A1:

object Test{

def main(args: Array[String])

{

println("Enter the List Count:");

var count1 = Console.readInt;

var list = new Array[Int](count1);

for(a<- 0 to (count1 -1))

{

println("Enter the List Element: " + (a+1));

list(a) = Console.readInt;

}

scala.util.Sorting.quickSort(list);

println("\n\n")

for(a<- 0 to (count1-1))

println("Element: "+(a+1)+"--------> "+list(a));

println("Enter the Element to be Searched: ");

var j = Console.readInt;

var i = binarySeachIterative(list, j);

var k = binarySeachRecursive(list, 0 , (count1-1), j);

if(i == (-1) && k == (-1))

println("Element not Found!")

else

println("Element found at position: "+(i+1))

}

def binarySeachIterative(list:Array[Int] , target : Int): Int = {

var left = 0;

var nf = -1;

var right = list.length -1;

while(left <= right){

var mid = left + (right - left)/2;

if(list(mid) == target)

return mid;

else if(list(mid)>target)

right = mid-1;

else

left = mid+1;

}

return nf;

}

def binarySeachRecursive(list:Array[Int] , l:Int,r:Int, target : Int):Int ={

var nf = -1;

if(r>=l){

var mid = l + (r-l)/2;

if(list(mid) == target)

return mid;

if(list(mid)>target)

return binarySeachRecursive(list,l,mid-1,target);

return binarySeachRecursive(list,mid+1,r,target);

}

else

return nf;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*A2:

#include <iostream>

#include <pthread.h>

using namespace std;

int n;

struct parameters{

int \* array;

int firstI;

int lastI;

};

void\* quick(void \* a){

parameters \*p = (parameters \*)a;

int firstI = p->firstI;

int lastI = p->lastI;

pthread\_t id = pthread\_self();

if(firstI<lastI){

int i = firstI;

int j = lastI;

int pivot = p->array[firstI];

while(i<j){

while(p->array[i]<=pivot && i<j)

i++;

while(p->array[j]>pivot && i>=j)

j--;

if(i<=j){

int temp = p->array[i];

p->array[i] = p->array[j];

p->array[j] = temp;

}

}

int temp = p->array[j];

p->array[j] = p->array[p->firstI];

p->array[p->firstI] = temp;

cout<<"Thread id: \t"<<id<<" and Pivot is\t"<<pivot<<" ";

for(int i = firstI;i<=lastI;i++)

cout<<p->array[i]<<" ";

cout<<endl;

pthread\_t threads[2];

parameters p1,p2;

p1.array = new int [n];

p1.array = p->array;

p1.firstI = p->firstI;

p1.lastI = j-1;

p2.array = new int[n];

p2.array = p->array;

p2.firstI = j+1;

p2.lastI = p->lastI;

pthread\_create(&threads[0],NULL,&quick,(void \*)&p1);

pthread\_create(&threads[1],NULL,&quick,(void \*)&p2);

pthread\_join(threads[0],NULL);

pthread\_join(threads[1],NULL);

}

}

int main()

{

parameters a;

cout<<"Enter the array size:\n";

cin>>n;

cout<<"Enter the elements:\n";

a.array = new int[n];

a.firstI = 0;

a.lastI = n-1;

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

cin>>a.array[i];

quick(&a);

cout<<"Sorted array is:\n";

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

cout<<a.array[i]<<"\t";

cout<<endl;

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*A3:

A3.l

%{

#include<string.h>

#include<stdio.h>

#include<math.h>

int line\_no = 1;

FILE\* opfile;

char name[20][20];

int cnt = 0;

int error\_line\_no[20];

char error\_desc[20][100];

int error\_cnt = 0;

%}

%%

[0-9]+ {printf("%d %s NUMBER\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

[-+\*/] {printf("%d %s OPERATOR\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

= {printf("%d %s ASSIGNMENT\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

include|main|return {printf("%d %s KEYWORD\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

int|float|char|double {printf("%d %s DATATYPE\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

[\t] ;

\n {line\_no++;fprintf(opfile,"\n");}

(\/\/.\*) ;

(\/\\*[^\*/]\*\\*\/);

(\/\\*[^\*/]\*) {error\_line\_no[error\_cnt] = line\_no; char str[100] = "Unterminated comment";strcat(str,yytext);strcpy(error\_desc[error\_cnt],str);error\_cnt++;}

# {printf("%d %s PREPROCESSOR\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

; {printf("%d %s TERMINATION\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\< {printf("%d %s LESS THAN\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\> {printf("%d %s GREATER THAN\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\{ {printf("%d %s START OF BLOCK\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\} {printf("%d %s END OF BLOCK\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\( {printf("%d %s OPEN\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\) {printf("%d %s CLOSE\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\[ {printf("%d %s OPEN\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

\] {printf("%d %s CLOSE\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

, {printf("%d %s SEPARATOR\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

printf|scanf {printf("%d %s LIBRARY FUNCTIONS\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

(\"[^\"]\*\") {printf("%d %s STRING CONSTANT\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

(\"[^\"\n]\*\n) {error\_line\_no[error\_cnt]=line\_no; char str[100]="Unterminated quotes ";strcat(str,yytext); strcpy(error\_desc[error\_cnt],str); error\_cnt++; line\_no++;}

([a-zA-Z0-9]+.h) {printf("%d %s HEADER FILE\n",line\_no,yytext); fprintf(opfile,"%s",yytext);}

[a-zA-Z][a-zA-Z0-9]\* {printf("%d %s IDENTIFIER\n",line\_no,yytext);fprintf(opfile,"%s ",yytext);

st\_add(yytext);}

[0-9]+[a-zA-Z]+ {error\_line\_no[error\_cnt] = line\_no; char str[100] = "Unrecognized token";strcat(str,yytext);strcpy(error\_desc[error\_cnt],str);error\_cnt++;}

[a-zA-Z][a-zA-Z0-9]\*[\?@$][a-zA-Z0-9] {error\_line\_no[error\_cnt] = line\_no; char str[100] = "Unrecognized token";strcat(str,yytext);strcpy(error\_desc[error\_cnt],str);error\_cnt++;}

%%

void st\_add(char s[20])

{

int i;

for(i = 0;i<cnt;i++)

{

if(strcmp(name[i],s)==0) return;

}

strcpy(name[cnt],s);

cnt++;

}

main()

{

char ifile[100],ofile[100];

printf("Enter the input file:\n");

scanf("%s",ifile);

yyin = fopen(ifile,"r");

printf("Enter the output file:\n");

scanf("%s",ofile);

opfile = fopen(ofile,"w");

int i;

printf("Line no LEXEME Token\n");

yylex();

printf("Symbol Table:\n");

for(i = 0;i < cnt;i++)

{

printf("\n %s",name[i]);

}

printf("\n Lexical errors found : %d\n",error\_cnt);

for(i = 0;i < error\_cnt; i++)

{

printf("\nLine no %2d \t %s",error\_line\_no[i],error\_desc[i]);

}

return 0;

}

int yywrap()

{

return 1;

}

========================================================================

Abc.c

# include < stdio.h >

# include < iostream.h >

# include < math.h >

main ( )

{

int a , b , c ;

printf ( " Hello" ) ;

int d , e ;

printf ( "HI" ) ;

scanf ( "Enter" ) ;

a = b + 20 ;

if ( )

{

c = d ;

char x , y ;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A4:

A4.l

%{

#include "y.tab.h"

#include <stdio.h>

extern int yylval;

%}

%%

[0-9]+ {yylval=atoi(yytext); return DIGIT; }

int|float|char|double {return DTYPE;}

"#include <"[.a-zA-Z ]\*">\n" {return HEADER;}

"main()"[\n]\* {return MAIN;}

"{"[\n]\* {return STARTB;}

"}"[\n]\* {return ENDB;}

";"[\n]\* {return ENDS;}

"," {return SEP;}

[-+/\*] {return OP;}

"=" {return EQ;}

"(" {return OPB;}

")" {return CLB;}

"if" {return IFC;}

"printf("[^\)]\*")" {printf("Function call\n");return FUNC;}

"scanf("[^\)]\*")" {printf("Function call\n");return FUNC;}

[a-zA-Z][\_a-zA-Z0-9]\* {return VAR;}

[ \t]+ ;

%%

int yywrap()

{return 1;}

========================================================================

A4.y

%{

#include <stdio.h>

extern FILE\* yyin;

%}

%token DIGIT VAR HEADER MAIN STARTB ENDB

%token ENDS DTYPE SEP FUNC OP EQ

%token IFC OPB CLB

%%

St: S MAIN STARTB stmts ENDB {printf("Program with main\n");};

S: HEADER S | HEADER {printf("HEADERs\n");};

stmts: stmt| stmts stmt| ;

stmt: expr|DTYPE varlist ENDS | FUNC ENDS | IFC OPB CLB STARTB stmts ENDB {printf("\nStatement\n");};

varlist: VAR| varlist SEP VAR ;

expr: VAR EQ dv opn ENDS{printf("Expression\n");};

opn: OP dv | OP dv opn|;

dv: DIGIT|VAR;

%%

void yyerror(char\* s)

{

printf("Incorrect Syntax");

}

int main()

{

char ifile[100];

printf("\nEnter input file name: ");

scanf("%s",ifile);

yyin=fopen(ifile,"r");

yyparse();

}

========================================================================

Abc.c

#include <stdio.h>

#include <iostream.h>

#include <math.h>

main()

{

int a,b,c;

printf("Hello");

int d,e;

printf("HI");

scanf("Enter");

a=b+20;

if()

{

c=d;

char x,y;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A5:

A5.l

%{

#include "y.tab.h"

#include "stdlib.h"

#include "string.h"

%}

%%

[\n] {}

[ \t] {}

main {return MAIN;}

int|void|long|char {strcpy(yylval.sval,yytext); return DT;}

([\_a-zA-Z]+[\_a-zA-Z0-9]\*) {strcpy(yylval.sval,yytext); return ID;}

[0-9]+ {strcpy(yylval.sval,yytext); return NUM;}

. {return yytext[0];}

========================================================================

A5.y

%{

#include "stdio.h"

#include "string.h"

extern FILE\* yyin;

struct sym\_row

{

char name[25];

char val[25];

}sym\_tbl[50];

struct quad\_row

{

char oprd1[25];

char oprd2[25];

char res[25];

char op;

}quad\_tbl[50];

struct stack\_item

{

char item[25];

}stk[50];

int sym\_index=0 , quad\_index=0 , temp\_index=0, top= -1;

void insertsymtbl(char\*,char\*);

void insertquadtbl(char\*,char\*,char\*,char);

void push(char\*);

char\* pop();

%}

%union

{

char sval[25];

}

%token MAIN

%token <sval> ID

%token <sval> DT

%token <sval> NUM

%left '+' '-'

%left '\*''/'

%start prg

%%

prg: MAIN '(' ')' '{' slist '}' {printf("Valid program.\n");}

;

slist: vstmnt s

;

vstmnt: varlist vstmnt

|

;

varlist: DT vardec ';'

;

vardec: ID ',' vardec {insertsymtbl($1,"0");}

| ID '=' NUM ',' vardec {insertsymtbl($1,$3);}

| ID '=' NUM {insertsymtbl($1,$3);}

| ID {insertsymtbl($1,"0");}

;

s: stmnt s

|

;

stmnt: ID '=' expr ';' {insertquadtbl(pop()," ",$1,'=');}

| expr ';'

;

expr: expr '+' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'+');

push(temp);

temp\_index++;

}

| expr '-' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'-');

push(temp);

temp\_index++;

}

| expr '\*' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'\*');

push(temp);

temp\_index++;

}

| expr '/' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'/');

push(temp);

temp\_index++;

}

| NUM { push($1);}

| ID { push($1);}

;

%%

int yyerror(char\* err)

{

printf("\nSyntax error.");

}

int yywrap()

{

return 1;

}

int main()

{

FILE\* in=fopen("input.c","r");

yyin=in;

yyparse();

printf("\n-----------------------------------------------------------------------------------");

printf("\nSymbol Table :");

printf("\n-----------------------------------------------------------------------------------");

printf("\nSymbol\t\tValue\n");

int i;

for(i=0;i<sym\_index;i++)

{

printf("%s\t\t%s\n",sym\_tbl[i].name,sym\_tbl[i].val);

}

printf("\n-----------------------------------------------------------------------------------");

printf("\nQuadrapule Table :");

printf("\n-----------------------------------------------------------------------------------");

printf("\nOperand 1\t\tOperand 2\t\tResult\t\tOperator\n");

int j;

for(j=0;j<quad\_index;j++)

{

printf("%s\t\t\t%s\t\t\t%s\t\t\t%c\n",quad\_tbl[j].oprd1,quad\_tbl[j].oprd2,quad\_tbl[j].res,quad\_tbl[j].op);

}

fclose(in);

return 0;

}

void insertsymtbl(char\* name1,char\* val1)

{

struct sym\_row sr;

strcpy(sr.name,name1);

strcpy(sr.val,val1);

sym\_tbl[sym\_index]=sr;

sym\_index++;

}

void insertquadtbl(char\* opd1,char\* opd2,char\* res1,char op1)

{

struct quad\_row qr;

quad\_tbl[quad\_index]=qr;

strcpy(qr.oprd1,opd1);

strcpy(qr.oprd2,opd2);

strcpy(qr.res,res1);

qr.op = op1;

quad\_tbl[quad\_index]=qr;

quad\_index++;

}

void push(char\* str)

{

struct stack\_item si;

strcpy(si.item,str);

stk[++top]=si;

}

char\* pop()

{

char\* str=stk[top--].item;

return str;

}

========================================================================

Input.c

%{

#include "stdio.h"

#include "string.h"

extern FILE\* yyin;

struct sym\_row

{

char name[25];

char val[25];

}sym\_tbl[50];

struct quad\_row

{

char oprd1[25];

char oprd2[25];

char res[25];

char op;

}quad\_tbl[50];

struct stack\_item

{

char item[25];

}stk[50];

int sym\_index=0 , quad\_index=0 , temp\_index=0, top= -1;

void insertsymtbl(char\*,char\*);

void insertquadtbl(char\*,char\*,char\*,char);

void push(char\*);

char\* pop();

%}

%union

{

char sval[25];

}

%token MAIN

%token <sval> ID

%token <sval> DT

%token <sval> NUM

%left '+' '-'

%left '\*''/'

%start prg

%%

prg: MAIN '(' ')' '{' slist '}' {printf("Valid program.\n");}

;

slist: vstmnt s

;

vstmnt: varlist vstmnt

|

;

varlist: DT vardec ';'

;

vardec: ID ',' vardec {insertsymtbl($1,"0");}

| ID '=' NUM ',' vardec {insertsymtbl($1,$3);}

| ID '=' NUM {insertsymtbl($1,$3);}

| ID {insertsymtbl($1,"0");}

;

s: stmnt s

|

;

stmnt: ID '=' expr ';' {insertquadtbl(pop()," ",$1,'=');}

| expr ';'

;

expr: expr '+' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'+');

push(temp);

temp\_index++;

}

| expr '-' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'-');

push(temp);

temp\_index++;

}

| expr '\*' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'\*');

push(temp);

temp\_index++;

}

| expr '/' expr { char temp[4];

sprintf(temp,"t%d",temp\_index);

insertquadtbl(pop(),pop(),temp,'/');

push(temp);

temp\_index++;

}

| NUM { push($1);}

| ID { push($1);}

;

%%

int yyerror(char\* err)

{

printf("\nSyntax error.");

}

int yywrap()

{

return 1;

}

int main()

{

FILE\* in=fopen("input.c","r");

yyin=in;

yyparse();

printf("\n-----------------------------------------------------------------------------------");

printf("\nSymbol Table :");

printf("\n-----------------------------------------------------------------------------------");

printf("\nSymbol\t\tValue\n");

int i;

for(i=0;i<sym\_index;i++)

{

printf("%s\t\t%s\n",sym\_tbl[i].name,sym\_tbl[i].val);

}

printf("\n-----------------------------------------------------------------------------------");

printf("\nQuadrapule Table :");

printf("\n-----------------------------------------------------------------------------------");

printf("\nOperand 1\t\tOperand 2\t\tResult\t\tOperator\n");

int j;

for(j=0;j<quad\_index;j++)

{

printf("%s\t\t\t%s\t\t\t%s\t\t\t%c\n",quad\_tbl[j].oprd1,quad\_tbl[j].oprd2,quad\_tbl[j].res,quad\_tbl[j].op);

}

fclose(in);

return 0;

}

void insertsymtbl(char\* name1,char\* val1)

{

struct sym\_row sr;

strcpy(sr.name,name1);

strcpy(sr.val,val1);

sym\_tbl[sym\_index]=sr;

sym\_index++;

}

void insertquadtbl(char\* opd1,char\* opd2,char\* res1,char op1)

{

struct quad\_row qr;

quad\_tbl[quad\_index]=qr;

strcpy(qr.oprd1,opd1);

strcpy(qr.oprd2,opd2);

strcpy(qr.res,res1);

qr.op = op1;

quad\_tbl[quad\_index]=qr;

quad\_index++;

}

void push(char\* str)

{

struct stack\_item si;

strcpy(si.item,str);

stk[++top]=si;

}

char\* pop()

{

char\* str=stk[top--].item;

return str;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

A6:

Generator.cpp

#include <iostream>

#include <fstream>

#include <ctime>

#include <cstdlib>

using namespace std;

int main(){

ofstream f("dataset.csv");

srand(time(0));

for(int i=0;i<150;i++){

int x = rand()%1000;

f<<x;

f<<",";

int y = rand()%1000;

f<<y;

f<<"\n";

}

f.close();

}

/\*char buffer[1024];

char \*line;

FILE \*fstream = fopen("input.txt","r");

while((line=fgets(buffer,sizeof(buffer),fstream)) != NULL)

{

cout << "X : " << strtok(line,",") << endl;

cout << "Y : " << strtok(NULL,",") << endl;

// cout << "----" << endl;

}

\*/

========================================================================

A6.cpp

#include <iostream>

#include<stdlib.h>

#include<time.h>

#include<math.h>

using namespace std;

class kMeans{

//for the dataset

int noOfPoints;

float \*xCord;

float \*yCord;

//for the random cluster centres

float \*centroidX;

float \*centroidY;

//for holding the cluster numbers of points

int \*prevCluster;

int \*currentCluster;

int noOfClusters;

public:

kMeans(int noPoints,int noClusters)

{

noOfPoints=noPoints;

xCord=new float(noOfPoints);

yCord=new float(noOfPoints);

noOfClusters=noClusters;

centroidX=new float(noOfClusters);

centroidY=new float(noOfClusters);

prevCluster=new int(noOfPoints);

currentCluster=new int(noOfPoints);

}

void acceptInput();

void generateCentorids();

void calculateDistance(int);

void copyArray();

void apriori();

int checkArray();

void generateMeanCentroids();

};

void kMeans::acceptInput()

{

cout<<"Enter the points ( x and y)"<<endl;

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

{

cin>>xCord[i]>>yCord[i];//accept the points

}

}

void kMeans::generateCentorids()

{

int randIndex;

cout<<"Centroids for Pass 1\n";//generate rhe random centroids for first pass

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

{

randIndex=1+rand()%(noOfPoints-i-1);//rand() gives a random value between 1 and (noOfPoints-i-1)

centroidX[i]=xCord[randIndex];

centroidY[i]=yCord[randIndex];

cout<<i<<") ("<<centroidX[i]<<","<<centroidY[i]<<")\n";

}

}

void kMeans::generateMeanCentroids()

{

float sumX[noOfClusters],sumY[noOfClusters];//for all the points add the coordinates of the points to the cluster it belongs

int count[noOfClusters];

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

{

sumX[i]=0;

sumY[i]=0;

}

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

{

count[i]=0;

}

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

{

for(int j=0;j<noOfPoints;j++)

{

if(currentCluster[j]==i)//if the point belongs to the cluster

{

count[i]=count[i]+1;

sumX[i]=sumX[i]+xCord[j];

sumY[i]=sumY[i]+yCord[j];

}

}

}

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

{

centroidX[i]=sumX[i]/count[i];

centroidY[i]=sumY[i]/count[i];

cout<<i<<") ("<<centroidX[i]<<","<<centroidY[i]<<")\n";

}

}

void kMeans::calculateDistance(int point)

{

float sum[noOfClusters];

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

{

sum[i]=0;

}

float x1,x2;

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

{

sum[i]=(fabs(centroidX[i]-xCord[point]))+(fabs(centroidY[i]-yCord[point]));//find distance from all the clusters

}

float min=sum[0];

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

{

if(sum[i]<min)

{

min=sum[i];

}

}

int index;

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

{

if(sum[i]==min)

{

index=i;//find the index of the cluster

}

}

currentCluster[point]=index;

}

void kMeans::copyArray()//copy the array

{

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

{

prevCluster[i]=currentCluster[i];

}

}

int kMeans::checkArray()

{

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

{

if(prevCluster[i]!=currentCluster[i])//check if the clusters have changed,if they havent the algorithm stops

return -1;

}

return 0;

}

void kMeans::apriori()//the main function

{

int flag=0;

int count=1;

while(flag!=1)

{

if(count==1){

generateCentorids();}//initially generate random centroids

else

{

cout<<"Centroids for Pass "<<count<<endl;

generateMeanCentroids();//for every other pass generate mean centroids

}

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

{

calculateDistance(i);//calculate the distance and determine the cluster to which the point belongs

}

if(checkArray()==0){//check if to stop

flag=1;}

copyArray();//if not copy the array for a new pass

count=count+1;

cout<<"==================================\n";

cout<<"POINT \t\t CLUSTER NUMBER\n";

cout<<"==================================\n";

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

{

cout<<"Point "<<i<<" ("<<xCord[i]<<","<<yCord[i]<<")\t\t"<<currentCluster[i]<<endl;

}

}

}

int main() {

int noPoints,noClusters;

cout<<"Enter number of points:\t";

cin>>noPoints;

cout<<"Enter number of clusters: ";

cin>>noClusters;

kMeans kmeans(noPoints,noClusters);//create an object of class kMeans

kmeans.acceptInput();//accept the input

kmeans.apriori();

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B2:

#include <iostream>

#include <cstring>

#include <cstdio>

#include <climits>

#include <cmath>

#include <omp.h>

using namespace std;

int adj[20][20],cost[20][20],n;

int dp[20][1<<20],npow;

int TSP(int start,int subset\_mask)

{

int temp,result = INT\_MAX;

if(dp[start][subset\_mask]!=-1)

return dp[start][subset\_mask];

cout<<start<<"-->"

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

{

int mask = (npow - 1) - (1<<i);

int masked = subset\_mask & mask;

if(masked!=subset\_mask){

temp = adj[start][i] + TSP(i,masked);

if(temp < result)

result = temp;

}

}

return dp[start][subset\_mask] = result;

}

int main(){

cout<<"Enter the number of cities:\t";

cin>>n;

cout<<"Enter the adjacency matrix:\n";

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

for(int j = 0; j < n;j++){

cin>>adj[i][j];

cost[i][j] = adj[i][j];

}

npow = (int)pow(2,n);

#pragma omp for

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

memset(dp , -1 ,sizeof(dp));

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

dp[i][0] = adj[i][0];

cout<<"Starting from source "<<i<<" minimum path distance is "<<TSP(i,npow-2)<<endl;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B5:

#include <iostream>

#include <string.h>

#include <sstream>

using namespace std;

class stack{

char data[20][3];

int top;

public:

stack(){

top=-1;

}

void push(char a[3]){

strcpy(data[++top],a);

}

char\* pop(){

return data[top--];

}

char\* return\_top(){

return data[top];

}

bool isEmpty(){

if(top==-1){return true;}

else {return false;}

}

void print\_stack(){

for(int i=top;i>=0;i--){

cout<<data[i]<<" , ";

}

}

};

stack reg\_stack;

stack temp\_stack;

class node{

public:

char data;

node\* left;

node\* right;

bool leftchild;

int label;

node(){

data=' ';

left=NULL;

right=NULL;

leftchild=true;

label=0;

}

node(char d,bool l){

data=d;

left=NULL;

right=NULL;

leftchild=l;

label=0;

}

};

void accept\_dag(node\* t){

//first the left subtree is created then the right subtree

cout<<" Enter data for node: ";

cin>>t->data;

//if it is an operator accept the operands

if(t->data=='+'||t->data=='-'||t->data=='\*'||t->data=='/'){

cout<<"\nEnter left child of "<<t->data;

//create a new left child node

node\* l=new node(' ',true);

t->left=l;

accept\_dag(l);

//create the right child of that node

cout<<"\nEnter right child of "<<t->data;

node\* l1=new node(' ',false);

t->right=l1;

accept\_dag(l1);

}

}

void labelling(node\* t){

//labelling is done to ensure partial ordering;each label corresponds to an element in the set

//labels basically show the path in the DAG. if l1>l2:there is a path from l1->l2. if l1 = l2 then parent has label l3 = l1+1 as it has to be unique,if l3=l1,then there would be a cycle l1->l2->l3

if(t!=null){

labelling(t->left); //left subtree is labelled first

labelling(t->right);//followed by the right subtree

if(t->left==NULL&&t->right==NULL){ //for leaf

if(t->leftchild==true){t->label=1;} //left leaf is labelled as 1

else {t->label=0;} //right leaf is labelled as 0

}

else{

int l1=t->left->label;//find the label of the left child

int l2=t->right->label;//find the label if the right child

if(l1==l2){

t->label=l1+1;//if the labels are equal,parent label increments by 1

}

else{

t->label=(l1>l2?l1:l2);//if not equal then the greater of the child labels is taken by the parent

}

}

}

}

void inorder(node\* t){

//inorder is left->root->right

if(t!=NULL){

inorder(t->left);

cout<<t->data<<" :"<<t->label<<" , ";

inorder(t->right);

}

}

void initialize\_stacks(){

int k;

cout<<"\n Enter Size of reg\_stack: ";

cin>>k;

//this function basically names the regular and the temporary registers by using the stringstream class to handle the integer value as string

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

//3 because a[0] is R/t[0] is t,a[1]/t[1] will contain the integer number and a[2]/t[2] is going to be the null termination character as c\_str()returns a pointer with null-terminated character.

char a[3]="R";

char t[3]="t";

int l=k-i-1;

//create a stringstream object strs

stringstream strs;strs<<l;

//concatenates a,str()returns the contents of the buffer in string type,c\_str() returns a pointer to an array that contains the null-terminated value of the string

strcat(a,strs.str().c\_str());

//push the concatenated value to the regular stack

reg\_stack.push(a);

//push the string value to the temporary stack.

temp\_stack.push(strcat(t,strs.str().c\_str()));

}

cout<<"\nRegister Stack :";

reg\_stack.print\_stack();

cout<<"\nTemp Variables Stack :";

temp\_stack.print\_stack();

}

void swap(stack s){

char\* a=s.pop();

char\* b=s.pop();

s.push(b);

s.push(a);

}

char\* op(char o){

if(o=='+') {return "ADD";}

else if(o=='-') {return "SUB";}

else if(o=='\*') {return "MUL";}

else {return "DIV";}

}

void Gen\_code(node\* n){

if(n->left==NULL&&n->right==NULL&&n->leftchild==true){ //for left child

cout<<"MOV "<<n->data<<" , "<<reg\_stack.return\_top()<<"\n";

}

else{

node\* n1=n->left;

node\* n2=n->right;

if(n2->label==0){//right child present then start with the left child as if there is an operator first operand is going to be the left child,and we have to check if it is an operator by checking for the right child

Gen\_code(n1);

cout<<op(n->data)<<" "<<n2->data<<" , "<<reg\_stack.return\_top()<<"\n";

}

else if(n1->label<n2->label){//left child label is less than right child label

swap(reg\_stack);//swap the top element with the second top element

Gen\_code(n2);

char\* r=reg\_stack.pop();//pop the right child so that left child can be evaluated

Gen\_code(n1);

cout<<op(n->data)<<" "<<r<<" , "<<reg\_stack.return\_top()<<"\n";

reg\_stack.push(r);//push the popped right child back to stack

swap(reg\_stack);//swap them again to get the original stack

}

else if(n2->label<n1->label){//first left ,then right child

Gen\_code(n1);

char\* r=reg\_stack.pop();

Gen\_code(n2);

cout<<op(n->data)<<" "<<r<<" , "<<reg\_stack.return\_top()<<"\n";

reg\_stack.push(r);

}

/\*else{

Gen\_code(n2);//for intermediate nodes:left child >1 and right child label !=0

char\* t=temp\_stack.pop();

cout<<"MOV "<<reg\_stack.return\_top()<<" , "<<t<<"\n";//second topmost element

Gen\_code(n1);

temp\_stack.push(t);//get original stack

cout<<op(n->data)<<" "<<t<<" , "<<reg\_stack.return\_top()<<"\n";

}\*/

}

}

int main() {

node\* root=NULL;

root=new node();

cout<<"\nStarting with the root node enter data: \n";

accept\_dag(root);

labelling(root);

cout<<"\nEntered Tree with labels is: \n";

inorder(root);

initialize\_stacks();

cout<<"\n\nGenerated Code is:\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

Gen\_code(root);

cout<<"\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

cout<<"\nFinal result is stored in register: "<<reg\_stack.return\_top();

return 0;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B8:

import xml.etree.ElementTree as ET

# XML is just a tag heirarchy.

tree = ET.parse('parsing\_table.xml')

root = tree.getroot()

# all available terminals and non-terminals in the grammar.

term = []

non\_term = []

# whatever is on the LHS and RHS of the production.

lsprod = []

rsprod = []

# no of states

n = 0

# traverse every node and populate it's contents into program variables. Can't reach into the XML file everytime

# we need something

for child in root:

if(child.tag == "states"):

n = int(child.text)

elif(child.tag == "term"):

term.append(child.text)

elif(child.tag == "nterm"):

non\_term.append(child.text)

elif(child.tag == "productions"):

# iterate over every production

for ch in child:

# as per the XML file, first tag inside the production is <l>

lsprod.append(ch[0].text)

# next is <r>

rsprod.append(ch[1].text)

elif(child.tag == "actiontable"):

# create n empty lists inside another list

action=[[] for x in range(n)]

i=0

# for every row of data

for ch in child:

# for every data inside the row

for c in ch:

# add it to the appropriate list

action[i].append(c.text)

i=i+1

elif(child.tag=="gototable"):

goto=[[] for x in range(n)]

i=0

for ch in child:

for c in ch:

goto[i].append(c.text)

i=i+1

nterm=len(term)

nnterm=len(non\_term)

nprod=len(lsprod)

print("Terminals: "),;print(term)

print("Non Terminals: "),;print(non\_term)

print("Grammar Productions are as follows: ")

for i in range(nprod):

print(lsprod[i]+" -> "+rsprod[i])

print("\nAction Table: ")

for i in range(n):

print("")

for j in range(nterm):

print(action[i][j]+" "),

print("")

print("Goto Table: ")

for i in range(n):

print(" ")

for j in range(nnterm):

print(str(goto[i][j])+" "),

while True:

print("\nEnter input String: "),

istr=raw\_input()

iptr=0

# the actual logic. till now everything was unnecessarily tedious

# stack[0] is seen as bottom

stack=['$',0]

done = False

while True:

print("Stack :")

print(stack)

# last element in list is stack top

stack\_top = stack[len(stack)-1]

# pointer has consumed all input

if iptr >= len(istr):

done = True

else:

# symbol loaded from input into isym

isym = istr[iptr]

# index of symbol in the terminal list

input\_sym\_index = term.index(isym)

# get the action from the action table

ac = action[stack\_top][input\_sym\_index]

print("Action for stack\_top="+str(stack\_top)+" and input symbol index "+str(input\_sym\_index)+" is "+ac)

if(ac == "Error"):

print("Syntax Error!!!")

break

elif(done or ac == "Accept"):

print("Correct Syntax!!")

break

#shifting onto the stack,append the symbol read and the state number

elif("s" in ac):

#append the read symbol onto the stack

stack.append(isym)

ns=ac.replace("s","")

#append the state number onto the stack

stack.append(int(ns))

#increment the pointer

iptr=iptr+1

#replace the element

elif("r" in ac):

#get the state number from ac

rrule=int(ac.replace("r",""))

print(rrule)

#print the production rule that can be used for reduction

print("Reduce using rule "+lsprod[rrule-1]+" -> "+rsprod[rrule-1])

#this is used for removing the reduced symbols from the stack. '2' signifies the symbol as well as the state associated with it

for i in range(2\*len(rsprod[rrule-1])):

stack.pop()

print(stack)

#append the non-terminal to the stack

stack.append(lsprod[rrule-1])

#find the topmost state of the stack

pstate=stack[len(stack)-2]

print pstate

#find the index of the non-terminal which replaced the read symbol

ntindex=non\_term.index(lsprod[rrule-1])

print(ntindex)

#find the goto state

nst=goto[pstate][ntindex]

#append the goto state to the stack\_top

stack.append(int(nst))

print(stack)

========================================================================

Parsing.xml

<parsetable>

<states>12</states>

<term>i</term>

<term>+</term>

<term>\*</term>

<term>(</term>

<term>)</term>

<term>$</term>

<nterm>E</nterm>

<nterm>T</nterm>

<nterm>F</nterm>

<productions>

<prod><l>E</l><r>E+T</r></prod>

<prod><l>E</l><r>T</r></prod>

<prod><l>T</l><r>T\*F</r></prod>

<prod><l>T</l><r>F</r></prod>

<prod><l>F</l><r>(E)</r></prod>

<prod><l>F</l><r>i</r></prod>

</productions>

<actiontable>

<tr><td>s5</td><td>Error</td><td>Error</td><td>s4</td><td>Error</td><td>Error</td></tr>

<tr><td>Error</td><td>s6</td><td>Error</td><td>Error</td><td>Error</td><td>Accept</td></tr>

<tr><td>r2</td><td>r2</td><td>s7</td><td>r2</td><td>r2</td><td>r2</td></tr>

<tr><td>r4</td><td>r4</td><td>r4</td><td>r4</td><td>r4</td><td>r4</td></tr>

<tr><td>s5</td><td>Error</td><td>Error</td><td>s4</td><td>Error</td><td>Error</td></tr>

<tr><td>r6</td><td>r6</td><td>r6</td><td>r6</td><td>r6</td><td>r6</td></tr>

<tr><td>s5</td><td>Error</td><td>Error</td><td>s4</td><td>Error</td><td>Error</td></tr>

<tr><td>s5</td><td>Error</td><td>Error</td><td>s4</td><td>Error</td><td>Error</td></tr>

<tr><td>Error</td><td>s6</td><td>Error</td><td>Error</td><td>s11</td><td>Error</td></tr>

<tr><td>r1</td><td>r1</td><td>s7</td><td>r1</td><td>r1</td><td>r1</td></tr>

<tr><td>r3</td><td>r3</td><td>r3</td><td>r3</td><td>r3</td><td>r3</td></tr>

<tr><td>r5</td><td>r5</td><td>r5</td><td>r5</td><td>r5</td><td>r5</td></tr>

</actiontable>

<gototable>

<tr><td>1</td><td>2</td><td>3</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>8</td><td>2</td><td>3</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>9</td><td>3</td></tr>

<tr><td>0</td><td>0</td><td>10</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

<tr><td>0</td><td>0</td><td>0</td></tr>

</gototable>

</parsetable>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*B9:

from math import log

from math import sqrt

from collections import Counter

from operator import itemgetter

#function to calculate inverse document frequency of terms

def idf(term, allDocuments):

numDocumentsWithThisTerm = 0

for cnt in allDocuments:

if term in cnt:

numDocumentsWithThisTerm = numDocumentsWithThisTerm + 1

if numDocumentsWithThisTerm > 0:

return round(log(float(float(len(allDocuments))/float(numDocumentsWithThisTerm)),2),3)

else:

return 0

#function to calculate frequency of individual terms

def tf(term, document):

return document.count(term)

#function to calculate tfidf

def caltfidf(term,doc):

return tf(term,doc)\*idf(term,terms)

#cosine similarity calculated using tfidf

def cosineSimilarity(doc,q,d):

a=0

for x in d:

a=a+caltfidf(x,doc)\*caltfidf(x,q)

b=lengthof(doc,d)\*lengthof(q,d)

if not b:

return 0

else:

return round(a/b,3)

def lengthof(doc,d):

val=0

for x in d:

val=val+pow(caltfidf(x,doc),2)

return sqrt(val)

#doc is all training files

#dataset has files with their 'class'

#contents of files are stored in f

f=[]

doc=['doc1.txt','doc2.txt','doc3.txt','doc4.txt','doc5.txt','doc6.txt']

dataset=[['doc1.txt','science'],['doc2.txt','science'],['doc3.txt','science'],['doc4.txt','entertainment'],['doc5.txt','entertainment'],['doc6.txt','entertainment']]

for x in doc:

f.append(open(x,'r').read())#put all the data into f

#q is file to be classified

testf=raw\_input('test file : ')

q=open(testf,'r').readline().lower()#lower() converts everything into lowercase

#terms has cleaned content of files

terms=[]

for x in f:

terms.append(x.lower().rstrip('\n'))

#fin\_terms is list of unique words

fin\_terms=[]

for x in terms:

fin\_terms=fin\_terms+x.split()

fin\_terms=set(fin\_terms)

fin\_terms=list(fin\_terms)

#print fin\_terms

print '\nCosine Similarity Values'

#cosine similarity of test file calculated wrt all training files

cnt=0

for x in terms:

dataset[cnt]=dataset[cnt]+[cosineSimilarity(q,x,fin\_terms)]

cnt=cnt+1

#knn classsification used with cosine similarity value to find 3 closest documents

k=3

#dataset is sorted on basis of cosine similarity values

sorted\_dataset=sorted(dataset,key=itemgetter(2),reverse=True)

top\_k=sorted\_dataset[:k]

top\_k[:] = (x for x in top\_k if x[2] != 0)

#show error msg if all similarity values are 0

if len(top\_k)== 0:

print 'Does not match'

else:

class\_counts=Counter(category for (document,category,value) in top\_k)

print class\_counts

#match class to the class which is max in top k

classification=max(class\_counts,key=lambda cls:class\_counts[cls])

print 'Class of test file is : ',classification

========================================================================doc1.txt:

Science is a systematic enterprise that builds and organizes knowledge in the form of testable explanations and predictions about the universe. In an older and closely related meaning, science also refers to this body of knowledge itself, of the type that can be rationally explained and reliably applied. Ever since classical antiquity, science as a type of knowledge has been closely linked to philosophy. In the West during the early modern period the term Science and Natural Philosophy were sometimes used interchangeably to refer to the study of natural phenomena and, until the 19th century, natural philosophy was considered a branch of philosophy.

========================================================================

Doc2.txt:

In modern usage science most often refers to a way of pursuing knowledge, not only the knowledge itself. In the 17th and 18th centuries scientists increasingly sought to formulate knowledge in terms of laws of nature. Over the course of the 19th century, the word "science" became increasingly associated with the scientific method itself, as a disciplined way to study the natural world, including physics, chemistry, geology and biology. It is in the 19th century also that the term scientist began to be applied to those who sought knowledge and understanding of nature.

========================================================================doc3.txt:

Modern science is typically subdivided into the natural sciences which study the material world, the social sciences which study people and societies, and the formal sciences like mathematics. The formal sciences are often excluded as they do not depend on empirical observations. Disciplines which use science like engineering and medicine may also be considered to be applied sciences.

========================================================================

Doc4.txt:

The experience of being entertained has come to be strongly associated with amusement, so that one common understanding of the idea is fun and laughter, although many entertainments have a serious purpose. This may be the case in the various forms of ceremony, celebration, religious festival, or satire for example. Hence, there is the possibility that what appears as entertainment may also be a means of achieving insight or intellectual growth.

========================================================================

Doc5.txt:

An important aspect of entertainment is the audience, which turns a private recreation or leisure activity into entertainment. The audience may have a passive role, as in the case of persons watching a play, opera, television show, or film; or the audience role may be active, as in the case of games, where the participant/audience roles may be routinely reversed. Entertainment can be public or private, involving formal, scripted performance, as in the case of theatre or concerts; or unscripted and spontaneous, as in the case of children's games. Most forms of entertainment have persisted over many centuries, evolving due to changes in culture, technology, and fashion. Films and video games, for example, although they use newer media, continue to tell stories, present drama, and play music. Festivals devoted to music, film, or dance allow audiences to be entertained over a number of consecutive days.

========================================================================

Doc6.txt:

Some activities that once were considered entertaining, particularly public punishments, have been removed from the public arena. Others, such as fencing or archery, once necessary skills for some, have become serious sports and even professions for the participants, at the same time developing into entertainment with wider appeal for bigger audiences. In the same way, other necessary skills, such as cooking, have developed into performances among professionals, staged as global competitions and then broadcast for entertainment. What is entertainment for one group or individual may be regarded as work by another.

=======================================================================

T.txt:

The action of providing or being provided with amusement or enjoyment.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B11:

import java.io.\*;

import java.util.\*;

import weka.classifiers.bayes.NaiveBayes;

import weka.classifiers.trees.J48;

import weka.classifiers.Evaluation;

import weka.core.Instances;

public class test

{

public static void main(String args[])

{

BufferedReader r;

try

{

r = new BufferedReader(new FileReader("src/test.arff"));

Instances train = new Instances(r);

train.setClassIndex(train.numAttributes()-1); // set attribute which is to be predicted //

NaiveBayes nb = new NaiveBayes();

nb.buildClassifier(train);

// build on training data //

J48 tree = new J48();

tree.buildClassifier(train);

Evaluation eval1 = new Evaluation(train);

eval1.crossValidateModel(nb,train,10,new Random(1));

Evaluation eval2 = new Evaluation(train);

eval2.crossValidateModel(tree,train,10,new Random(1));

System.out.println("=====NB======");

System.out.println(eval1.toSummaryString("Results",true));

System.out.println("=======DT=====");

System.out.println(eval2.toSummaryString("Results",true));

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

B13:

import csv

import random

import math

import operator

def loadDataset(filename, split, trainingSet=[] , testSet=[]):

with open(filename, 'rb') as csvfile:

lines = csv.reader(csvfile) #reads csv file

dataset = list(lines)# returns a list of the lines

for x in range(len(dataset)-1):#iterate over the dataset,and put each data list either into training set or test set

for y in range(4):

dataset[x][y] = float(dataset[x][y])

if random.random() < split:

trainingSet.append(dataset[x])

else:

testSet.append(dataset[x])

def euclideanDistance(instance1, instance2, length):

distance = 0

for x in range(length):#for all the attributes

distance += pow((instance1[x] - instance2[x]), 2)# find the sum of squares of all attributes

return math.sqrt(distance)

def getNeighbors(trainingSet, testInstance, k):

distances = []

length = len(testInstance)-1 #ie 3,basically we remove the label and keep the attributes

for x in range(len(trainingSet)):

dist = euclideanDistance(testInstance, trainingSet[x], length)

distances.append((trainingSet[x], dist))

distances.sort(key=operator.itemgetter(1))#sort according to the distance

neighbors = []

for x in range(k):#find the nearest 3 neighbours

neighbors.append(distances[x][0])#return the labels of the nearest

return neighbors

def getResponse(neighbors):

classVotes = {}

for x in range(len(neighbors)):

response = neighbors[x][-1]#response is the last value in the item

if response in classVotes:#if it is present increment its count

classVotes[response] += 1

else:

classVotes[response] = 1 #else add it to the response

sortedVotes = sorted(classVotes.iteritems(), key=operator.itemgetter(1), reverse=True)#classvotes contain the label and the count

return sortedVotes[0][0]

def getAccuracy(testSet, predictions):

correct = 0

for x in range(len(testSet)):

if testSet[x][-1] == predictions[x]:

correct += 1

return (correct/float(len(testSet))) \* 100.0

def main():

# prepare data

trainingSet=[]

testSet=[]

split = 0.67

loadDataset('iris.data', split, trainingSet, testSet)

print 'Train set: ' + repr(len(trainingSet))#repr returns a strin containing the printable version of an object

print 'Test set: ' + repr(len(testSet))

# generate predictions

predictions=[]

k = 3

for x in range(len(testSet)):

neighbors = getNeighbors(trainingSet, testSet[x], k)

result = getResponse(neighbors)

predictions.append(result)

print('> predicted=' + repr(result) + ', actual=' + repr(testSet[x][-1]))

accuracy = getAccuracy(testSet, predictions)

print('Accuracy: ' + repr(accuracy) + '%')

main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*