PUNE INSTITUTE OF COMPUTER TECHNOLOGY

DHANKAWADI, PUNE - 411 043.

ROLL NO 4204 DIV. 2 YEAR 2018 - 2019 SEMISTER VII

LIST OF LABORATORY EXPERIMENTS / REPORTS / COMPUTER PROGRAMMING / WORKSHOP PRACTICE

Sr. No.	TITLE	Date	Page	Signature	Remarks
1.	Parallel Reduction using CUDA	20/7/18	1.		
2.	Vector and Matrix Operations using CUDA	20/7/18	7.		
3.	Parallel sorting Algorithms using OpenMP	20/7/18	11		
4.	Parallel searching Algorithms using Open-MPI	20/7/18	17		
5.	HPC Mini Project Report (CUDA OpenGL Interoperability)	15/10/18	23		
6.	Analysis on Iris Flower Dataset	03/09/18	47		
7.	Näive-Bayes Algorithm	03/09/18	51		
8.	Hadoop Programming (WordCount application)	03/09/18	55		
9.	Trip History Analysis	03/09/18	61		
10.	Predicting Purchase Patterns DA Mini Project Report.	15/10/18	67		
11.	A* algorithm : 8-puzzle	10/9/18	81.		
12.	Alpha-beta pruning	10/9/18	85		
13.	Goal Stack Planning	10/09/18	89		

17/10/18

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

DHANKAWADI, PUNE - 411 043.

ROLL NO 4204 DIV. 2 YEAR 2018 - 2019 SEMESTER 7

LIST OF LABORATORY EXPERIMENTS / REPORTS / COMPUTER PROGRAMMING / WORKSHOP PRACTICE

Sr. No.	TITLE	Date	Page	Signature	Remarks
14	Heuristic Search Algorithm (BFS and A* search)	10/09/18	93		
15.	AIR MiniProject Report (Constraint Satisfaction using N-Queens problem)	15/10/18	101.	17 Nov	

Page No.
Date : / /

Assignment A1

Title : Parallel Reduction using CUDA

Problem statement :

a) Implement parallel reduction using min, max, sum and average operations.

b) Write a CUDA program that, given the N-number vector, find
- Maximum element in the vector
- Minimum element in the vector
- Arithmetic mean of the vector
- Standard deviation of 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.

Objectives :
- To learn parallel programming concepts.
- To learn parallel computing using CUDA.

Outcomes :

We will be able to -
- learn parallel computing concepts using CUDA.

Requirements :
- OS : Fedora 20 / Ubuntu (64-bit)
- Nvidia GPU (GeForce 920 M)
- CUDA API with C/C++ (nvcc compiler)

Page No.
Date : / /

Mathematical model :

Let S be the system set.

$$S = \{ S', e, X, Y, F_{me}, DD, NDD, F_c, S_c \}$$

where

S' = start state

e = end state

X = set of inputs

Y = output set = { min, max, average, std_dev }

DD = deterministic data

NDD = non-deterministic data

F_c = failure case

F_{me} = set of functions = $\{ F_1, F_2, F_3, F_4 \}$

Theory :

- CUDA :
  - CUDA is a parallel computing platform and API model created by NVIDIA.
  - It enables programmers to use a CUDA-enabled GPU for general purpose processing.
  - The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements, for the execution of compute kernels.

- CUDA was initially released in 2007. by NVIDIA Corporation
- CUDA 8.0 comes with following libraries :
  - CUDART : CUDA Runtime library
  - CUDLAS : CUDA Basic Linear Algebra Subroutines Library
  - CUDFFT : CUDA Fast Fourier Transform library.

CUDA Programming :
- nvcc compiler is used for compilation. It separates both the host code (CPU) and device code (GPU) in compilation phase.
- Source code file for CUDA has .cu extension.

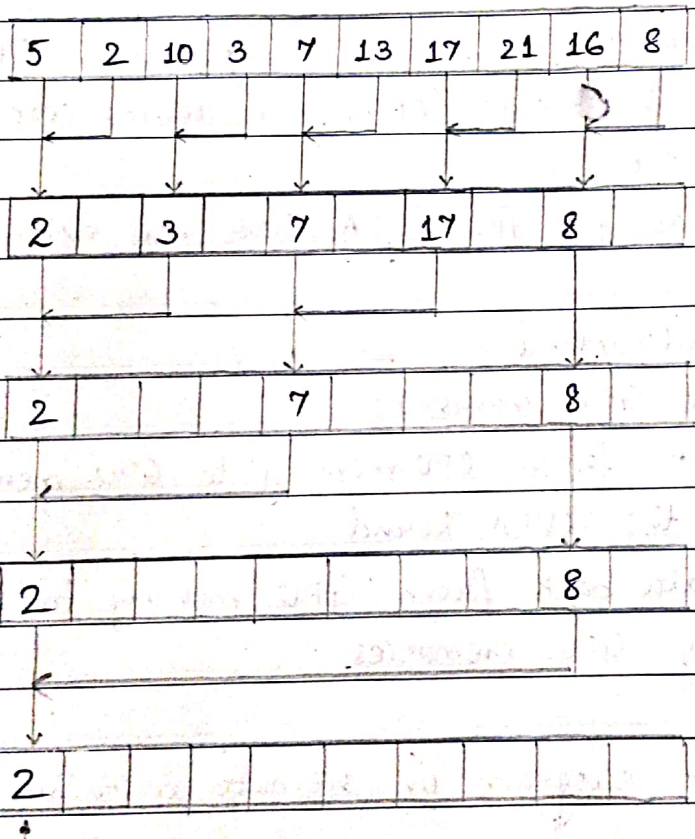
CUDA program structure :
1. Allocate GPU memories
2. Copy data from CPU memory to GPU memory
3. Invoke the CUDA kernel
4. Copy data back from GPU memory to CPU memory
5. Destroy GPU memories.

How to run CUDA program on remote machine :
1. Open terminal.
2. Get login to remote system which has CUDA & GPU.
   eg: student@10.10.15.21
3. Create a CUDA file with .cu extension and write code in it.
4. Compile CUDA program with nvcc compiler.
5. It will create an executable file a.out. Run it.

Page No.
Date : / /

Parallel Reduction :

Suppose we have an array with 10 elements.
- Decompose this array into subgroups of 2 elements.
- Find minimum from each subgroup parallely.
- Repeat this process.



minimum element.

Conclusion:

Thus, we successfully executed the parallel reductions using CUDA.

Page No.
Date : / /

Test Cases and Analysis :

Function	Input size	Sequential Time	Parallel Time	Efficiency
Average	n = 128	0.136	0.129	1.054
	n = 256	0.142	0.123	1.154
	n = 512	0.138	0.120	1.15
Max	n = 128	0.02	0.142	0.014
	n = 1024	0.137	0.113	1.212
	n = 2048	0.135	0.128	1.05
Min	n = 64	0.02	0.189	0.010
	n = 1024	0.134	0.114	1.175
	n = 2048	0.131	0.133	0.98
Sum	n = 512	0.01	0.181	0.055
	n = 1024	0.01	0.113	0.088
	n = 2048	0.01	0.126	0.079
Standard Deviation	n = 64	0.02	0.05	0.4
	n = 256	0.133	0.175	0.76
	n = 1024	0.133	0.158	0.841

$$\text{Efficiency} = \frac{\text{WCSA}}{\text{WCPA}}$$

Here, we observe that as the size of the input increases, parallel algorithm give better performance than the sequential algorithm.

Page No.
Date : / /

Input :

Size of array = 8
Array = 4 0 4 4 3 0 6 1

Output :

Average = 2.750000
Min = 0
Max = 6
Sum = 22
Standard deviation = 2.0463

17/10/18

Page No. 3

Page No.
Date : / /

Assignment A2

Title : Vector and Matrix Operations using CUDA.

Problem Statement :

Design parallel algorithm to :

1. Add two large vectors
2. Multiply vector and matrix
3. Multiply two N×N arrays with n^2 processors.

Objectives :

- To learn about CUDA architecture
- To learn CUDA programming concepts.

Outcomes :

We will be able to -

- learn about CUDA architecture and programming.

Requirements :

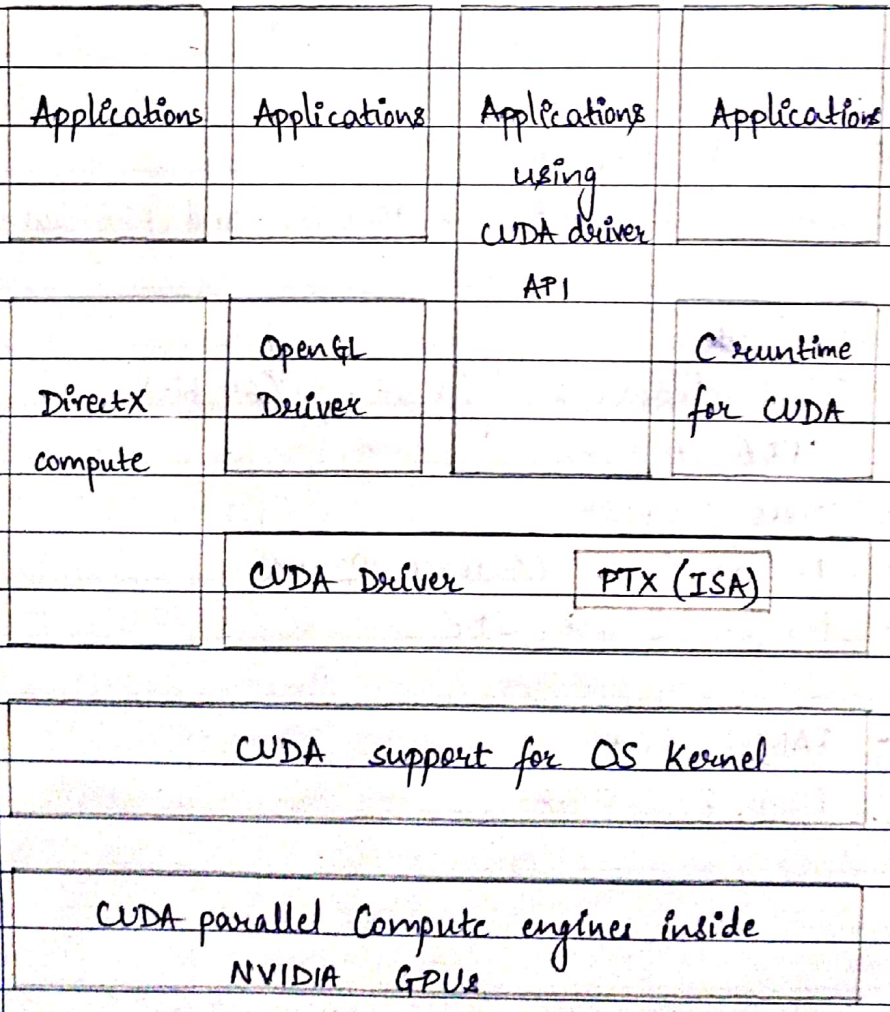
- OS : Fedora 20 / Ubuntu (64-bit)
- CUDA API
- nvcc compiler
- NVidia GPU. (Geforce 920 M)
- Nsight Eclipse IDE

- RAM : 4GB
- HDD : 500 GB

Page No.
Date : / /

Theory :

- CUDA Architecture :
  - The architecture consists of several components as :
    1. Parallel compute engines inside NVIDIA GPU's
    2. OS - kernel - level support
    3. User - mode driver providing device - level - API
    4. PTX instruction set architecture for parallel computing kernels & functions.



Page No.
Date : / /

- Advantages of CUDA:
  1. Unified memory
  2. Scattered reads - code can read from arbitrary addresses in memory
  3. Faster computations to and from GPU
  4. Full support for integer and bitwise operations.
  5. Shared memory between threads.

- Limitations of CUDA:
  1. Newer versions of CUDA are now processed according to C++ syntax rules instead of C rules.
  2. Copying between host and device memory may incur a performance hit due to the system bus bandwidth and latency.
  3. CUDA-enabled GPU's are available from NVIDIA only.
  4. Threads should be running in groups of atleast 32 for best performance.

- CUDA programming:

Let M and N be the input vectors (or matrices) and P be the result obtained from M and N. such that

P = M + N ... vector addition (n-sized vectors)
P = M * N ... vector-matrix multiplication.
P = M * N ... matrix multiplication (nxn order)

Page No.
Date : / /

- With CUDA programming, each element in P can be obtained from one thread.
- As a result, the problem is decomposed into n (or n×n) threads for parallel execution.
- Thread ID's are used to differentiate the data with execution and storing the result.

Conclusion:

Thus, we successfully implemented various vector and matrix operations parallely using CUDA

Test Cases and Analysis:

Operation	Input size	Sequential time	Parallel time	Efficiency
Vector Addition	n = 256	0.01	0.02	0.5
	n = 1024	0.01	0.02	0.5
	n = 2048	0.02	0.01	2.0
Vector Matrix Multiplication	n = 256	0.003	0.082	0.037
	n = 1024	0.093	0.135	0.689
	n = 2048	0.367	0.133	2.759
Matrix Multiplication	n = 256	0.480	0.02	24.0
	n = 1024	0.620	0.133	4.66
	n = 2048	0.754	0.136	5.544

Page No.
Date : / /

Efficiency = $\frac{WCSA}{WCPA}$

Here, we observe that parallel algorithm is a major improvement for matrix-matrix multiplication while it is sufficiently significant for increasing input size for vector matrix multiplication.
No major improvement is gained for vector additions of similar sizes but it gets better for large values of n.

Input :

Vector 1 : 5 7 9 11 4 7 6 2 0 1

Vector 2 : 6 2 7 1 0 4 13 17 12 5

Output :

Sum of vectors : 11 9 16 12 4 11 19 19 12 6.

17/10/18

Page No.
Date : / /

Assignment A3

Title : Parallel sorting Algorithms.

Problem Statement :

Parallel sorting algorithms for bubble sort and merge sort, based on existing sequential algorithms, design and implement utilizing all resources available.

Objective :

- To study parallel execution of sorting algorithms.
- To study OpenMP for parallel computing.

Outcomes :

We will be able to -

- study and understand the parallel sorting algorithms.
- implement algorithms using openMP.

Software and Hardware Requirements :

- OS : Fedora 20 / Ubuntu (64-bit)
- openMP API
- Editor : gedit
- G++ compiler
- RAM : 4 GB
- Hard drive : 500 GB

Page No.
Date : / /

Theory :

- Parallel sorting :

A) Parallel Bubble sort :
- Implemented as a pipeline
- Let local_size = n / no_of_processors.
  We divide the array into blocks, and each process executes the bubble sort on its part, including comparing the last element with the first one belonging to the next thread.
- Implemented with the loop
  for (j = 0 ; j < n-1 ; j++)
- For every iteration of i, each thread needs to wait until previous thread has finished that iteration.
- Synchronization mode to be used is 'barrier'.

B) Parallel Merge Sort :
- Three steps are performed :
  1. Divide
  2. Conquer
  3. Combine
- Collect sorted list onto one processor
- Merge elements as they come together
- Simple tree structure is obtained.

Page No.
Date : / /

Algorithms :

A) Parallel Bubble Sort :

1. for k = 0 to n-2
   1.1. if k is even then
      1.1.1. for i = 0 to (n/2)-1 do in parallel
         1.1.1.1. if A[2i] > A[2i+1] then
            1.1.1.1.1. Swap
      1.1.2. end for
   1.2. else
      1.2.1. for i = 0 to (n/2)-2 do in parallel
         1.2.1.1 if A[2i+1] > A[2i+2] then
            1.2.1.1.1 Swap
      1.2.2. end for.
   1.3. end if
2. end for.

B) Parallel Merge Sort :

1. mid = size / 2
2. if both children present in tree then
   2.1. send mid, firstchild
   2.2. send size - mid, secondchild
   2.3. send list, mid, firstchild
   2.4. send list from mid, size - mid, secondchild
   2.5. call merge (list, 0, mid, list, mid + 1, size, temp, 0, size)

Page No.
Date : / /

2.6. store temp in another array list
3. else
3.1. call parallelMergeSort (list , 0 , size)
4. end if
5. if i>0 then
5.1. send list , size , parent
6. end if.

Analysis :

Time complexity of both algorithms :

Type	Case	Bubble Sort	Merge Sort
Sequential	Best case	$O(n)$	$O(n \log n)$
	Worst case	$O(n^2)$	$O(n \log n)$
	Average case	$O(n^2)$	$O(n \log n)$
Parallel.	Best case	$O(n)$	$O(n)$
	Worst case	$O(n \log n)$	$O(n \log n)$
	Average case	$O(n \log n)$	$O(n \log n)$

Conclusion :

Thus, we successfully implemented parallel bubble sort and merge sort using openMP.

Page No. _____
Date : / /

Test Cases and Analysis :

Sorting	Input size	Sequential Time	Parallel Time	Efficiency
Bubble sort	n = 256	0.020	0.05	0.4
	n = 1024	0.070	0.011	6.36
	n = 2048	0.037	0.018	2.06
Merge sort	n = 256	0.001	0.02	0.05
	n = 1024	0.003	0.03	0.1
	n = 2048	0.002	0.02	0.1

$$\text{Efficiency} = \frac{\text{WCSA}}{\text{WCPA}}$$

We observe that for bubble sort, there was improvement in performance of the algorithm but for merge sort, sequential algorithm proves to be better.

Input :

Input array : 5 9 5 2 4 11 5 1 5 0

Output :

Sorted array : 0 1 2 4 5 5 5 5 9 11

17/10/18

Page No.
Date : / /

Assignment A4

Title : Parallel Searching Algorithms.

Problem Statement :

Design and implement parallel algorithm utilizing all available resources for

- Binary search for sorted array
- Depth First Search (DFS) OR Breadth First Search (BFS) OR Best First Search.

Objective :

- To study and learn about parallel implementation of searching algorithms.
- To learn about MPI API in C/C++

Outcomes :

We will be able to -

- learn about parallel searching techniques
- learn about MPI

Software & Hardware Requirements :

- OS : Fedora 20 / Ubuntu (64-bit)
- GCC / G++ compiler
- Editor : gedit
- MPICC compiler using OpenMPI.
- RAM : 4GB
- HDD : 500 GB

Page No.
Date : / /

Theory :

A> Binary search :

- Binary search, also known as logarithmic search is an algorithm that finds the position of the target value within a sorted array.
- It compares the target value with the middle element of an array. If they are not equal, the half in which target element cannot lie is eliminated and the search continues on the remaining half.
- If the search ends with remaining half being empty, the target is not in the array.
- Binary search runs in logarithmic time in the worst case, making $O(\log n)$ comparisons where n = size of array.

B> Breadth First Search :

- BFS is the most common graph traversal algorithm.
- It starts traversing from the source and traverse the graph layerwise, thus, exploring the neighbor nodes first.
- In sequential implementation, a queue is maintained of the neighbor nodes in each layer.

Page No.
Date : / /

• Open MPI :

- It is a Message Passing Interface library which provides extremely high and competitive performance.
- The Open MPI code has 3 major code modules :
  1. OMPI - MPI code
  2. ORTE - Open Run Time Environment
  3. OPAL - Open Portable Access Layer.
- mpicc compiler is used to compile the C/C++ codes embedded with Open MPI.

×

Algorithms :

A) Parallel Binary Search

parallel_binary_search (sorted_array)

1. Divide the array into M blocks of size n/M.
2. Apply one step of comparison to the middle element of each block.
3. If equality obtained, return address and terminate.
4. Otherwise, identify the adjacent blocks and form a new block starting from the element following the one that signalled (>) and ending at the element preceeding the one that signalled (<)
5. If they are same element, return index.
6. otherwise, parall_binary_search (new_block)

B> Breadth First Search :

```
BFS (Graph root G, source S)

1. enque (S)
2. Mark S as visited.
3. while (Q is not empty)
       // remove the vertex from Q whose neighbor
          will be visited now
    3.1.  V = deque (Q)
    // processing all the neighbors of V
    // W = neighbor of V & neighbors.
    3.2.  if (W is not visited)
              3.2.1.  enque (W)
    3.3.  end if.
4. end while.
```

Conclusion :

Thus, we successfully implemented parallel binary search and breadth first search using Open MPI.

17/10/18

Test Cases and Analysis:

Searching	Input size	Sequential Time	Parallel Time	Efficiency
Binary search (key=54)	n= 1024	1.153	1.542377	0.7477
	n= 2048	1.673	1.236108	1.353
	n= 4096	1.075	0.933585	1.150
Depth First Search Traversal	n= 1024	0.011	0.007	1.57
	n= 2048	0.05	0.019	2.63
	n= 4096	0.109	0.026	4.19

$$\text{Efficiency} = \frac{\text{WCSA}}{\text{WCPA}}.$$

We observe that for increasing values of n, the parallel algorithm performs better than sequential algorithm.

The above calculations are done with keeping the number of processors constant and equal to 5.

Input :

key to be searched = 54

Output :

Original size = 4096

New array size = 3576

Block size per processor = 3576/5 = 715

Key found at index = 15 by Processor No. = 0.

Page No. _____
Date : / /

Assignment A1.

Title : Analysis on Iris Flower Dataset.

Problem statement :

Download the iris flower dataset or any other dataset into a dataframe. Use Python /R and perform following :

1. How many features are there and what are their types?
2. Compare and display summary statistics for each feature available in dataset (eg: min, max, mean, std-dev, variance, percentile)
3. Data visualization - create a histogram for each feature in the dataset to illustrate feature distribution.
4. Create a box plot for each feature in the dataset. All of the box plots should be combined into a single plot. compare distributions and find outliers.

Objective :

- To learn the concept and terminologies in data analytics.
- To learn how to display summary statistics and charts for each feature.

Outcome :

We will be able to -

- learn the concepts in data analytics.
- learn how to summarize and plot charts.

Requirements :

- OS : Windows 10 / Ubuntu (64-bit)
- Python (SciPy libraries) / RStudio with R. libraries.

Page No.
Date : / /

Theory :

A> Iris flower dataset.

- The dataset is a multivariate dataset introduced by the British statistician and biochemist Ronald Fisher in 1936.
- Dataset consists of 50 samples from each of 3 species of Iris, which are sentosa, virginica and versicolor.
- Four features measured from each sample are length and width of sepals & petals in mm.

B> Summary statistics :

1. Mean : It identifies the average value of set of values.

$$\bar{x} = \frac{\sum x_i}{n}$$ where x_i = value of attribute
n = total no of items

2. Range : It shows the mathematical model between the lowest and highest values in the dataset. It measures the variability of dataset.

range = max - min

Page No.
Date : / /

3. Standard Deviation : It measures the variability of dataset like range. The smaller standard deviation indicates less variability.

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

4. Variance : It measures how far the data is spread out.

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

C> Applications :

1. Histogram :
   - It is suitable for visualizing distribution of numeric data over a continous interval, or a certain time period.
   - The histogram organizes large amount of data, and provides a visualization quickly, using a single dimension.

2. Box plot :
   - It allows quick graphical examination of one or more datasets.
   - It may seem primitive than a histogram but they do have some advantages.
   - They take up space and are particularly useful for comparing distributions between several groups of data.

Page No.
Date : / /

3. Data visualization :
   - It quickly creates insightful data visuals.
   - They allow anyone to organize and present information quickly.

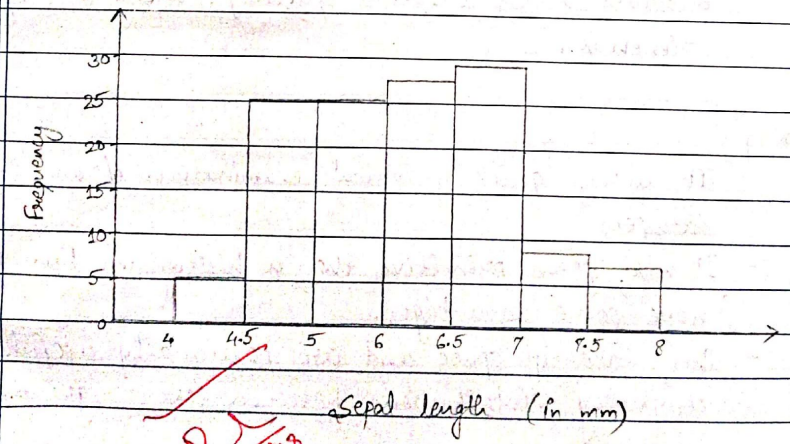
Conclusion :

Thus, we studied about concepts in data analytics, and the dataset. We also presented the data in charts and box plots.

Test case :

Input	Output
Column of sepal length	Mean = 5.843 mm.

Histogram of sepal length :



17/10/18

Page No.
Date : / /

Assignment A2

Title : Näive - Bayes Algorithm.

Problem Statement :

Download Pima Indians Diabetes dataset. Use Näive Bayes algorithm for classification.

1. load the data into csv file and split it into training and test datasets.
2. Summarize properties in the training dataset so that we can calculate probability and make predictions.
3. Classify samples from the test dataset and a summarized training dataset.

Objective :

- To learn classification algorithms like Näive - Bayes.
- To implement such algorithms to predict data.

Outcomes :

We will be able to -

- learn classification algorithms
- make predictions using the training datasets.

Software & Hardware Requirements :

- OS : Windows 10 / Ubuntu (64-bit)
- Python SciPy libraries / RStudio with R libraries.
- Gedit editor.
- RAM : 4 GB
- Hard drive : 500 GB.

Page No.
Date : / /

Theory :

A. Bayes' Theorem :

- It is a way of finding a probability, when we know certain other probabilities.
- Formula :

$$P(A/B) = \frac{P(A) \cdot P(B/A)}{P(B)}$$

where

$P(A/B)$ = how often A happens given that B happens

$P(B/A)$ = how often B happens given that A happens

$P(A)$ = how likely A is on its own.

$P(B)$ = how likely B is on its own.

Example :

If dangerous fires are rare (1%) but smoke is fairly common (10%) due to barbeques, and 90% of dangerous fires make smoke then,

$$P(\text{fire}/\text{smoke}) = \frac{P(\text{fire}) \cdot P(\text{smoke}/\text{fire})}{P(\text{smoke})}$$

$$= \frac{0.01 \times 0.9}{0.1}$$

$$= 9\%$$

∴ Probability of dangerous fire when there is smoke = 9%.

Page No.
Date : / /

B. Näive- Bayes classification :

- It is a simple, yet effective and commonly used, machine learning classifier.
- It is a probabilistic classifier that makes classifications using the maximum Aposteriori decision rule in a Bayesian setting. It can be represented using a very simple Bayesian network.
- It is especially popular for text classification and is a traditional solution for problems such as spam detection.

C. Applications :

1. Real Time Prediction :

Näive - Bayes is an eager learning classifier and it is very fast. Thus, it could be used to make predictions in real time.

2. Multi- class prediction :

This algorithm is also well known for multi- class prediction feature. Here we can predict the probability of multiple classes of target variable.

3. Text classification :

It is used to have higher success rate as compared to other algorithms. As a result, it is widely used in spam filtering and sentiment analysis.

Page No.
Date : / /

Conclusion :

Thus, we successfully learnt and implemented Naïve-Bayes classification algorithm.

Test Case :

Input : Diabetes dataset.

Output :

Confusion Matrix

	0	1
0	125	37
1	25	43

Accuracy = 0.7304

Test set was 30% of the dataset and 73% of predicted values were obtained correctly.

12/11/18

Page No.
Date : / /

Assignment A3

Title : Hadoop Programming

Problem Statement :

Write the hadoop program in Java that counts number of occurences of each word in a text file.

Objective :

- To study Hadoop concepts and basics.
- To implement hadoop program in Java.

Outcomes :

We will be able to -

- Study the basics and concepts of hadoop
- implement hadoop program in Java.

Software & Hardware Requirements :

- OS : Ubuntu / Fedora 20 (64-bit)
- Eclipse IDE
- Java jdk 1.8
- Hadoop single node cluster
- RAM : 4 GB
- HDD : 500 GB
- Web browser : Mozilla Firefox / Google Chrome.

Page No.
Date : / /

Theory :

A. Hadoop :
- It is an open-source framework provided by Apache.
- It is mostly used to store, process and analyze data which is very large in volume.
- It is distributed, highly scalable, fault tolerant.
- It can execute unlimited jobs in parallel concurrently.
- The major components of Hadoop Framework are:
  1. HDFS
  2. MapReduce
  3. YARN
  4. Hadoop Common.

B. Hadoop YARN :
- It is a framework for job scheduling and cluster resource management.

C. Hadoop Common :
- These are the Java libraries and utilities required by other Hadoop modules. These libraries provides file systems and OS level abstractions and contains the necessary Java files required to start Hadoop.

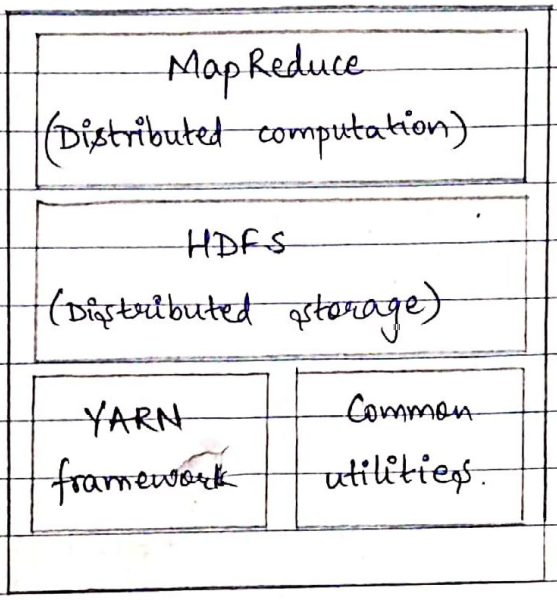


Fig. Hadoop Architecture.

D. MapReduce Algorithm:

- It executes in three stages:
  1. Map stage
  2. Reduce stage
  3. Shuffle stage.

1. Map stage: Mapper's job is to process the input data. The mapper processes the data and creates small chunks of data.

2. Reduce stage:
   - This is the combination of shuffle and reduce stage.
   - The reducer's job is to process the data that

comes from mapper. After processing, it produces a new set of output, which will be stored in HDFS.

- During a MapReduce job, hadoop sends the map & reduce tasks to the appropriate servers in the cluster.
- Most of the computing takes place on nodes with data on local disks that reduces a network traffic.
- After completion of given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.

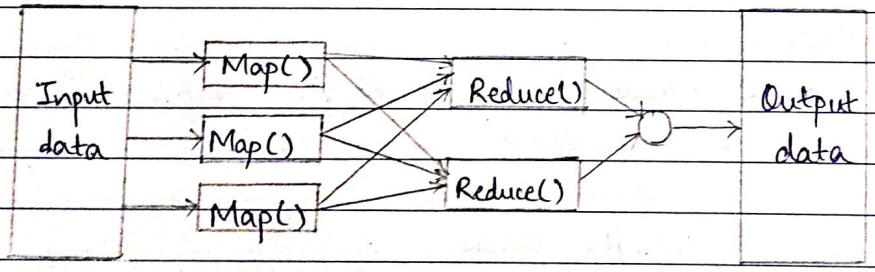


Fig: Hadoop MapReduce

Conclusion:

Thus, we successfully studied and implemented Hadoop concepts and a program for word count in Java.

Test Case :

1. start-all.sh
2. hdfs dfs -mkdir -p /user/hadoop input
3. Create a text file 'sample.txt'

Welcome to PICT
Welcome
PICT
PICT
Welcome

4. hdfs dfs -put sample.txt /user/hadoop/input/
5. cd $HADOOP_HOME
6. hadoop jar share/hadoop/~~xxxx~~mapreduce/hadoop-2.9.0.jar wordcount /user/hadoop/input /user/hadoop/output
7. hdfs dfs -ls /user/hadoop/output
8. hdfs dfs -cat /user/hadoop/output/part-r-00000

PICT 3
to 1
Welcome 3

9. localhost:50070/ can be used to check the HDFS with web GUI
10. localhost:8088/ can be used as Web GUI for execution states.

17/10/18

Page No.
Date : / /

Assignment A4

Title : Trip History Analysis

Problem statement :

Use trip history dataset that is from a bike sharing service in the United states. The data is provided quarter-wise from 2010 (Q4) onwards. Each file has 7 columns. Predict the class of the user.

Objectives :

- To summarize the properties of dataset to calculate probabilities.
- To make predictions based on the data.

Outcomes :

We will be able to -

- summarize the properties of dataset.
- make predictions to identify the class of user.

Requirements :

- OS : Ubuntu / Fedora 20 (64-bit)
- Python compiler
- Python SciPy libraries. / Rstudio with R libraries.
- Editor : gedit
- RAM : 4 GB
- HDD : 500 GB.

Page No.
Date : / /

Theory :

A. Naïve - Bayes :

- It is a probabilistic classification method based on Bayes' theorem with a few changes.
- Bayes' theorem gives relationship between the probabilities of the two events and their conditional properties.
- It assumes that a presence and absence of particular feature of a class is unrelated to the presence or absence of other features.
- The input variables are generally categorical but the variation of an algorithm can accept continuous variables.
- There are also ways to convert continuous variables into categorical ones. This process is often referred to as discretization of continuous variables.

for example :

An object can be classified based on its attributes such as shape, color and weight. A reasonable classification of an object that is spherical, yellow and less than 60 grams in weight may be a tennis ball. Even if these features depend on each other, a Naïve-Bayes classifier considers all these properties to contribute independently to the probability that the object is a tennis ball.

B. Confusion Matrix :

		Predicted class	
		Positive	Negative
Actual class	Positive	True positive	False negative
	Negative	False positive	True negative

- True positives (TP) are the number of positive instances of the classifier currently identified as positives.
- False positives (FP) are the number of instances in which the classifier identified as positive but in reality are negatives.
- True negatives (TN) are the number of negative instances of the classifier currently identified as negative.
- False negatives (FN) are the negative instances classified as negatives, but in reality are positives.

- In a two-phase classification, a present threshold may be used to separate positives from negatives. TP and TN are correct guesses.
- A good classifier should have large TP and TN and small (ideally zero) number for FP and FN.

Page No.
Date : / /

Steps :

1. Download the dataset.
2. Read csv into variables
3. View data on board.
4. Split data into partial data.
5. Make two subsets : Training data and Test data
6. Perform Naïve-Bayes algorithm on data.
7. Predict values of test data.
8. Compare predicted values of Test data
9. Store result in data.
10. View result in tabular form.

Conclusion :

Thus, we have successfully classified the user into various classes through prediction.

Test Case :

Input : Trip History dataset.

Output :

Confusion Matrix

	0	1
0	0	0
1	0	7200

Page No. _____

Date : / /

Accuracy is 1.

Prediction was made whether biker was regular or a casual member.

All the predictions made by the algorithm were correct.

17/10/18

89

Page No.
Date : / /

Assignment A1

Title : A* algorithm

Problem statement :

Solve 8-puzzle problem using A* algorithm. Assume any initial configuration and define goal configuration clearly.

Objectives :

- To learn and understand use and need of A* algorithm.
- To apply A* algorithm to real time problem
- To implement A* algorithm using suitable programming language.

Outcomes :

We will be able to -

- learn about A* algorithm.
- apply A* algorithm to gaming problem.
- implement A* algorithm using Prolog / Python / Java.

Hardware & Software Requirements :

- OS : Fedora 20 / Ubuntu (64-bit)
- RAM : 4 GB
- HDD : 500 GB
- Eclipse IDE
- Java JDK v1.8
- Python libraries.

Page No.
Date : / /

Theory :

- A* is one of the most popular heuristic search algorithm for finding paths in a graph.
- It is really a smart algorithm which separates it from other conventional algorithms.

- Consider a square grid having many obstacles and we are given a starting cell and a target cell.
- We want to reach target cell from the starting cell as quickly as possible.
- What A* algorithm does is at each step, it picks the node according to a value '-f' which is a parameter equal to sum of other two parameters -g and h.
- At each step, it picks the node cell having least 'f' and process that node/cell.

- We define 'g' and 'h' simply as possible
  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 dont know the actual division until we find the path because all sorts of things can be in the way.

Page No. _____

Date : / /

Algorithm:

1. Initialize the open list
2. Initialize the ~~node~~ closed list
   put the starting node on the open list.
3. While the open list is not empty
   1. find the node with the least f on the open list. Call it 'q'.
   2. pop 'q' off open list
   3. generate 'q's successors
   4. for each successor
      1. if successor is the goal, stop search successor.g = q.g + distance(successor.q)
         successor.h = distance from goal to successor
         successor.f = successor.g + successor.h
      2. if a node with the same position as successor is in the OPEN list which has a lower 'f' than successor, skip this successor.
      3. 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, and the node to the open list.
   5. end for
   6. push q on the closed list
4. end while.

Page No.
Date : / /

Test cases :

Initial Configuration :	Final Configuration
1 2 x	1 2 3
4 5 3	4 5 6
7 8 6	7 8 x

Output :

The puzzle was solved in 18 moves.

Conclusion :

We successfully implemented A* algorithm for 8-puzzle problem.

17/10/23

Page No. _____
Date : / /

Assignment A2

Title : Alpha beta pruning

Problem Statement :

Implement alpha beta pruning graphically with example.

Objectives :

- To understand need of alpha beta pruning
- To compare MinMax algorithm with alpha beta pruning so as to get useful insights.
- To implement alpha beta pruning to understand difference visibly.

Outcomes :

We will be able to

- understand alpha beta pruning
- compare MinMax algorithm with alpha beta pruning.

Hardware & Software Requirements :

- OS : Fedora 20 / Ubuntu (64-bit)
- RAM : 4 GB
- Java JDK v1.8
- Editor : Eclipse IDE

Theory :

- Alpha beta pruning is not actually a new algorithm, rather an optimization technique for min max algorithm.

Page No.
Date : / /

- It reduces the computation by a huge factor. This allows us to search much faster and even go into deeper levels in game tree which need not be searched. because there already exist a better move available.
- It is called alpha-beta pruning because it passes two extra parameters in the minmax function namely alpha, beta.
- Alpha is the best value that the maximizer currently can gaurantee at that level or above.
- Beta is the best value that the minimizer currently can gaurantee at that level or above.

Pseudocode :

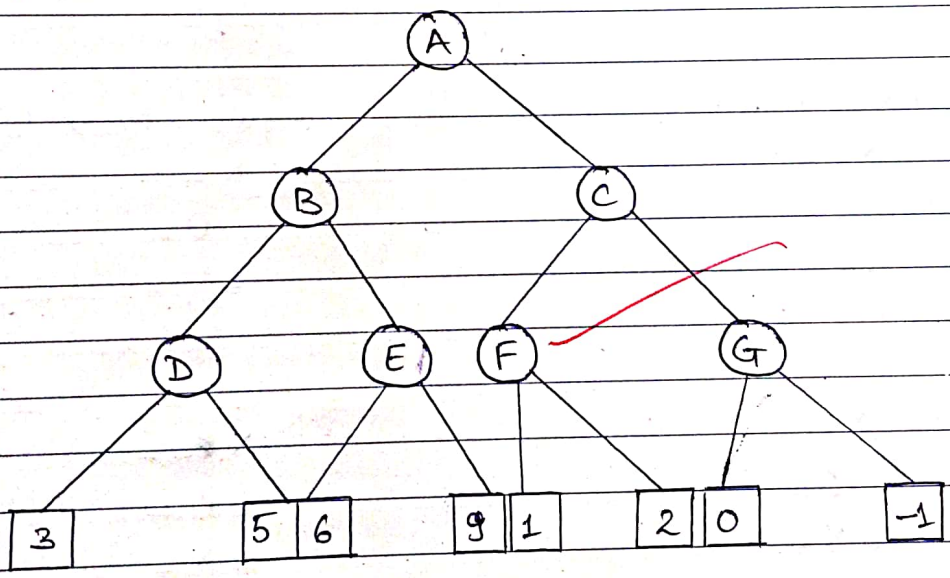
```
function minmax (node, depth, isMax, alpha, beta)

  if node is a leaf node :
        return value of the node
  if isMax :
        bestVal = -INFINITY
        for each child node :
              Value = minmax(node, depth+1, false,
                                    alpha, beta)
              bestVal = max (bestVal, Value)
              alpha = max (alpha, bestVal)
              if beta <= alpha :
                    break
        return bestVal
```

Page No. _____
Date : / / .

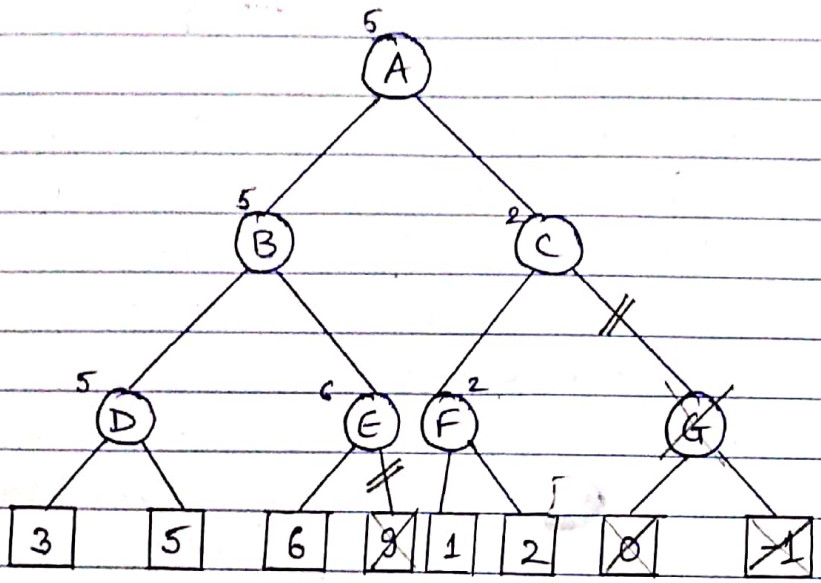
```
else
    bestVal = + INFINITY
    for each child node :
        value = minmax (node , depth + 1 , true , alpha , beta)
        bestVal = min (bestVal , value)
        beta = min (beta , bestVal)
        if beta <= alpha :
            break
return bestVal
```

Consider following game tree :



With the minmax algorithm, all leaf nodes get evaluated.
But with alpha-beta algorithm, nodes get pruned as follows

Implement algorithmic techniques by virtue ofadvance programming skill set and Free and



Test Case :

User Moves	Computer Moves
X _ _	X _ _
1,1 : _ _ _	_ O _
_ _ _	_ _ _
3,2 : X _ _	X O _
_ O _	O O _
_ X _	_ X _
2,3 : X _ _	X _ O
O O X	O O X
_ X _	_ X _

Page No. _____

Date : / /

1,2 :

X X O	X X O
O O X	O O X
_ X _	O X _

Game Over. PC wins.

Conclusion :

We successfully implemented alpha beta pruning using a game of Tic-Tac-Toe.

RL
17/10/18

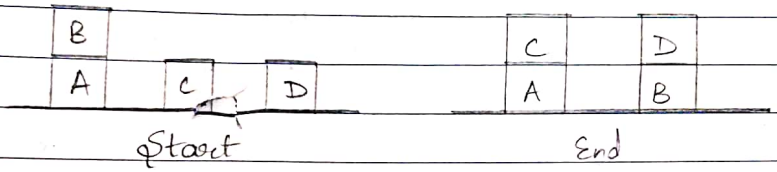
Page No.
Date : / /

Assignment A3

Title : Goal stack Planning

Problem statement :

Implement goal stack planning for the following configuration from the blocks world.



Objectives :

- To learn and understand concept of goal stack planning.
- To study need and real time use of goal stack planning.
- To implement goal stack planning algorithm using suitable programming language.

Outcomes :

We will be able to

- learn the concept of goal stack planning.
- study need and use of goal stack planning.
- implement goal stack planning.

Software and Hardware Requirements :

- OS : Ubuntu / Fedora 20 (64-bit)
- RAM : 4 GB
- Hard Drive : 500 GB
- Java JDK 1.8 / Python libraries / Prolog.
- Editor : gedit.

Page No.
Date : / /

Theory :

- Goal Stack Planning :
  - One of the earliest techniques in planning using goal stack.
  - Problem solver uses single stack that contains
    - sub goals and operates both.
    - sub goals are solved linearly and then finally the conjoined sub goal is solved.
  - Plans generated by this method will contain complete sequence of operations for solving one goal followed by complete sequence of operations for the next, etc.
  - Problem solver relies on
    - A database that describes the current situation.
    - set of operators with pre-condition, add and delete lists.

- Let us assume that goal to be satisfied is
  $GOAL = G_1 \wedge G_2 \wedge G_3 \wedge \ldots \wedge G_N$

- Sub-goals G_1, G_2, G_3, are stacked with compound goal $G_1 * G_2 * G_3 \hat{} \ldots \hat{} G_N$ at the bottom.

Top
G_1
G_2
:
G_N

Bottom
$G_1 \wedge G_2 \wedge G_3 \wedge \ldots \wedge G_N$

Page No.
Date : / /

Algorithm :

1. Find an operator that satisfies sub goal G1 (makes it true) and replace G1 by the operator.
   1. If more than one operator satisfies sub goals then apply some heuristic to choose one.
2. In order to execute the top most operation, its pre-conditions are added onto the stack.
   1. Once the preconditions of an operator are satisfied then we are guaranteed that operator can be applied to produce a new state.
   2. New state is obtained by using ADD and DELETE lists of an operator to the existing database.
3. Problem solver keep track of operations applied.
   1. This process is continued till the goal stack is empty and problem solver returns plan of the problem

Consider given example :

Initial state :

$$ON(B,A) \wedge ONT(C) \wedge ONT(A) \wedge ONT(D) \wedge CL(B) \wedge CL(C) \wedge CL(D) \wedge AE$$

Goal state :

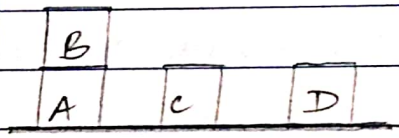
$$ON(C,A) \wedge ON(B,D) \wedge ONT(A) \wedge ONT(D) \wedge CL(C) \wedge CL(B) \wedge AE$$

Page No.

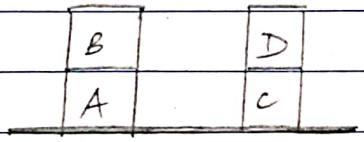
Date : / /

Test Case :

Input



Output



Conclusion :

We successfully implemented goal stack planning in Python to implement the above case.

19/10/18

93

Page No. _____

Date : / /

Assignment A4

Title : Heuristic Search Algorithm

Problem Statement :

Use heuristic search techniques to implement Best First Search (Best solution but not optimal) and A* algorithm (Always gives optimal solution).

Objectives :

- To study both BFS and A* search algorithm
- To understand difference between the two.
- To implement both algorithms using suitable language to prove point of optimality.

Outcomes :

We will be able to

- study both BFS and A* search algorithms
- implement both algorithms using a programming language.

Hardware & Software Requirements :

- OS : Ubuntu / Fedora 20 (64-bit)
- RAM : 4 GB
- Hard drive : 500 GB
- Python 2.7 / 3
- Editor : gedit.

Page No.
Date : / /

Theory :

A. Best First Search (BFS) :

- In BFS, when we are at a node, we can consider any of the adjacent as next node. So both BFS and DFS blindly explore paths without considering any cost function.
- The idea of Best First Search is to use an evaluation function to decide which adjacent is most promising and then explore.
- Best First Search falls under the category of heuristic search or Informed search. We use a priority queue to store costs of nodes.
- So the implementation is a variation of BFS we just need to change Queue to PriorityQueue.

B. A* search algorithm :

- A* search algorithm is one of the best and popular technique used in path-finding and graph traversal.
- Informally speaking, A* search algorithm unlike other traversal techniques, it has "brains". What it means is that it really is a smart algorithm which separates it from the other conventional algorithms.
- This fact is worth mentioning that many game and web-based maps use this algorithm to

Page No.
Date : / /

Algorithm :

A. Best-First-Search (Graph g, Node start)

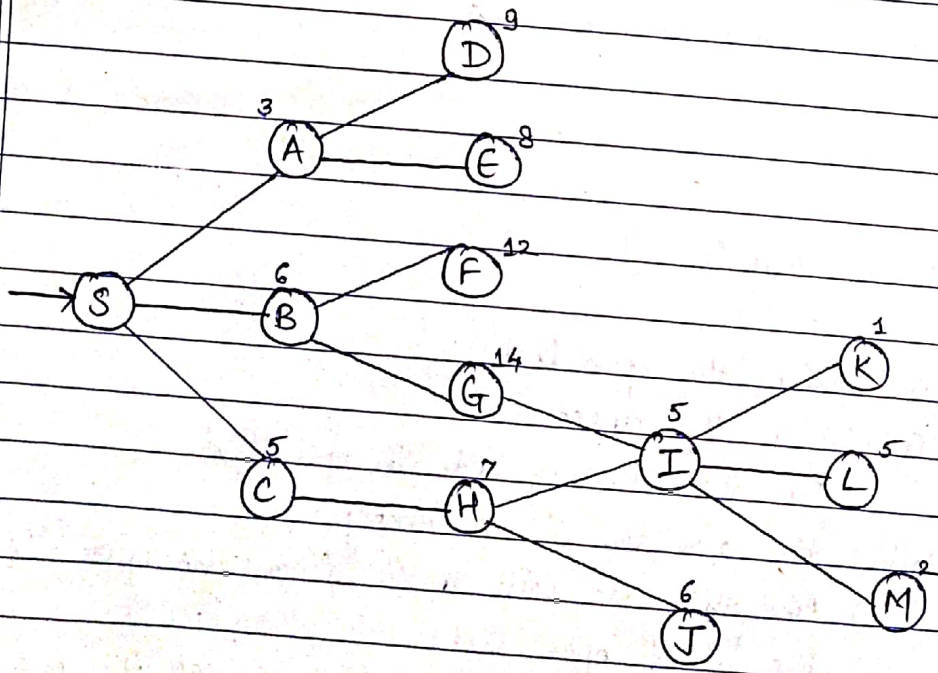
1. Create an empty PriorityQueue pq.
2. Insert "start" in pq
   pq.insert(start)
3. Until PriorityQueue is empty
   u = PriorityQueue.DeleteMin
   If u is the goal
   Exit
   Else
   for each neighbor v of u
   if v = "unvisited"
   mark v = "visited"
   pq.insert(v)
   Mark u = examined.
4. End.

B. A* search Algorithm :

1. Initialize the open list.
2. Initialize the closed list
   Put the starting node in open list.
3. While the open list is not empty
   a) Find the node with least 'f' on the open list, call it 'q'.
   b) Pop 'q' off open list
   c) Generate 'q's successor and set its parent to q.

d) for each successor
   1. if successor is the goal, stop.
      successor.g = q.g + distance bet" successor and q.
   2. successor.h = distance between goal & successor.
   3. successor.f = successor.g + successor.h
   4. if a node with same position as successor is in CLOSED list which has lower f than successor, then skip.
e) end for
f) push q on the closed list
4. end while.

Consider the graph:



Page No.
Date : / /

With Best first search, path traversed is
A – C – H – I with cost = 24
With A* search algorithm, path traversed is
A – B – G – I with cost = 9.

Test Case :

0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0

Input = 0, 0
Goal = 7, 6

Output = [(0,0), (1,1), (2,2), (3,3), (4,3), (5,4), (6,5), (7,6)]

17/10/18

Conclusion :
We successfully implemented Best First Search and A* algorithm with the help of a game of maze.