition, Addison-Wesley, 2003, ISBN: 0-201-64865-2.
INSTRUCTIONS FOR WRITING JOURNAL	<ol style="list-style-type: none">1. Date2. Assignment no.3. Problem definition4. Learning objective5. Learning Outcome6. Concepts related Theory7. Related Mathematics8. Algorithm.9. Test Cases10. Conclusion and applications

Assignment No: - HPC-Group1-2

Problem statement: Write a program to implement Parallel Bubble Sort and Merge sort using OpenMP. Use existing algorithms and measure the performance of sequential and parallel algorithms.

- **Aim**

Write a program to design and implement parallel Bubble Sort and Merge Sort algorithm.

- **Prerequisites**

- Concept of existing sequential algorithms.
- Concept of High Performance Computing.

- **Learning Objectives**

- To understand concept of Bubble Sort and Merge Sort based on sequential algorithm.
- To understand concept of parallel algorithm.
- To compare performance by varying number of processors used and also with sequential algorithm.

- **Learning Outcomes**

After successfully completing this assignment, you should be able to

- Display result for parallel Bubble Sort and Merge Sort.
- Analyze performance by varying number of processors used and also with sequential algorithm.
- Calculate speedup, efficiency, throughput

- **Concepts related Theory :-**

- **Bubble Sort Algorithm:-**

- Sequential Bubble Sort Algorithm:**

- One of the straight-forward sorting methods

- Cycles through the list

- Compares consecutive elements and swaps them if necessary

- Stops when no more out of order pair.

- _Slow & inefficient

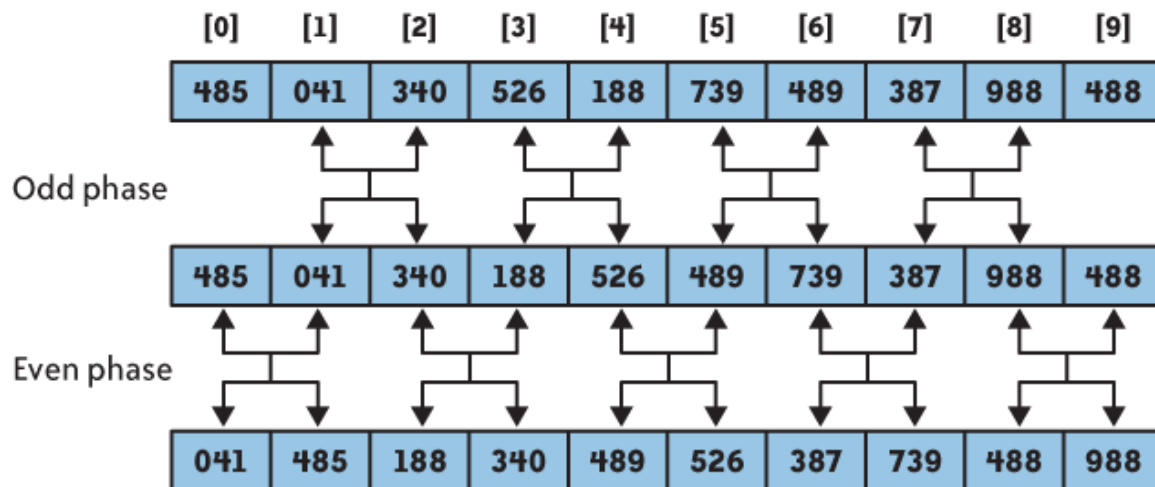
- _Average performance is $O(n^2)$.

- Parallel Bubble Sort**

- Compare all pairs in the list in parallel.

- When to stop?

- Shared flag, sorted, initialized to true at beginning of each iteration (2 phases), if any processor perform swap, sorted = false



Parallel Bubble Sort Complexity

Sequential bubble sort, $O(n^2)$.

Parallel bubble sort? (if we have unlimited # of processors)

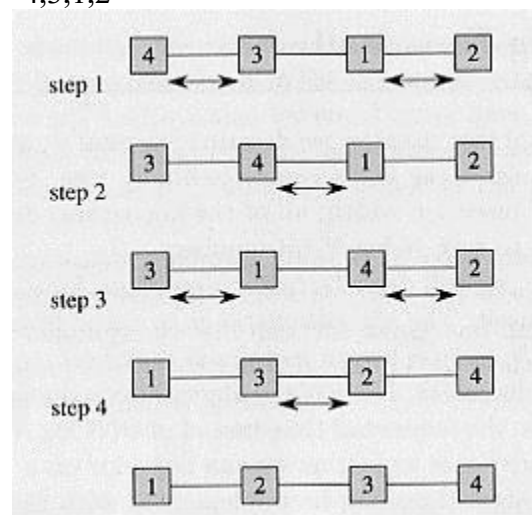
Do $n-1$ comparisons for each iteration $\Rightarrow O(n)$

Parallel Bubble Sort Example:

How many steps does it take to sort the following sequence from least to greatest using the

Parallel Bubble Sort? How does the sequence look like after 2 cycles?

•4,3,1,2



•Merge Sort Algorithm:-

Sequential Merge Sort:

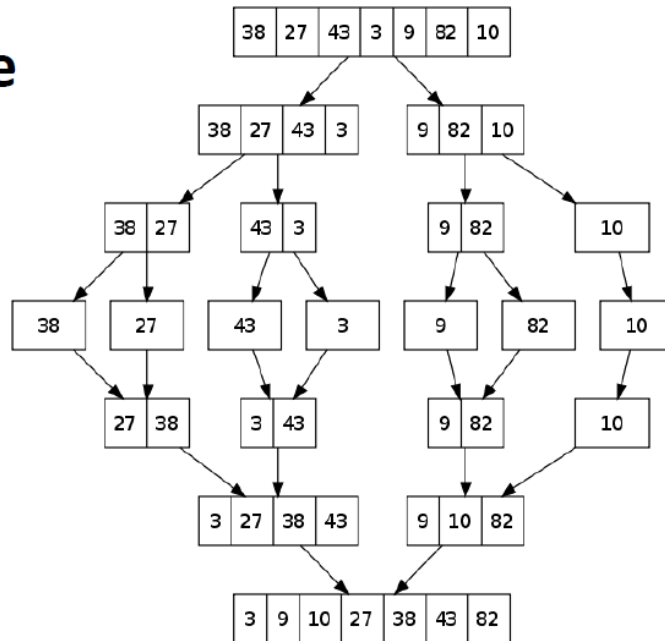
Divide and Conquer:

- Dividing problem into sub-problems
- Division usually done through recursion
- Solutions from sub-problems then combined to give solution to the original problem.

Collects sorted list onto one processor

- Merges elements as they come together
- Simple tree structure
- Parallelism is limited when near the root

Example



Merge Sort Complexity:

$$T(n) = \begin{cases} b & n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & n > 1 \end{cases}$$

Solve the recurrence relation

$$T(n) = O(n \log n)$$

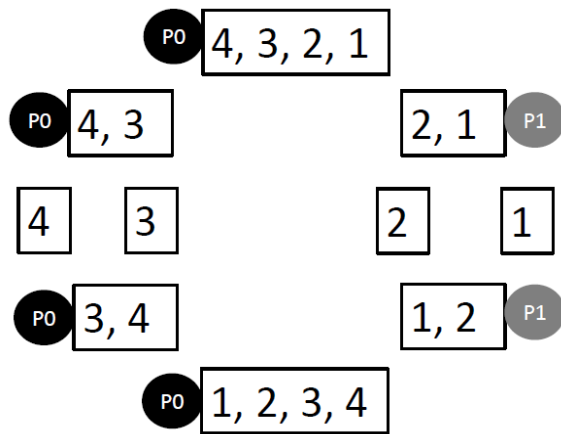
Parallel Merge Sort:

- Parallelize processing of sub-problems
- Max parallelization achieved with one processor per node (at each layer/height).

Parallel Merge Sort Example:

Perform Merge Sort on the following list of elements. Given 2 processors, P0 & P1, which processor is responsible for which comparison?

•4,3,2,1



Parallel Merge Sort Complexity:

- Merge sort, $O(n \log n)$
- Easy way to remember complexity, n (elements) $\times \log n$ (tree depth)
- If we have n processors, $O(\log n)$

- **Test Cases:**

Students should write test cases depending on their input and test the program on large input data

- **YouTube video links:**

<https://youtu.be/F8Cluc31bJc>

<https://youtu.be/SpBPp3JjFb4>

<https://youtu.be/QaiEB4BjjNg>

- **Conclusion :**

After successfully completing this assignment, student should be able to understand and implement parallel bubble sort and merge sort in OpenMP.

Assignment No: - HPC-Group1-3

TITLE	Parallel Computing Using CUDA
PROBLEM STATEMENT / DEFINITION	Implement Min, Max, Sum and Average operations using Parallel Reduction.
OBJECTIVE	<ul style="list-style-type: none">• Learn parallel decomposition of problem.• Learn parallel computing using CUDA.
S/W PACKAGES AND HARDWARE APPARATUS USED	<ol style="list-style-type: none">1. Operating System : 64-bit Open source Linux or its derivative2. Programming Language: C/C++3. NVidia GPU4. CUDA API
REFERENCES	<ul style="list-style-type: none">• Jason sanders, Edward Kandrot, “CUDA by Example”, Addison-Wesley, ISBN-13: 978-0-13-138768-3• Shane Cook, “CUDA Programming: A Developer's Guide to Parallel Computing with GPUs”, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA 2013 ISBN: 9780124159884
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	<ol style="list-style-type: none">1. Date2. Assignment no.3. Problem definition4. Learning objective5. Learning outcome6. Concepts related Theory7. Test cases8. Program code with proper documentation.9. Output of program.10. Conclusion and applications (the verification and testing of outcomes)

Assignment No: - HPC-Group1-3

Problem statement: Implement Min, Max, Sum and Average operations using Parallel Reduction.

- **Aim:**
Parallel Computing Using CUDA.

- **Prerequisites**
C/C++ Programming

- **Learning Objectives**
 - Learn parallel decomposition of problem.
 - Learn parallel computing using CUDA.

- **Learning Outcome:**

Students will be able to

1. decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.
2. Analyse the performance using parameters like speedup, efficiency, throughput

- **Theory**

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called **decomposition**. Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem. Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size.

A recursive program for finding the minimum in an array of numbers A of length n

Suppose we have an array A with n elements. Decompose this array into subgroups with elements 2..So the total subgroups will be $n/2$. Find minimum from each subgroup parallelly. As a result , we get $n/2$ elements. Apply this same procedure recursively till we get single element. This element will be the smallest among all the elements of the given array.

Overall recursive procedure to find minimum element is as follows:

procedure RECURSIVE_MIN (A, n)

begin

if ($n = 1$) then

```

min := A[0];

else

lmin := RECURSIVE_MIN (A, n/2);

rmin := RECURSIVE_MIN (&(A[n/2]), n - n/2);

if (lmin < rmin) then

min := lmin;

else

min := rmin;

endelse;

endelse;

return min;

end RECURSIVE_MI

```

Consider an array {4,9,1,7,8,11,2,12}. Divide this array into subgroups as shown in following figure. So we have {4,9}, {1,7}, {8,11}, {2,12}. Find minimum from each subgroup. So, we get {4,1,8,2}. Again divide this into subgroups {4,1}, {2,12}.. Find minimum from each subgroup; we get {1,2}. Find minimum among 1 and 2. That is 1. Hence 1 is the minimum or smallest among all the elements of array.



Fig: Finding Minimum by recursive decomposition

Similarly, we can find maximum among elements in an array. We can find sum of all the elements of array with the same procedure i.e. by decomposition and recursion. For average, take sum by recursion and divide it by number of elements. Standard deviation is given by formula

$$\sigma = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

where \bar{x} is mean.

How to run CUDA Program on Remote Machine

1. Open Terminal

2. Get log in to remote system which has GPU and CUDA installed in it.
e.g. `ssh student@10.10.15.21`

3. Once you get logged in to system, create a cude file with extension .cu and write code in it.

e.g. `cat >> sample.cu`

Write code here

Press Ctrl+D to come outside the cat command.

4. Compile CUDA program using `nvcc` command.

e.g. `nvcc sample.cu`

5. It will create executable file `a.out`. Rut it.

e.g. `./a.out`

When you are compiling using `nvcc` command, you may get compiler error “`nvcc` command not found”

In this case, on remote machine, of which you are using GPU, you have to run following commands:

Open the terminal and type:

`gedit ~/.bashrc`

This will open `.bashrc` for editing.

Note: You have check path of CUDA bin folder. Suppose path is `/usr/local/cuda-8.0/bin`

Add the following to the end of your `.bashrc` file.

`export PATH="$PATH:/usr/local/cuda-8.0/bin"`

This sets your `PATH` variable to the existing `PATH` plus what you add to the end.

Run following command to reload the configuration.

`source ~/.bashrc`

- **Test data:**

Take array with different values of elements, test the program on large input data.

- **Conclusion :**

After successfully completing this assignment, student should be able to understand and implement Min, Max, Sum and Average operations in CUDA

Assignment No: - HPC-Group1-4

TITLE	Parallel Computing Using CUDA
PROBLEM STATEMENT / DEFINITION	Write a CUDA Program for : 1. Addition of two large vectors 2. Matrix Multiplication using CUDA C
OBJECTIVE	<ul style="list-style-type: none">• Learn parallel decomposition of problem.• Learn parallel computing using CUDA.
S/W PACKAGES AND HARDWARE APPARATUS USED	1. Operating System : 64-bit Open source Linux or its derivative 2. Programming Language: C/C++ 3. NVidia GPU 4. CUDA API
REFERENCES	<ul style="list-style-type: none">• Jason sanders, Edward Kandrot, “CUDA by Example”, Addison-Wesley, ISBN-13: 978-0-13-138768-3• Shane Cook, “CUDA Programming: A Developer's Guide to Parallel Computing with GPUs”, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA 2013 ISBN: 9780124159884
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	1.Date 2.Assignment no. 3. Problem definition 4. Learning objective 5. Learning outcome 6. Related Mathematics 7. Concepts related Theory 8. Test cases 9. Program code with proper documentation. 10. Output of program. 11. Conclusion and applications (the verification and testing of outcomes)

Assignment No: - HPC-Group1-4

Problem statement: Write a CUDA Program for :

1. Addition of two large vectors
2. Matrix Multiplication using CUDA C

- **Aim:**
Parallel Computing Using CUDA.

- **Prerequisites**
C/C++ Programming

- **Learning Objectives**
 - Learn parallel decomposition of problem.
 - Learn parallel computing using CUDA.

- **Learning Outcome:**
Students will be able to decompose problem into sub problems, to learn how to use GPUs, to learn to solve sub problem using threads on GPU cores.

- **Theory**

Dividing a computation into smaller computations and assigning them to different processors for parallel execution are the two key steps in the design of parallel algorithms. The process of dividing a computation into smaller parts, some or all of which may potentially be executed in parallel, is called **decomposition**. Tasks are programmer-defined units of computation into which the main computation is subdivided by means of decomposition. Simultaneous execution of multiple tasks is the key to reducing the time required to solve the entire problem. Tasks can be of arbitrary size, but once defined, they are regarded as indivisible units of computation. The tasks into which a problem is decomposed may not all be of the same size.

In addition of two vectors, we have to add i th element from first array with i th element of second array to get i th element of resultant array. We can allocate this each addition to distinct thread. Same thing can be done for the product of two vectors.

There can be three cases for addition of two vectors using CUDA.

1. n blocks and one thread per block.
2. 1 block and n threads in that block.
3. m blocks and n threads per block.

In addition of two matrices, we have to add (i,j) th element from first matrix with (i,j) th element of second matrix to get (i,j) th element of resultant matrix.. We can allocate this each addition to distinct thread.

There can be two cases for addition of two matrices using CUDA.

1. Two dimensional blocks and one thread per block.
2. One block and two dimensional threads in that block.

Same cases can be considered for multiplication of two matrices.

How to run CUDA Program on Remote Machine

1. Open Terminal
2. Get log in to remote system which has GPU and CUDA installed in it.
e.g. `ssh student@10.10.15.21`
3. Once you get logged in to system, create a cude file with extension `.cu` and write code in it.

e.g. `cat >> sample.cu`

Write code here

Press `Ctrl+D` to come outside the `cat` command.

4. Compile CUDA program using `nvcc` command.
e.g. `nvcc sample.cu`

5. It will create executable file `a.out`. Rut it.
e.g. `./a.out`

When you are compiling using `nvcc` command, you may get compiler error “`nvcc` command not found”

In this case, on remote machine, of which you are using GPU, you have to run following commands:

Open the terminal and type:

`gedit ~/.bashrc`

This will open `.bashrc` for editing.

Note: You have check path of CUDA bin folder. Suppose path is `/usr/local/cuda-8.0/bin`

Add the following to the end of your `.bashrc` file.

`export PATH="$PATH:/usr/local/cuda-8.0/bin"`

This sets your `PATH` variable to the existing `PATH` plus what you add to the end.

Run following command to reload the configuration.

`source ~/.bashrc`

- **Test data:**

Take vector or matrices with different values of elements.

- **Conclusion :**

After successfully completing this assignment, student should be able to understand and implement Addition of two large vectors and Matrix Multiplication operations in CUDAC

Assignment No: - DL-Group1-1

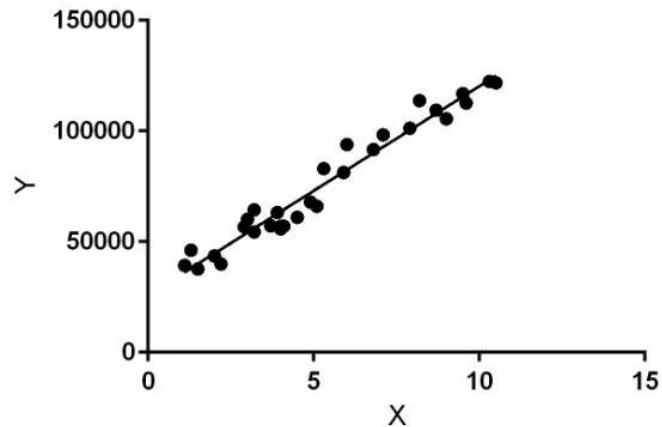
TITLE	Linear regression by using Deep Neural network
PROBLEM STATEMENT /DEFINITION	Implement Boston housing price prediction problem by Linear regression using Deep Neural network. Use Boston House price prediction dataset.
OBJECTIVE	To build a regression model to predict the price of houses.
OUTCOME	To understand the exploratory data analysis,split the training and testing data, Model Evaluation and Prediction by the linear regression on the Boston housing dataset.
S/W PACKAGES AND HARDWARE/ APPARATUS USED	Jupyter notebook IDE, python3 PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	1. https://studygyaan.com/data-science-ml/linear-regression-machine-learning-project-for-house-price-prediction 2. https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155
STEPS	Installing Jupyter notebook with python3 1) import the required libraries- numpy,matplotlib.pyplot,pandas,seaborn 2) Importing Data (kaggle and scikit-learn library) and Checking out 3) Exploratory Data Analysis for House Price Prediction 4) Get Data Ready For Training a Linear Regression Model 5) Split Data into Train, Test 6) Creating and Training the LinearRegression Model 7) LinearRegression Model Evaluation 8) Predictions from our Linear Regression Model 9) Regression Evaluation Metrics.
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition

	<ol style="list-style-type: none">4. Learning objective5. Learning Outcome6. Concepts related Theory7. Algorithm8. Test cases9. Conclusion/Analysis
--	--

Prerequisites: Programming language, machine learning

Concepts related Theory:

Linear Regression is a Supervised Machine Learning Model for finding the relationship between independent variables and dependent variable. Linear regression performs the task to predict the response (dependent) variable value (y) based on a given (independent) explanatory variable (x). So, this regression technique finds out a linear relationship between x (input) and y (output).



Algorithm:

Import Libraries: Install the required libraries and setup for the environment for the assignment. importing SciKit-Learn, Pandas, Seaborn, Matplotlib and Numpy.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

%matplotlib inline
```

The purpose of "%matplotlib inline" is to add plots to your Jupyter notebook.

Importing Data and Checking out: As data is in the CSV file, we will read the CSV using pandas read_csv function and check the first 5 rows of the data frame using head().

```
HouseDF.info()
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income          5000 non-null float64
Avg. Area House Age       5000 non-null float64
Avg. Area Number of Rooms 5000 non-null float64
Avg. Area Number of Bedrooms 5000 non-null float64
Area Population            5000 non-null float64
Price                     5000 non-null float64
Address                   5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

```
HouseDF.describe()
```

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

```
HouseDF.columns
```

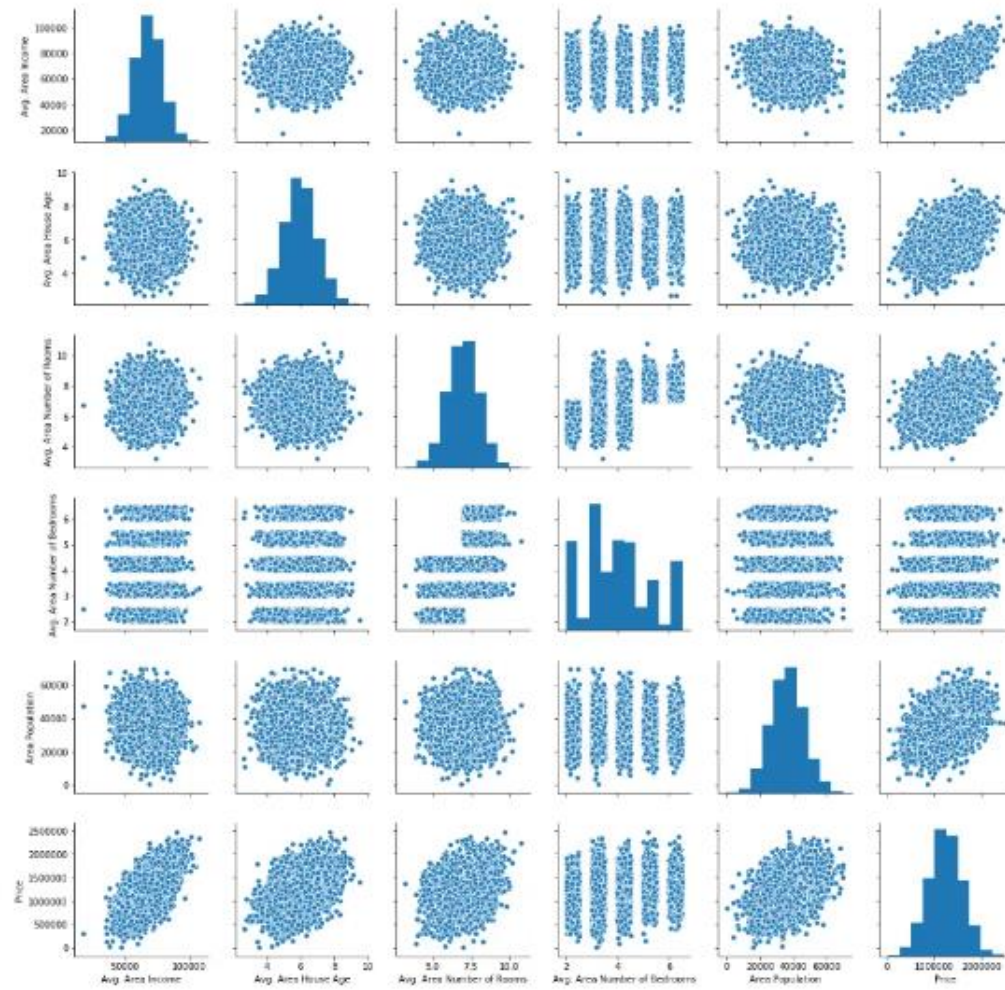
OUTPUT

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms', 'Avg. Area Numt
```

Exploratory Data Analysis for House Price Prediction : create some simple plot for visualizing the data.

```
sns.pairplot(HouseDF)
```

```
<seaborn.axisgrid.PairGrid at 0x67b60e8d08>
```



```
sns.heatmap(HouseDF.corr(), annot=True)
```

<matplotlib.axes._subplots.AxesSubplot at 0x67b868c948>



Get Data Ready For Training a Linear Regression Model: now begin to train out the regression model. We will need to first split up our data into an X list that contains the features to train on, and a y list with the target variable, in this case, the Price column. We will ignore the Address column because it only has text which is not useful for linear regression modeling.

X and y List

```
X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
              'Avg. Area Number of Bedrooms', 'Area Population']]

y = HouseDF['Price']
```

Split Data into Train, Test: Now split our dataset into a training set and testing set using sklearn train_test_split(). The training set will be used for training the model and testing set for testing the model. We are creating a split of 40% training data and 60% of the training set.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

X_train and y_train contain data for the training model. X_test and y_test contain data for the testing model. X and y are features and target variable names.

Creating and Training the LinearRegression Model: import and create sklearn linear_model LinearRegression object and fit the training dataset in it.

```
from sklearn.linear_model import LinearRegression

lm = LinearRegression()

lm.fit(X_train,y_train)
```

OUTPUT

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

LinearRegression Model Evaluation: Now let's evaluate the model by checking out its coefficients and how we can interpret them

```
print(lm.intercept_)
```

OUTPUT

```
-2640159.796851911
```

```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient']) coeff_df
```

	Coefficient
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

What does coefficient of data says:

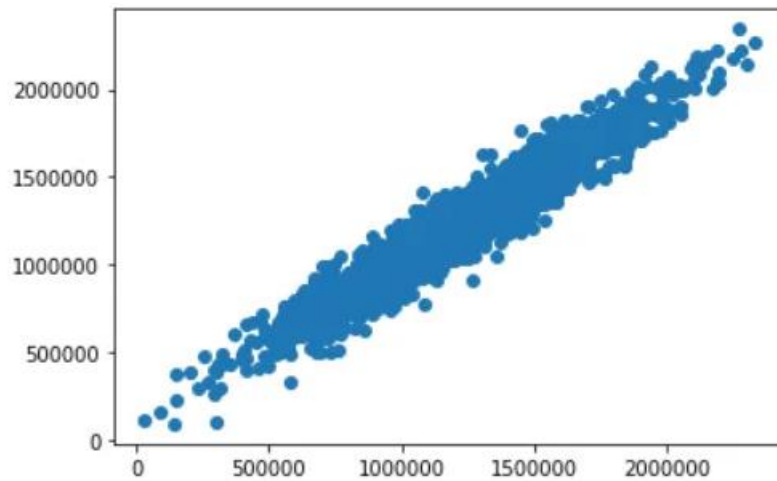
- Holding all other features fixed, a 1 unit increase in **Avg. Area Income** is associated with an **increase of \$21.52** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area House Age** is associated with an **increase of \$164883.28** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Rooms** is associated with an **increase of \$122368.67** .
- Holding all other features fixed, a 1 unit increase in **Avg. Area Number of Bedrooms** is associated with an **increase of \$2233.80** .
- Holding all other features fixed, a 1 unit increase in **Area Population** is associated with an **increase of \$15.15** .

Predictions from our Linear Regression Model: Let's find out the predictions of our test set and see how well it performs.

```
predictions = lm.predict(X_test)
```

```
plt.scatter(y_test,predictions)
```

<matplotlib.collections.PathCollection at 0x67b87ccc88>



In the above scatter plot, we see data is in a line form, which means our model has done good predictions.

Regression Evaluation Metrics:Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand because it's the average error.
- MSE is more popular than MAE because MSE "punishes" larger errors, which tends to be useful in the real world.
- RMSE is even more popular than MSE because RMSE is interpretable in the "y" units.

All of these are **loss functions** because we want to minimize them.

```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions)) print('MSE:', metrics.mean_squared_error(y_test, predictions))
```

OUTPUT

```
MAE: 82288.22251914957
MSE: 10460958907.209501
RMSE: 102278.82922291153
```

Conclusion-We have analyzed a Linear Regression Model which we help the real estate agent for estimating the house price.

Review Questions:

- 1) Difference between machine learning and deep learning?
- 2) How is linear regression used in house price prediction?
- 3) What is the purpose of house price prediction?
- 4) What is a Linear Regression?
- 5) What is the primary difference between R square and adjusted R square?
- 6) List out the formulas to find RMSE and MSE?
- 7) What are the disadvantages of the linear regression model?
- 8) Name a possible method of improving the accuracy of a linear regression model?

Assignment No: - DL-Group1-3

TITLE	Plant Disease analysis and detection using Convolutional neural network (CNN)
PROBLEM STATEMENT /DEFINITION	Convolutional neural network (CNN) (Any One from the following) <ul style="list-style-type: none">• Use any dataset of plant disease and design a plant disease detection system using CNN.• Use MNIST Fashion Dataset and create a classifier to classify fashion clothing into categories (Analyze and detect the plant diseases using CNN Model by training the data set.)
OBJECTIVE	To build a CNN model to detect Plant diseases .
OUTCOME	To understand the exploratory data analysis,split the training and testing data, Model Evaluation and Prediction by the CNN on the Plant disease detection data set
S/W PACKAGES AND HARDWARE/ APPARATUS USED	Jupyter notebook IDE, python3 PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	https://www.kaggle.com/code/deveshkaushik/plant-disease-detection-using-cnn https://www.kaggle.com/code/emmarex/plant-disease-detection-using-keras
STEPS	<ol style="list-style-type: none">1. Installing Jupyter notebook with python32. import the required libraries- numpy,matplotlib.pyplot,pandas,seaborn3. Importing Data (kaggle and scikit-learn library) and Checking out4. Exploratory Data Analysis for stock price Prediction and time series analysis5. Get Data Ready For Training RNN model6. Split Data into Train, Test
INSTRUCTIONS FOR WRITING JOURNAL	<ol style="list-style-type: none">1. Date2. Assignment no.3. Problem definition

	<ol style="list-style-type: none">4. Learning objective5. Learning Outcome6. Concepts related Theory7. Algorithm8. Test cases9. Conclusion/Analysis
--	--

Prerequisites: Programming language, machine learning

Concepts related Theory:

Deep Convolutional Neural Network is utilized in this study to identify infected and healthy leaves, as well as to detect illness in afflicted plants. The CNN model is designed to suit both healthy and sick leaves; photos are used to train the model, and the output is determined by the input leaf.

Algorithm:

Importing Libraries: Importing the required libraries which will be used to train the model such as numpy, tensor flow.

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from keras.losses import categorical_crossentropy
from sklearn.metrics import confusion_matrix, classification_report
from keras.utils.vis_utils import plot_model
```

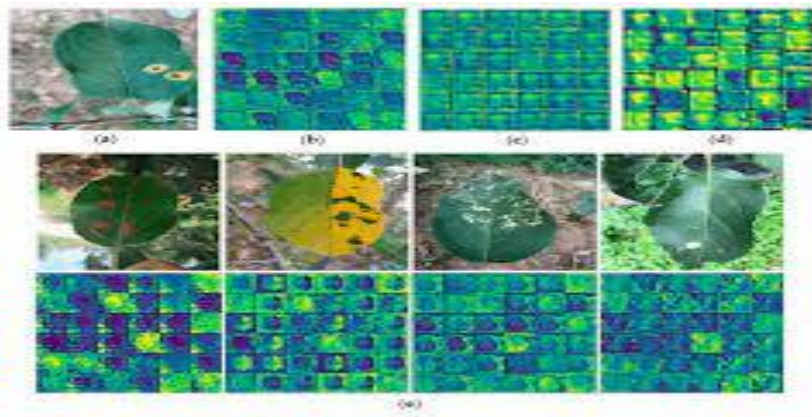
Importing Data and Checking out: As data is in the CSV file, we will read the CSV using pandas read_csv function and check the first 5 rows of the data frame using head().

```
HouseDF.info()
```

OUTPUT

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income                5000 non-null float64
Avg. Area House Age             5000 non-null float64
Avg. Area Number of Rooms       5000 non-null float64
Avg. Area Number of Bedrooms    5000 non-null float64
Area Population                 5000 non-null float64
Price                          5000 non-null float64
Address                        5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

Exploratory Data Analysis for Plant Disease :



Get Data Ready For Training using CNN Model:

```
print("[INFO] Splitting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 42)
```

```
aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2, horizontal_flip=True,
    fill_mode="nearest")
```

Split Data into Train, Test: Now split our dataset into a training set and testing set using `sklearn train_test_split()`. The training set will be used for training the model and testing set for testing the model. We are creating a split of 40% training data and 60% of the training set.

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

`X_train` and `y_train` contain data for the training model. `X_test` and `y_test` contain data for the testing model. `X` and `y` are features and target variable names.

Creating and Training the CNN Model : import and create `sklearn linear_model LinearRegression` object and fit the training dataset in it.

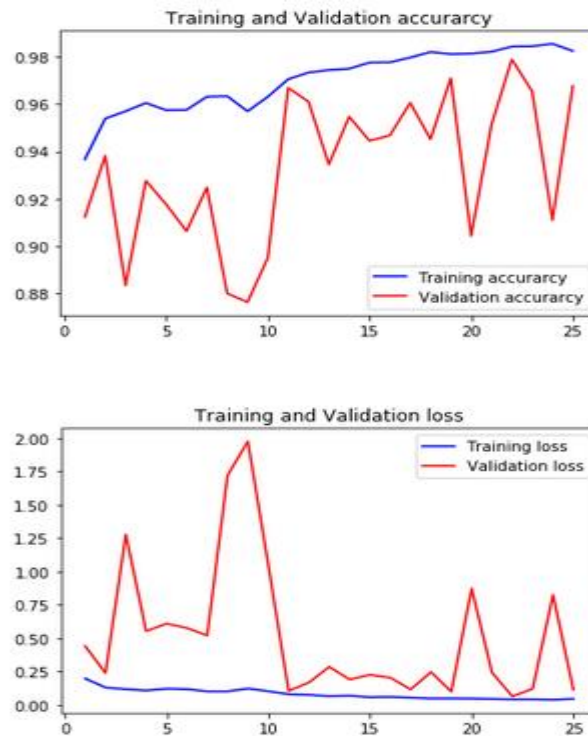
```

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()

plt.figure()
#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

```

CNN Model Evaluation: Now let's evaluate the model by checking out its coefficients and how we can interpret them.



10.

In the above scatter plot, we see data is in a line form, which means our model has done good

predictions.

Conclusion-We have analyzed a CNN Model which will help to predict plant disease.

Review Questions:

1. What is the use of the convolution layer in CNN?
2. What are the advantages of using CNN over DNN?
3. Why is CNN preferred over ANN for image data?
4. How would you visualise features of CNN in an image classification task?
5. What do you understand about shared weights in CNN?
6. Explain the role of a fully connected (FC) layer in CNN.
7. What is the importance of parameter sharing
8. Explain the different types of Pooling.

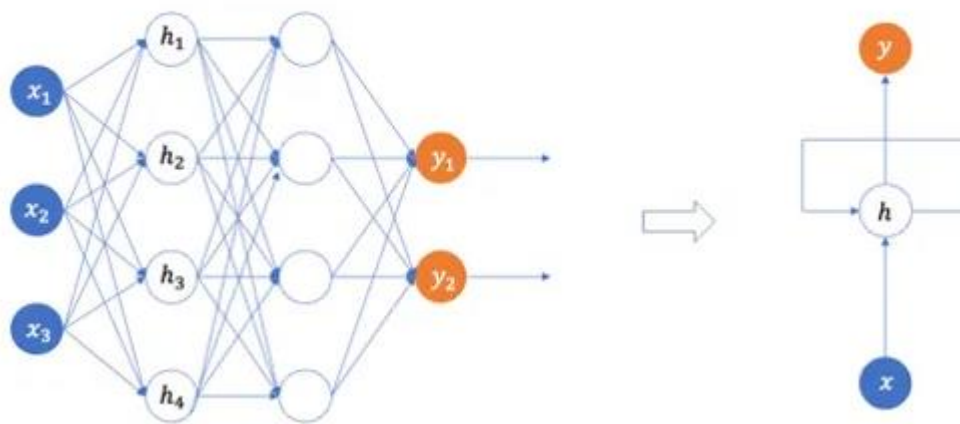
Assignment No: - DL-Group1-4

TITLE	Time series analysis
PROBLEM STATEMENT /DEFINITION	Recurrent neural network (RNN) Use the Google stock prices dataset and design a time series analysis and prediction system using RNN.
OBJECTIVE	To build a RNN model to predict the stock price and time analysis .
OUTCOME	To understand the exploratory data analysis,split the training and testing data, Model Evaluation and Prediction by the RNN on the google stock price dataset..
S/W PACKAGES AND HARDWARE/ APPARATUS USED	Jupyter notebook IDE, python3 PC with the configuration as Latest Version of 64 bit Operating Systems, Open Source Fedora-GHz. 8 G.B. RAM, 500 G.B. HDD, 15"Color Monitor, Keyboard, Mouse
REFERENCES	https://www.kaggle.com/code/kandij/google-stock-prediction-using-lstm-keras https://medium.com/analytics-vidhya/share-price-prediction-using-rnn-and-lstm-8776456dea6f
STEPS	Installing Jupyter notebook with python3 import the required libraries- numpy,matplotlib.pyplot,pandas,seaborn Importing Data (kaggle and scikit-learn library) and Checking out Exploratory Data Analysis for stock price Prediction and time series analysis Get Data Ready For Training RNN model Split Data into Train, Test
INSTRUCTIONS FOR WRITING JOURNAL	Date Assignment no. Problem definition Learning objective Learning Outcome Concepts related Theory Algorithm Test cases Conclusion/Analysis

Prerequisites: Programming language, machine learning

Concepts related Theory:

Recursive Neural Network (RNN) should be specially designed. RNN translates the provided inputs to machine readable vectors. Then the system processes each of this sequence of vectors one by one, moving from very first vector to the next one in a sequential order. While processing, the system passes the information through the hidden state (memory) to the next step of the sequence. Once the hidden state has collected all the existing information in the system until time period t , it is ready to move towards the next step and in this newer step the hidden step is classified as the previous hidden state defined. The outputs of the previous periods should somewhat become the inputs of the current periods. And the hidden layers will recursively take the inputs of previous periods. The hidden layer receives the inputs from the input layer, and there is a line to connect a hidden layer back to itself to represent the recursive nature.



Training through RNN

1. A single time step of the input is provided to the network.
2. Then calculate its current state using set of current input and the previous state.
3. The current h_t becomes h_{t-1} for the next time step.
4. One can go as many time steps according to the problem and join the information from all the previous states.
5. Once all the time steps are completed the final current state is used to calculate the output.
6. The output is then compared to the actual output i.e the target output and the error is generated.
7. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

Advantages of Recurrent Neural Network

1. An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
2. Recurrent neural network are even used with convolutional layers to extend the effective

pixel neighborhood.

Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using tanh or relu as an activation function.

For example , we have considered the amazon stock data for prediction and explained in detail in the similar way Google stock prediction should be implemented and plot the graph

Algorithm:

Import Libraries: Install the required libraries and setup for the environment for the assignment. importing SciKit-Learn, Pandas, Seaborn, Matplotlib ,Tensorflow and Numpy.

```
1  #importing libraries
2  import math
3  import pandas_datareader as web
4  import numpy as np
5  import pandas as pd
6  from sklearn.preprocessing import MinMaxScaler
7  from keras.models import Sequential
8  from keras.layers import Dense, LSTM
9  import matplotlib.pyplot as plt
10 plt.style.use('fivethirtyeight')
```

import_library.py hosted with ♥ by GitHub

[view raw](#)

Importing Data and Checking out: As data is in the CSV file, we will read the CSV using pandas read_csv function and check the first 5 rows of the data frame using head().

```
1  #dataset of amazon stock info
2  df = web.DataReader('AMZN', data_source='yahoo', start='2012-01-01', end='2020-08-17')
3  df
```

import_dataset.py hosted with ♥ by GitHub

[view raw](#)

Visualize the data using matplotlib

```
1 #Visualize the closing price history of amazon
2 plt.figure(figsize=(16,8))
3 plt.title('Close Price History of AMAZON')
4 plt.plot(df['Close'])
5 plt.xlabel('Date',fontsize=18)
6 plt.ylabel('close price USD ($)',fontsize=18)
7 plt.show()
```

import_visualize.py hosted with ❤ by GitHub

[view raw](#)



13.Training the data

```
1 #Create a new data frame with only the 'Close column'
2 data = df.filter(['Close'])
3
4 #Convert the dataframe to numpy array
5 dataset = data.values
6
7 #Get the number of rows to train the model on
8 training_data_len = math.ceil(len(dataset) * .8 )
9
10 training_data_len
```

import_columns.py hosted with ❤ by GitHub

[view raw](#)

Creating the Training data

```
1  #Create the training data set
2  #Crete the scaled training set
3  train_data = scaled_data[0:training_data_len , :]
4
5  #Split the data into x_train and _train data sets
6  x_train = []
7  y_train = []
8
9  for i in range(120, len(train_data)):
10     x_train.append(train_data[i-120:i, 0])
11     y_train.append(train_data[i,0])
12     if i<=121:
13         print(x_train)
14         print(y_train)
15         print()
```

import_training.py hosted with ♥ by GitHub

[view raw](#)

Building the Long Short Term Model

```
1  #compile the model
2  model.compile(optimizer='adam', loss='mean_squared_error')
```

import_compile.py hosted with ♥ by GitHub

[view raw](#)

→Compiling the model

```
1  #Train the model
2  model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Training the model

```
1 #Train the model
2 model.fit(x_train, y_train, batch_size=1, epochs=1)
```

Creating the test data

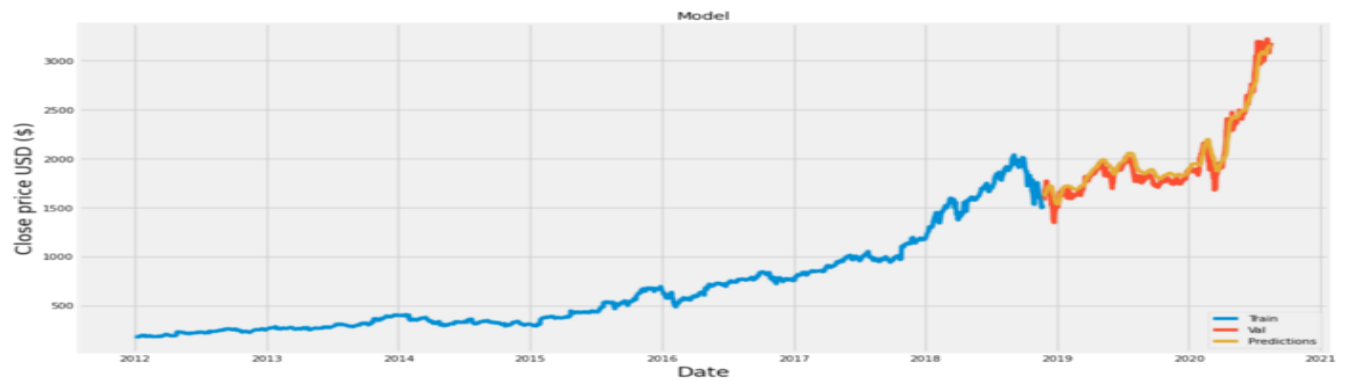
```
1 #Create the testing data set
2 #Create a new array containing scaled values from index 1616 to 2170
3 test_data = scaled_data[training_data_len - 120: , :]
4
5 #Create the data sets x_test and y_test
6 x_test = []
7 y_test = dataset[training_data_len:, :]
8 for i in range(120, len(test_data)):
9     x_test.append(test_data[i-120:i, 0])
```

To get predicted price values

```
1 #Get the model predicted price values
2 predictions = model.predict(x_test)
3 predictions = scaler.inverse_transform(predictions)
```

Evaluation and Visualizing the predicted values of the amazon stock prices

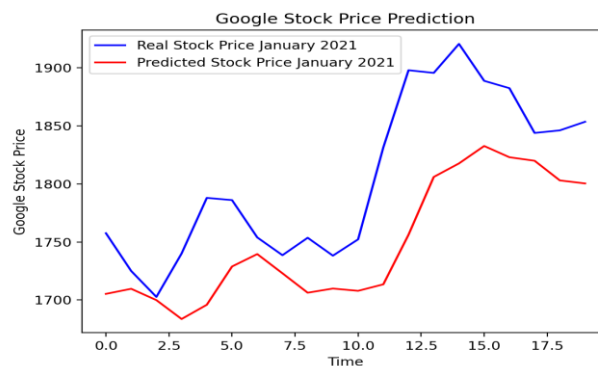
```
1 #plot the data
2 train = data[:training_data_len]
3 valid = data[training_data_len:]
4 valid['Predictions'] = predictions
5
6 #Visualize the data
7 plt.figure(figsize=(16,8))
8 plt.title('Model')
9 plt.xlabel('Date', fontsize=18)
10 plt.ylabel('Close price USD ($)', fontsize=18)
11 plt.plot(train['Close'])
12 plt.plot(valid[['Close', 'Predictions']])
13 plt.legend(['Train', 'Val', 'Predictions'], loc='lower right')
14 plt.show()
```



Conclusion-We have analyzed a RNN Model on amazon stock prediction and in similar way Google stock prediction can be calculated whose prediction model will be as below graph.

Prediction Results

Comparing the real Google stock values and predicted Google stock values that are generated using RNN model, for the test period, the first month of 2021. RNN based on 5 LSTMs was able to properly predict all upward and downward trends as we see that the red line corresponding to the predicted stock prices follows the same pattern as the blue line which corresponds to the real stock prices.



Review Questions:

1. What's the difference between Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) and in which cases would use each one?
2. How many dimensions must the inputs of an RNN layer have? What does each dimension represent? What about its outputs?
3. What are the main difficulties when training RNNs?
4. What are the uses of using RNN in NLP?
5. What's the difference between Traditional Feedforward Networks and Recurrent Neural Networks?
6. How to calculate the output of a Recurrent Neural Network (RNN)?
7. How does LSTM compare to RNN?