HPC Assignment 01

TITLE	**Parallel Computing Using CUDA**
PROBLEM STATEMENT / DEFINITION	a) Implement Parallel Reduction using Min, Max, Sum and Average operations. b) Write a CUDA program that, given an N-element vector, find- • The maximum element in the vector • The minimum element in the vector • The arithmetic mean of the vector • The standard deviation of the values in the vector Test for input N and generate a randomized vector V of length N (N should be large). The program should generate output as the two computed maximum values as well as the time taken to find each value.
OBJECTIVE	• 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. NVidea GPU 4. CUDA API
REFERENCES	• 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. A1

- **Aim:**
  Parallel Computing Using CUDA.
- **Problem Statement / Definition:**

a) Implement Parallel Reduction using Min, Max, Sum and Average operations.
b) Write a CUDA program that, given an N-element vector, find-

- The maximum element in the vector
- The minimum element in the vector
- The arithmetic mean of the vector
- The standard deviation of the values in the vector

Test for input N and generate a randomized vector V of length N (N should be large). The program should generate output as the two computed maximum values as well as the time taken to find each value.

- **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.

- **Related Mathematics**

Mathematical Model
Let S be the system set:
S = {s; e; X; Y; Fme;DD;NDD; Fc; Sc}
s=start state
e=end state
X=set of inputs
X = {X1}
where
X1 = Elements of Vector
Y= Output Set (Minimum / Maximum / Mean / Sum / Standard Deviation)
Fme is the set of main functions
Fme = {f1,f2,f3}
where
f1 = decomposition function
f2 = function to find Minimum / Maximum / Sum
f3 = function to merge results
f4 = function to Mean / Standard Deviation
DD= Deterministic Data
Vector of elements
NDD=Non-deterministic data

No non deterministic data
Fc =failure case:
No failure case identified for this application

- **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_MIN
```

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.

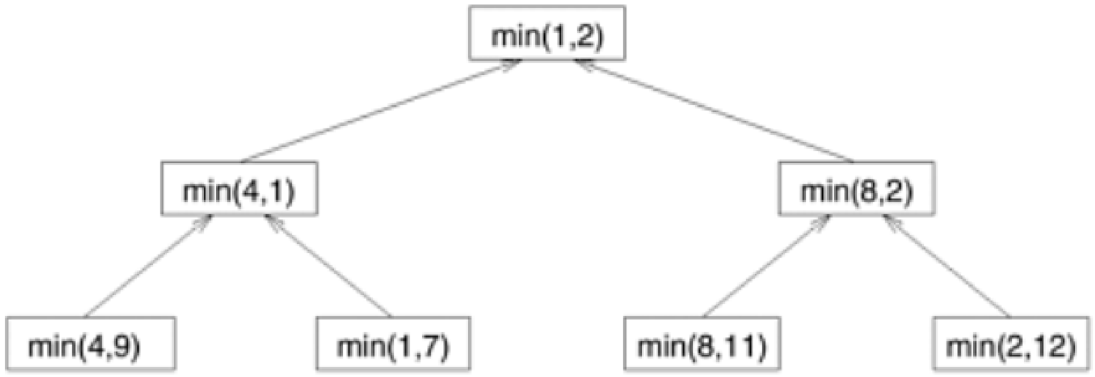


Fig: Finding Minimum by recursive decomposition

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

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

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.

HPC Assignment No 02

TITLE	**Parallel Computing Using CUDA**
PROBLEM STATEMENT / DEFINITION	Vector and Matrix Operations- Design parallel algorithm to 1. Add two large vectors 2. Multiply Vector and Matrix 3. Multiply two N × N arrays using n2 processors
OBJECTIVE	• Learn parallel decomposition of problem. • Learn parallel computing using CUDA.
S/W PACKAGES AND HARDWARE APPARATUS USED	5. Operating System : 64-bit Open source Linux or its derivative 6. Programming Language: C/C++ 7. NVidea GPU 8. CUDA API
REFERENCES	• 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	12. Date 13. Assignment no. 14. Problem definition 15. Learning objective 16. Learning outcome 17. Related Mathematics 18. Concepts related Theory 19. Test cases 20. Program code with proper documentation. 21. Output of program. 22. Conclusion and applications (the verification and testing of outcomes)

Assignment No. A2

- **Aim:**
  Parallel Computing Using CUDA.
- **Problem Statement / Definition:**

Vector and Matrix Operations-
Design parallel algorithm to
1. Add two large vectors
2. Multiply Vector and Matrix
3. Multiply two N × N arrays using n 2 processors

- **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.

- **Related Mathematics**

Mathematical Model
Let S be the system set:
S = {s; e; X; Y; Fme;DD;NDD; Fc; Sc}
s=start state
e=end state
X=set of inputs
X = {X1} where X1 is element of Vector or Matrix.
where
X1 = Elements of Vector
Y= Output Set (Sum or Product of elements of Vector/Matrix)
Fme is the set of main functions
Fme = {f1,f2,f3}
where
f1 = decomposition function
f2 = function to find Sum / Product
f3 = function to merge results
DD= Deterministic Data
Vector / Matrix of Elements.
NDD=Non-deterministic data
No non deterministic data
Fc =failure case:
No failure case identified for this application

- **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 ith element from first array with ith element of second array to get ith element of resultant array. We can allocate this each addition to distinct thread. Same thing can be done for the product of two vecors.

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.

HPC Assignment No: - A3

TITLE	Parallel Sorting Algorithms
PROBLEM STATEMENT /DEFINITION	For Bubble Sort and Merge Sort, based on existing sequential algorithms, design and implement parallel algorithm utilizing all resources available.
OBJECTIVE	9. To understand concept of Bubble Sort and Merge Sort based on sequential algorithm. 10. To understand concept of parallel algorithm. 11. 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	• 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	**Date** **Assignment no.** **Problem definition** **Learning objective** **Learning Outcome** **Concepts related Theory** **Related Mathematics** **Algorithm.** **Test Cases** **Conclusion and applications**

Assignment A3

23. Aim

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

24. Prerequisites

25. Concept of existing sequential algorithms.
26. Concept of High Performance Computing.

27. Learning Objectives

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

31. Learning Outcome

After successfully completing this assignment, you should be able to

32. Display result for parallel Bubble Sort and Merge Sort.

33. Analyze performance by varying number of processors used and also with sequential algorithm.

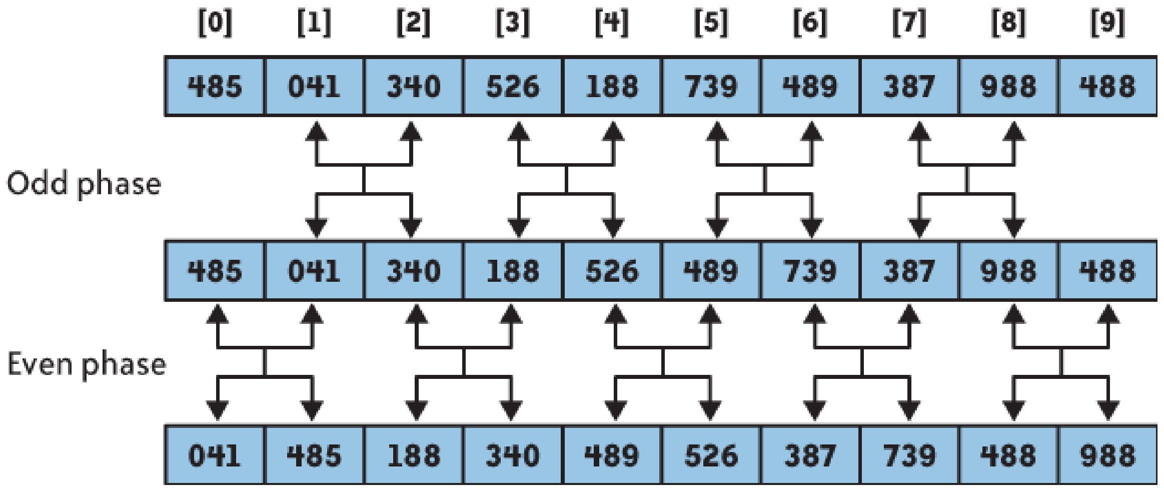
- **Concepts related Theory :-**

  - **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.

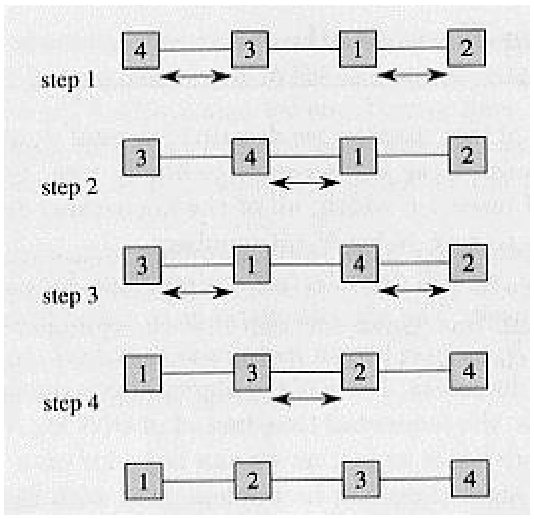
  - Slow & inefficient
  - Average performance is O(n^2).

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 => 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



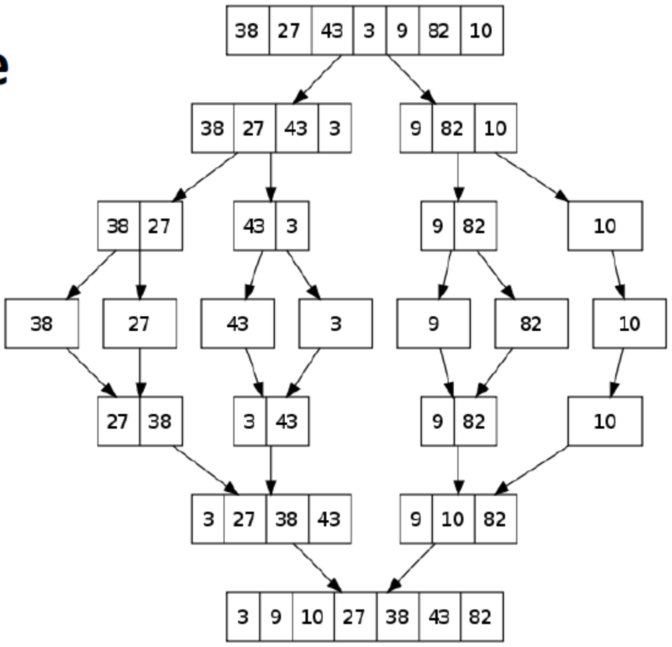
- **Merge Sort Algorithm:-**

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

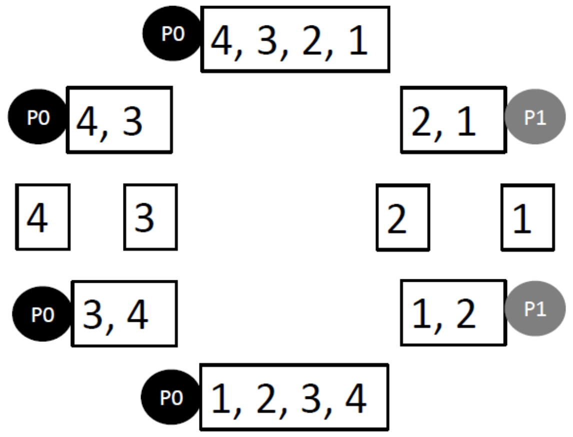


Merge Sort Complexity:

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

Solve the recurrence relation
T(n) = O(nlogn)
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 reponsible for which comparison?
•4,3,2,1



Parallel Merge Sort Complexity:
•Merge sort, O(nlogn)
•Easy way to remember complexity, n (elements) x logn (tree depth)
•If we have n processors, O(logn)

- **Test Cases:**

Students should write test cases depending on their input.

- **Conclusion :**

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

HPC Assignment No: - A5

TITLE	K Nearest Neighbors Classifier using multithreading in java.
PROBLEM STATEMENT/DEFINITION	Parallel Implementation of the K Nearest Neighbors Classifier.
OBJECTIVE	12. To understand concept of K Nearest Neighbors classifier. 13. To understand concept of parallel algorithm. 14. 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. Multithreading in java.
REFERENCES	• Ananth Grama, Anshul Gupta, George Karypis, and Vipin Kumar, "Introduction to Parallel Computing", 2nd edition, Addison-Wesley, 2003, ISBN: 0-201-64865-2. • Java-The Complete Reference by Herbert Schildt, 7th edition. • Introduction to Data Mining and Knowledge Discovery, Third Edition by Two Crows Corporation, pdf
INSTRUCTIONS FOR WRITING JOURNAL	**Date** **Assignment no.** **Problem definition** **Learning objective** **Learning Outcome** **Concepts related Theory** **Related Mathematics** **Algorithm.** **Test Cases** **Conclusion and applications**

Assignment A5

34. Aim

Write a program to design and implement parallel K Nearest Neighbors Classifier algorithm.

35. Prerequisites

36. Concept of existing sequential algorithms.
37. Concept of High Performance Computing.

38. Learning Objectives

39. To understand concept of K Nearest Neighbors Classifier based on sequential algorithm.
40. To understand concept of parallel algorithm.
41. To compare performance by varying number of threads used and also with sequential algorithm.

42. Learning Outcome

After successfully completing this assignment, you should be able to

43. Display result for parallel K Nearest Neighbors Classifier.

44. Analyze performance by varying number of threads used and also with sequential algorithm.

- **Algorithm** :

Step 1: Initialize value of K.
Step 2: Calculate distance between input sample and training samples.
Step 3: Sort the distances.
Step 4: Take top K- nearest neighbors.
Step 5: Apply simple majority.
Step 6: Predict class label for input sample.

- **Concepts related Theory :-**
  K-Nearest Neighbor or KNN algorithm is part of supervised learning that has been used in many applications including data mining, statistical pattern recognition, and image processing. To identify the class of an input, the algorithm chooses the class to which the majority of the input's k closest neighbors belong to. The KNN algorithm is considered as one of the simplest machine learning algorithms.

KNN is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure. In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression:

- In k-NN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

- In k-NN regression, the output is the property value for the object. This value is the average of the values of its k nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms.

The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the k-NN algorithm is that it is sensitive to the local structure of the data. The algorithm has nothing to do with and is not to be confused with k-means, another popular machine learning technique.

- **Test Cases:**

Students should write test cases depending on their input.

Data Analytics Assignment 1:

TITLE	**Summary statistics,data visualization and boxplot for the features on the Iris dataset or any other dataset.**
PROBLEM STATEMENT / DEFINITION	Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris) Use Python/R and Perform following: ● How many features are there and what are their types (e.g., numeric, nominal)? ● Compute and display summary statistics for each feature available in the dataset. (e.g. minimum value, maximum value, mean, range, standard deviation, variance and percentiles ● Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram. ● Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.
OBJECTIVE	• Learn to use dataset, dataframes, features of dataset in an application • Learn to compute summary statistics for the features. • Learn to use visualization techniques.
S/W PACKAGES AND HARDWARE APPARATUS USED	15. Operating System : 64-bit Open source Linux or its derivative 16. Programming Languages: PYTHON/R
FERENCES	• Mark Gardner, “Beginning R: The Statistical Programming Language”, Wrox Publication, ISBN: 978-1-118-16430-3 • David Dietrich, Barry Hiller, “Data Science and Big Data Analytics”, EMC education services, Wiley publications, 2012, ISBN0-07-120413-X • Luis Torgo, “Data Mining with R, Learning with Case Studies”, CRC Press, Talay and Francis Group, ISBN9781482234893
STEPS	Refer to theory, algorithm, test input, test output
INSTRUCTIONS FOR WRITING JOURNAL	45. Date 46. Assignment no. 47. Problem definition 48. Learning objective 49. Learning outcome 50. Related Mathematics 51. Concepts related Theory 52. Test cases

	53. Program code with proper documentation. 54. Output of program. 55. Conclusion and applications (the verification and testing of outcomes)

DA Assignment No. A1

- **Aim:**

Summary statistics, data visualization and boxplot for the features on the Iris dataset or any other dataset.

- **Problem Statement / Definition:**

  - Download the Iris flower dataset or any other dataset into a DataFrame. (eg https://archive.ics.uci.edu/ml/datasets/Iris) Use Python/R and Perform following:
    - ➢ How many features are there and what are their types (e.g., numeric, nominal)?
    - ➢ Compute and display summary statistics for each feature available in the dataset. (eg. minimum value, maximum value, mean, range, standard deviation, variance and percentiles
    - ➢ Data Visualization-Create a histogram for each feature in the dataset to illustrate the feature distributions. Plot each histogram.
    - ➢ Create a boxplot for each feature in the dataset. All of the boxplots should be combined into a single plot. Compare distributions and identify outliers.

- **Prerequisites**

  Database management system, Python/R programming

- **Learning Objectives**
  - Learn to use dataset, dataframes, features of dataset in an application
  - Learn to compute summary statistics for the features.
  - Learn to use visualization techniques.

- **Learning Outcome:**

Students will be able to compute statistics on the features of the dataset, use histograms and boxplot on the features of the dataset.

- **Related Mathematics**

Mathematical Model

Let S be the system set:

S = {s; e;X; Y; Fme;DD;NDD; Fc; Sc} where Dataset is loaded into the dataframe

s=start state

e=end state i.e. Summary statistics for each feature is computed.

X=set of inputs
X = {X1}
where
X1 = IRIS or any other dataset
where ,
Y=set of outputs
1) Number of features and their types.
2) Summary statistics of the each feature (minimum value, maximum value, mean, range, standard deviation, variance and percentiles)
3) Data Visualization- histogram for each feature in the dataset , boxplot for each feature in the dataset
Fme is the set of main functions
Fme = {f1,f2,f3}
where
f1 = function to load dataset into dataframe
f2 = function to to get number of features
f3 = function to get feature type
f3 = function to get minimum,maximum,mean,range,standard deviation,variance and percentile for each feature
f4 = function to draw histogram for each feature
f5 = function to draw boxplot for each feature
DD= Deterministic Data
IRIS dataset
NDD=Non-deterministic data
No non deterministic data
Fc =failure case:
No failure case identified for this application

- **Theory:**

Data analysis is a process of inspecting, cleansing, transforming, and modelling data with the goal of discovering useful information, informing conclusions, and supporting decision-making. Data analysis has multiple facets and approaches, encompassing diverse techniques under a variety of names, while being used in different business, science, and social science domains.
A data set (or dataset) is a collection of data. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question.

Mean, standard deviation, regression, sample size determination and hypothesis testing are the fundamental data analytics methods.

Mean: The sum of all the data entries divided by the number of entries.

$$\text{Population Mean: } \mu = \frac{\Sigma x}{N}$$

$$\text{Sample Mean: } \overline{x} = \frac{\Sigma x}{n}$$

Range: The difference between the maximum and minimum data entries in the set.

Range = (Max. data entry) – (Min. data entry)

Standard deviation:

The standard deviation measure variability and consistency of the sample or population. In most real-world applications, consistency is a great advantage. In statistical data analysis, less variation is often better.

$$\text{Population Standard Deviation} = \sigma = \sqrt{\frac{\sum(x-\mu)^2}{N}}$$

$$\text{Sample Standard Deviation} = s = \sqrt{\frac{\sum(x-\overline{x})^2}{n-1}}$$

Variance: The average squared deviation from the mean is also known as the variance.

Percentile: Let p be any integer between 0 and 100. The pth percentile of data set is the data value at which p percent of the value in the data set are less than or equal to this value.

- How to calculate percentiles: Use the following steps for calculating percentiles for small data sets.
- Step 1: Sort the data in ascending order (from smallest to largest)
- Step Step 3: 2: Calculate ith = $\left(\frac{p}{100}\right) n,$ the 100 where p is the percentile and n is the sample size.

  Step 3: If i is an integer the pth percentile is the mean of the data values in position i and i+1.If i is not an integer then round up to the next integer and use the value in this position.

R commands:

- R command to load dataset from an URL.

```
url<- "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
filename<-"./iris.csv"
        download.file(url=url, destfile = filename, method ="curl")
```

- To get number of rows in the dataset:

```
nrow(dataset)
```

- To get number of features in the dataset:

ncol(dataset)

- To get minimam in the column: min(dataset$column_name)
- To get maximam in the column: max(data$column_name)
- To get mean in the column:colMeans(x=dataset, na.rm = TRUE)
- To get range in the column: range(as.data.frame(dataset[,col], drop=false))
- To get standard deviation and variance in the dataset:

  apply(dataset, 2, sd)
  apply(dataset,2,var)

- **Test data:**

  Iris data from https://archive.ics.uci.edu/ml/datasets/Iris dataset.

Data Analytics : Assignment No. A2

TITLE	**Naive Bayes algorithm for classification on Pima Indians Diabetes dataset.**
PROBLEM STATEMENT / DEFINITION	Download Pima Indians Diabetes dataset. Use Naive Bayes Algorithm for classification ● Load the data from CSV file and split it into training and test datasets. ● summarize the properties in the training dataset so that we can calculate probabilities and make predictions. ● Classify samples from a test dataset and a summarized training dataset.
OBJECTIVE	● Learn Naive Bayes algorithm ● Learn to use Naive Bayes algorithm for classification on given dataset
S/W PACKAGES AND HARDWARE APPARATUS USED	17. Operating System : 64-bit Open source Linux or its derivative 18. Programming Languages: PYTHON/R
REFERENCES	• Mark Gardner, "Beginning R: The Statistical Programming Language", Wrox Publication, ISBN: 978-1-118-16430-3 • David Dietrich, Barry Hiller, "Data Science and Big Data Analytics", EMC education services, Wiley publications, 2012, ISBN0-07-120413-X • Luis Torgo, "Data Mining with R, Learning with Case Studies", CRC Press, Talay and Francis Group, ISBN9781482234893
STEPS	Refer to theory
INSTRUCTIONS FOR WRITING JOURNAL	56. Date 57. Assignment no. 58. Problem definition 59. Learning objective 60. Learning outcome 61. Related Mathematics 62. Concepts related Theory 63. Test cases 64. Program code with proper documentation. 65. Output of program. 66. Conclusion and applications (the verification and testing of

	outcomes)

Assignment No. A2

- **Aim:**
  Naive Bayes algorithm for classification on Pima Indians Diabetes dataset.
- **Problem Statement / Definition:**
  - Download Pima Indians Diabetes dataset. Use Naive Bayes Algorithm for classification
  - Load the data from CSV file and split it into training and test datasets.
  - summarize the properties in the training dataset so that we can calculate probabilities and make predictions.
  - Classify samples from a test dataset and a summarized training dataset.

- **Prerequisites**
  Database management system, Python/R programming

- **Learning Objectives**
  - Learn Naive Bayes algorithm
  - Learn to summarize the properties in the training dataset.
  - Learn to split dataset into training and test datasets
  - Learn to classify samples from a test dataset and a summarized training dataset.

- **Learning Outcome:**

Students will be able to summarize the properties of dataset, split dataset into training and test data, apply Naive Bayes algorithm for classification application.

- **Related Mathematics**

Mathematical Model

Let S be the system set:
S = {s; e;X; Y; Fme;DD;NDD; Fc; Sc} where Dataset is loaded into the dataframe
s=start state
e=end state i.e. classification of samples from the test dataset
X=set of inputs
X = {X1}
where
X1 = Pima Indians Diabetes dataset
where ,
Y=set of outputs
1) Splitting of dataset into training and test datasets
2) Naive Bayes classifier

Fme is the set of main functions
Fme = {f1,f2,f3}
where
f1 = function to load dataset into dataframe
f2 = function to split dataset into training and test datasets
f3 = function to invoke Naive Bayes classifier
DD= Deterministic Data
PIMA Indians diabetes dataset
NDD=Non-deterministic data
null values in the dataset
Fc =failure case:
Failed to classify the record into correct class

- **Theory:**

  The Bayesian Classification represents a supervised learning method as well as a statistical method for classification. Assumes an underlying probabilistic model and it allows us to capture uncertainty about the model in a principled way by determining probabilities of the outcomes. It can solve diagnostic and predictive problems. This Classification is named after Thomas Bayes, who proposed the Bayes Theorem. Bayesian classification provides practical learning algorithms and prior knowledge and observed data can be combined. Bayesian Classification provides a useful perspective for understanding and evaluating many learning algorithms. It calculates explicit probabilities for hypothesis and it is robust to noise in input data.

  Example

  Consider a weather dataset in figure 1.

  Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for playing golf.

  The dataset is shown in figure 1. The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

  - Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
  - Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

Outlook	Temperature	Humidity	Windy	Play Golf	
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Figure 1

Assumption:

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

With relation to dataset, this concept can be understood as:

- Assume that no pair of features are dependent. For example, the temperature being 'Hot' has nothing to do with the humidity or the outlook being 'Rainy' has no effect on the winds. Hence, the features are assumed to be **independent**.
- Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can't predict the outcome accuratey. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

Bayes' Theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ?

- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.

- P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
- P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes' theorem in following way:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

where, y is class variable and X is a dependent feature vector (of size *n*) where:

$$X = (x_1, x_2, x_3, \ldots\ldots, x_n)$$

X = (Rainy, Hot, High, False)
y = No

So P(X|y) means, the probability of "Not playing golf" given that the weather conditions are "Rainy outlook", "Temperature is hot", "high humidity" and "no wind".

Naive assumption

Now, its time to put a naive assumption to the Bayes' theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

Hence, we reach to the result:

$$P(y|x_1, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)\ldots P(x_n|y)P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

which can be expressed as:

$$P(y|x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1, \ldots, x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now, create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

Now calculate P(y) and P(x_i | y).

P(y) is also called **class probability** and P(x_i | y) is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(x_i | y).

Now apply the above formula on weather dataset.

Find P(x_i | y_j) for each x_i in X and y_j in y.

So, in the figure 1 above, calculate P(x_i | y_j) for each x_i in X and y_j in y. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, find class probabilities (P(y)). For example, P(play golf = Yes) = 9/14.

So now the classifier is ready!

Now test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, it can be ignored and find proportional probabilities as:

P(Yes|today) ∝ 2/9 · 2/9 · 6/9 · 6/9 · 9/14 ≃ 0.0141

And

P(No|today) **∝ 3/5 · 2/5 · 1/5 · 2/5 · 5/14 ≃ 0.0068**

Now, since

P(Yes|today) + P(No|today) =1

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

$$P(Yes|today) = \frac{0.0141}{0.0141+0.0068} = 0.67$$

and

$$P(No|today) = \frac{0.0068}{0.0141+0.0068} = 0.33$$

Since

$$P(Yes|today) > P(No|today)$$

So, prediction that golf would be played is ‘Yes’.

The method is applicable for discrete data. In case of continuous data, need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

Gaussian Naive Bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below in figure 2:

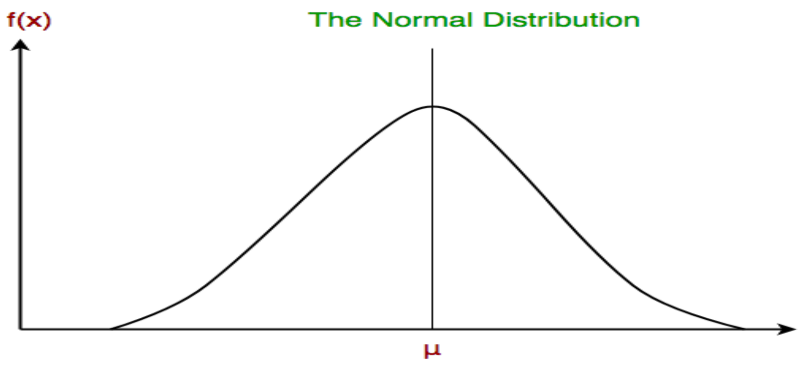


Figure 2.

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Other popular Naive Bayes classifiers are:

- **Multinomial Naive Bayes**: Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.
- **Bernoulli Naive Bayes**: In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

R commands:
Commands to include necessary libraries:

```
library(caret)

library("e1071")
library(klaR)

library(GGally)
```

- Command to split dataset into training and test data:
  labels<-allData[,#col] #labels
  To partition the data (80% training, 20% testing)
  trainingData <- createDataPartition(y=labels, p=.8, list=FALSE)

- Command to use a Naive Bayes model for the classifier ('nb'):

  classifierModel <- train(trainingFeatures, trainingLabels, 'nb',

```
trControl = trainControl(method='cv', number=10))
```

- **Test data:**
  Pima Indians Diabetes dataset

AI Assignment 08	
ASSINGMENT NO.	8
TITLE	Artificial Intelligence & Robotics
PROBLEM STATEMENT /DEFINITION	Solve following 6-tiles problem stepwise using A* algorithm, Initial Configuration:

B	W	B	W	B	W	

Final Configuration:

B	B	B	W	W	W	

Constraint: Tiles can be shifted left or right 1 or 2 positions with cost 1 and 2 respectively.

OBJECTIVE	To learn and implement A* algorithm
OUTCOME	Students will be able to implement A* algorithm for 6-tiles problem
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System recommended :- 64-bit Open source Linux or its derivative Programming Languages: Prolouge-python
REFERENCES	1. Deepak Khemani, “A First Course in Artificial Intelligence”, McGraw Hill Education(India), 2013, ISBN : 978-1-25-902998-1 2. Elaine Rich, Kevin Knight and Nair, “Artificial Intelligence”, TMH, ISBN-978-0-07- 008770-5 3
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 9. Conclusion/Analysis

A*SearchAlgorithm:
A* Search algorithm is one of the best and popular technique used in path-finding and graph traversals.
Explanation:
Consider a square grid having many obstacles and we are given a starting cell and a target cell. We want to reach the target cell (if possible) from the starting cell as quickly as possible. Here A* Search Algorithm comes to the rescue.
What A* Search Algorithm does is that at each step it picks the node according to a value-‘**f**’ which is a parameter equal to the sum of two other parameters – ‘**g**’ and ‘**h**’. At each step it picks the node/cell having the lowest ‘**f**’, and process that node/cell.
We define ‘**g**’ and ‘**h**’ as simply as possible below:
g = the movement cost to move from the starting point to a given square on the grid, following the path generated to get there.
h = the estimated movement cost to move from that given square on the grid to the final

destination. This is often referred to as the heuristic, which is nothing but a kind of smart guess. We really don’t know the actual distance until we find the path, because all sorts of things can be in the way (walls,

water, etc.). There can be many ways to calculate this ‘h’ which are discussed in the later sections.

It is best-known form of Best First search. It avoids expanding paths that are already expensive, but expands most promising paths first.

$f(n) = g(n) + h(n)$, where

$g(n)$ the cost (so far) to reach the node

$h(n)$ estimated cost to get from the node to the goal

$f(n)$ estimated total cost of path through n to goal. It is implemented using priority queue by increasing $f(n)$.

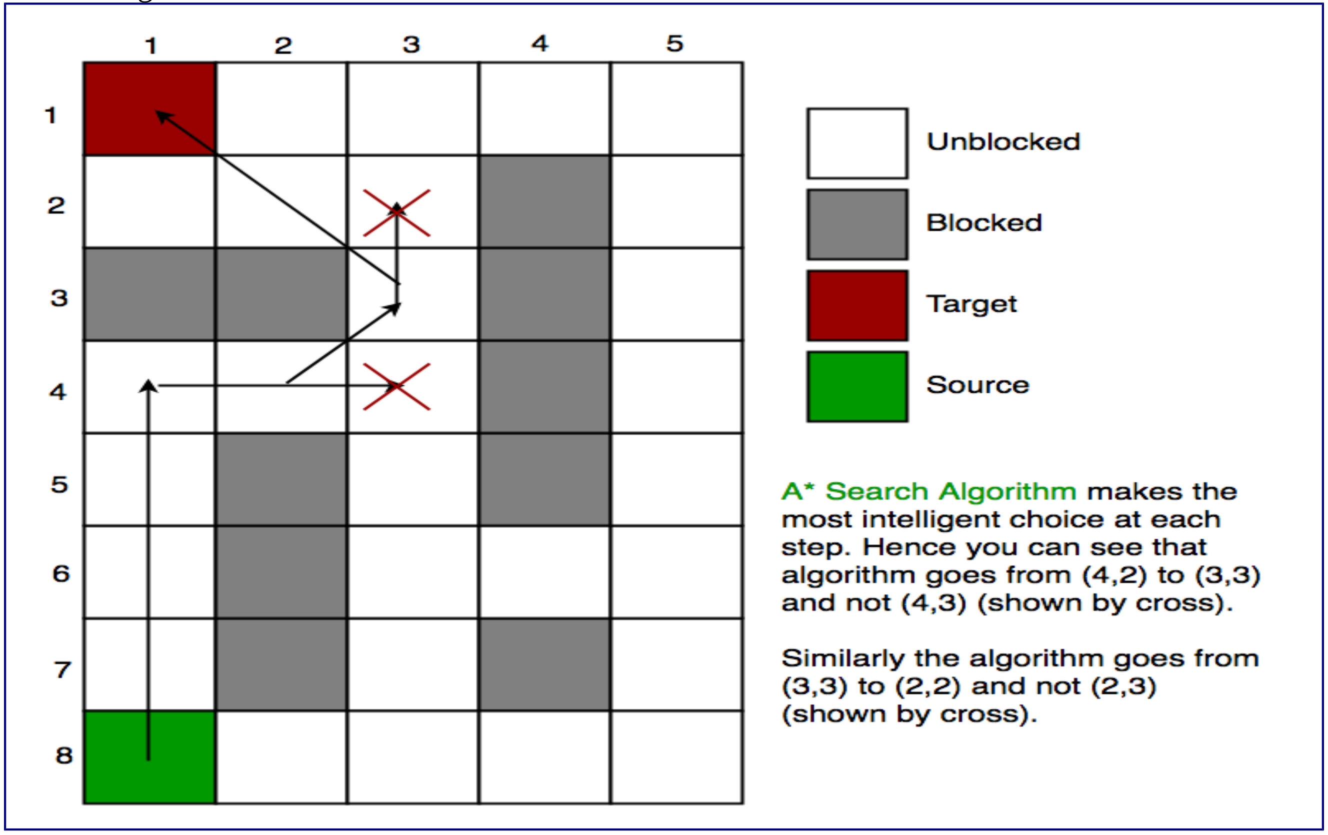
Algorithm:

We create two lists – Open List and Closed List (just like Dijkstra Algorithm)

```
// A* Search Algorithm
1.  Initialize the open list
2.  Initialize the closed list
    put the starting node on the open
    list (you can leave its f at zero)

3.  while the open list is not empty
    a) find the node with the least f on
       the open list, call it "q"

    b) pop q off the open list

    c) generate q's 8 successors and set their
       parents to q

    d) for each successor
        i) if successor is the goal, stop search
          successor.g = q.g + distance between
                              successor and q
          successor.h = distance from goal to
          successor (This can be done using many
          ways, we will discuss three heuristics-
          Manhattan, Diagonal and Euclidean
          Heuristics)

          successor.f = successor.g + successor.h

        ii) if a node with the same position as
            successor is in the OPEN list which has a
           lower f than successor, skip this successor

        iii) if a node with the same position as
            successor  is in the CLOSED list which has
            a lower f than successor, skip this successor
            otherwise, add  the node to the open list
     end (for loop)
```

e) push q on the closed list
end (while loop)

So suppose as in the below figure if we want to reach the target cell from the source cell, then the A* Search algorithm would follow path as shown below. Note that the below figure is made by considering Euclidean Distance as a heuristics.



1
2
3
4
5
1
2
3
4
5
6
7
8
Unblocked
Blocked
Target
Source
A* Search Algorithm makes the most intelligent choice at each step. Hence you can see that algorithm goes from (4,2) to (3,3) and not (4,3) (shown by cross).
Similarly the algorithm goes from (3,3) to (2,2) and not (2,3) (shown by cross).

Advantages:
1.It is the best one from other techniques.
2.It is used to solve very complex problems.
3.It is optimally efficient, i.e. there is no other optimal algorithm guaranteed to expand fewer nodes than A*.
4.It is complete and optimal.
Disadvantages:
1.This algorithm is complete if the branching factor is finite and every action has fixed cost.
2.The speed execution of A* search is highly dependant on the accuracy of the heuristic algorithm that is used to compute h (n).
3.It has complexity problems.

Review Questions:
1. What is the other name of informed search strategy?
2. How many types of informed search method are in artificial intelligence?
3.Which search uses the problem specific knowledge beyond the definition of the problem?
4. Which function will select the lowest expansion node at first for evaluation?
5.Which search is complete and optimal when h(n) is consistent?
6.Which is used to improve the performance of heuristic search?
7.Which search method will expand the node that is closest to the goal?

8.Is A∗'s search behavior necessarily “exponentially explosive”?
9.If h() is admissible and s is the start node, how is h(s) related to the cost of the solution found by A∗search?
10.What happens if we use a heuristic h() in A∗ search that does not have the guarantee that $h(n) \leq h*(n)$ for all states n.

AI ASSINGMENT NO.	10
TITLE	Artificial Intelligence & Robotics.
PROBLEM STATEMENT /DEFINITION	Use Heuristic Search Techniques to Implement Hill-Climbing Algorithm.
OBJECTIVE	To learn and implement Hill-Climbing Algorithm.
OUTCOME	Students will be able to implement Hill-Climbing Algorithm using Heuristic Search Techniques
S/W PACKAGES AND HARDWARE APPARATUS USED	Operating System recommended :- 64-bit Open source Linux or its derivative Programming Languages: Prolouge-python
REFERENCES	1. Deepak Khemani, "A First Course in Artificial Intelligence", McGraw Hill Education(India), 2013, ISBN : 978-1-25-902998-1 2. Elaine Rich, Kevin Knight and Nair, "Artificial Intelligence", TMH, ISBN-978-0-07- 008770-5 3
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 9. Conclusion/Analysis