***ASSIGNMENT-8***

1. ***WAP to implement Sparse Matrix using  
    a. Triplet representation***.

#include<stdio.h>

#include<stdlib.h>

void create(int arr[][100],int trp[][3],int row,int col);

void display(int trp[][3]);

int main()

{

int arr[1000][100];

int trp[1000][3]={0};

int x,y,ch;

int m,n;

printf("Enter rows and columns in sparse matrix\n");

scanf("%d%d",&m,&n);

for(x=0;x<m;x++)

for(y=0;y<n;y++)

scanf("%d",&arr[x][y]);

create(arr,trp,m,n);

printf("Triplet Representation of Sparse Matrix:\n");

display(trp);

return 0;

}

void create(int arr[][100],int trp[][3],int row,int col)

{

int i,j;

trp[0][0] = row;

trp[0][1] = col;

int x=1;

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

{

for(j=0;j<col;j++)

{

if(arr[i][j]!=0)

{

trp[x][0] = i;

trp[x][1] = j;

trp[x][2] = arr[i][j];

trp[0][2]++;

x++;

}

}

}

}

void display(int trp[][3])

{

int cnt=0;

while(cnt<=trp[0][2])

{

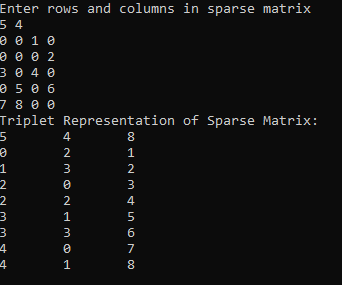
printf("%d\t%d\t%d\n",trp[cnt][0],trp[cnt][1],trp[cnt][2]);

cnt++;

}

}

***OUTPUT :***



***b. Multi-linked representation***

#include<stdio.h>

#include<stdlib.h>

struct mlnode{

int row,col,val;

struct mlnode \*nextrow,\*nextcol;

};

typedef struct mlnode MLNODE;

MLNODE \*above(MLNODE \*head,int r,int c);

MLNODE \*left(MLNODE \*head,int r,int c);

MLNODE \*insert(MLNODE \*head,int r,int c,int val);

MLNODE \*delete(MLNODE \*head,int r,int c);

void display(MLNODE \*head);

int main()

{

MLNODE \*head = (MLNODE \*)malloc(sizeof(MLNODE));

head->col = head->row = -1;

head->nextcol = head->nextrow = head;

int n,m;

printf("Enter number of rows and column : ");

scanf("%d%d",&n,&m);

int i;

MLNODE \*p=head;

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

{

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->col = -1;

tmp->row = i;

p->nextrow = tmp;

tmp->nextrow = head;

tmp->nextcol = tmp;

p = p->nextrow;

}

p=head;

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

{

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->col = i;

tmp->row = -1;

p->nextcol = tmp;

tmp->nextcol = head;

tmp->nextrow = tmp;

p = p->nextcol;

}

int arr[100][100];

int j,x;

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

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

{

scanf("%d",&x);

if(x!=0)

head = insert(head,i,j,x);

}

int ch,val,r,c;

printf("Enter your choice what you want to do\n");

printf("1.insert an element\n");

printf("2.Delete an element\n");

printf("3.Display the matrix\n");

printf("4.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter row,column and value : ");

scanf("%d%d%d",&r,&c,&val);

head = insert(head,r,c,val);

break;

case 2:

printf("Enter row and column : ");

scanf("%d%d",&r,&c);

head = delete(head,r,c);

break;

case 3:

display(head);

break;

case 4:

exit(1);

}

}

return 0;

}

MLNODE \*above(MLNODE \*head,int r,int c)

{

MLNODE \*p=head,\*q;

while(p->col!=c)

p = p->nextcol;

do{

q=p;

p=p->nextrow;

}while(p->row < r && p->row!=-1);

return q;

}

MLNODE \*left(MLNODE \*head,int r,int c)

{

MLNODE \*p=head,\*q;

while(p->row!=r)

p = p->nextrow;

do{

q=p;

p = p->nextcol;

}while(p->col < c && p->col!=-1);

return q;

}

MLNODE \*insert(MLNODE \*head,int r,int c,int val)

{

MLNODE \*p = above(head,r,c);

MLNODE \*q = left(head,r,c);

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->val = val;

tmp->col = c;

tmp->row = r;

tmp->nextcol = q->nextcol;

tmp->nextrow = p->nextrow;

q->nextcol = p->nextrow = tmp;

return head;

}

MLNODE \*delete(MLNODE \*head,int r,int c)

{

MLNODE \*p = above(head,r,c);

MLNODE \*q = left(head,r,c);

MLNODE \*x = p->nextrow;

p->nextrow = x->nextrow;

q->nextcol = x->nextcol;

free(x);

return head;

}

void display(MLNODE \*head)

{

MLNODE \*p=head->nextrow,\*q;

while(p->row!=-1)

{

q=p->nextcol;

while(q->col!=-1)

{

printf("%d ",q->val);

q=q->nextcol;

}

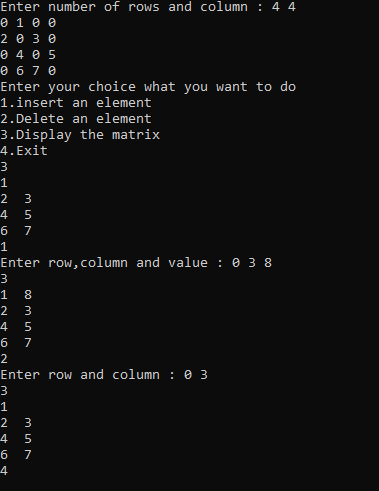
printf("\n");

p = p->nextrow;

}

}

***OUTPUT :***



1. ***WAP to implement the following for both the representations.  
    a. Delete the i-th row from the Sparse Matrix***

***b. Swap the i-th row and j-th row of the Sparse Matrix***

***(i)For MultiLinked List :***

#include<stdio.h>

#include<stdlib.h>

struct mlnode{

int row,col,val;

struct mlnode \*nextrow,\*nextcol;

};

typedef struct mlnode MLNODE;

MLNODE \*above(MLNODE \*head,int r,int c);

MLNODE \*left(MLNODE \*head,int r,int c);

MLNODE \*insert(MLNODE \*head,int r,int c,int val);

MLNODE \*swaprow(MLNODE \*head,int r1,int r2);

MLNODE \*deleterow(MLNODE \*head,int r);

void display(MLNODE \*head);

int main()

{

MLNODE \*head = (MLNODE \*)malloc(sizeof(MLNODE));

head->col = head->row = -1;

head->nextcol = head->nextrow = head;

int n,m;

printf("Enter number of rows and column : ");

scanf("%d%d",&n,&m);

int i;

MLNODE \*p=head;

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

{

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->col = -1;

tmp->row = i;

p->nextrow = tmp;

tmp->nextrow = head;

tmp->nextcol = tmp;

p = p->nextrow;

}

p=head;

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

{

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->col = i;

tmp->row = -1;

p->nextcol = tmp;

tmp->nextcol = head;

tmp->nextrow = tmp;

p = p->nextcol;

}

int arr[100][100];

int j,x;

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

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

{

scanf("%d",&x);

if(x!=0)

head = insert(head,i,j,x);

}

int ch,val,r,c;

printf("Enter your choice what you want to do\n");

printf("1.insert an element\n");

printf("2.Swap two rows of matrix\n");

printf("3.Display the matrix\n");

printf("4.Delete a row from matrix\n");

printf("5.Exit\n");

while(1){

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter row,column and value : ");

scanf("%d%d%d",&r,&c,&val);

head = insert(head,r,c,val);

break;

case 2:

printf("Enter row1 and row2 : ");

scanf("%d%d",&r,&c);

head = swaprow(head,r,c);

break;

case 3:

display(head);

break;

case 4:

printf("Enter row number : ");

scanf("%d",&r);

head = deleterow(head,r);

break;

case 5:

exit(1);

}

}

return 0;

}

MLNODE \*above(MLNODE \*head,int r,int c)

{

MLNODE \*p=head,\*q;

while(p->col!=c)

p = p->nextcol;

do{

q=p;

p=p->nextrow;

}while(p->row < r && p->row!=-1);

return q;

}

MLNODE \*left(MLNODE \*head,int r,int c)

{

MLNODE \*p=head,\*q;

while(p->row!=r)

p = p->nextrow;

do{

q=p;

p = p->nextcol;

}while(p->col < c && p->col!=-1);

return q;

}

MLNODE \*insert(MLNODE \*head,int r,int c,int val)

{

MLNODE \*p = above(head,r,c);

MLNODE \*q = left(head,r,c);

MLNODE \*tmp = (MLNODE \*)malloc(sizeof(MLNODE));

tmp->val = val;

tmp->col = c;

tmp->row = r;

tmp->nextcol = q->nextcol;

tmp->nextrow = p->nextrow;

q->nextcol = p->nextrow = tmp;

return head;

}

MLNODE \*delete(MLNODE \*head,int r,int c)

{

MLNODE \*p = above(head,r,c);

MLNODE \*q = left(head,r,c);

MLNODE \*x = p->nextrow;

p->nextrow = x->nextrow;

q->nextcol = x->nextcol;

free(x);

return head;

}

MLNODE \*swaprow(MLNODE \*head,int r1,int r2)

{

MLNODE \*p=head,\*q=head;

while(p->row!=r1)

p = p->nextrow;

while(q->row!=r2)

q = q->nextrow;

p = p->nextcol;

q = q->nextcol;

while(p->col!=-1 && q->col!=-1)

{

if(p->col < q->col)

{

head = insert(head,r2,p->col,p->val);

int x = p->col;

p = p->nextcol;

head = delete(head,r1,x);

}

else if(p->col > q->col)

{

head = insert(head,r1,q->col,q->val);

int x = q->col;

q = q->nextcol;

head = delete(head,r2,x);

}

else

{

int tmp;

tmp = p->val;

p->val = q->val;

q->val = tmp;

p = p->nextcol;

q = q->nextcol;

}

}

if(p->col!=-1)

{

while(p->col!=-1)

{

head = insert(head,r2,p->col,p->val);

int x = p->col;

p = p->nextcol;

head = delete(head,r1,x);

}

}

if(q->col!=-1)

{

while(q->col!=-1)

{

head = insert(head,r1,q->col,q->val);

int x = q->col;

q = q->nextcol;

head = delete(head,r2,x);

}

}

return head;

}

MLNODE \*deleterow(MLNODE \*head,int r)

{

MLNODE \*p=head;

while(p->row!=r)

p = p->nextrow;

MLNODE \*q=p->nextrow,\*x=p,\*s;

do

{

s = above(head,r,p->col);

s->nextrow = p->nextrow;

s = p;

p = p->nextcol;

free(s);

}while(p!=x);

while(q!=head)

{

x = q;

do

{

(x->row)--;

x = x->nextcol;

}while(x!=q);

q = q->nextrow;

}

return head;

}

void display(MLNODE \*head)

{

MLNODE \*p=head->nextrow,\*q;

while(p->row!=-1)

{

q=p->nextcol;

while(q->col!=-1)

{

printf("%d ",q->val);

q=q->nextcol;

}

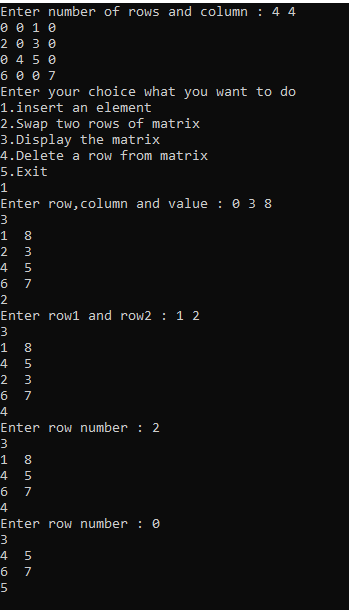
printf("\n");

p = p->nextrow;

}

}

***OUTPUT :***



***(ii) For Triplet Respresentation***

#include<stdio.h>

#include<stdlib.h>

void create(int arr[][100],int trp[][3],int row,int col);

void deleterow(int trp[][3],int row);

void swap(int trp[][3],int row1,int row2);

void display(int trp[][3]);

int main()

{

int arr[100][100];

int trp[1000][3]={0};

int x,y,ch;

int m,n;

printf("Enter rows and columns in sparse matrix : ");

scanf("%d%d",&m,&n);

for(x=0;x<m;x++)

for(y=0;y<n;y++)

scanf("%d",&arr[x][y]);

create(arr,trp,m,n);

printf("1.Delete row of Sparse Matrix\n");

printf("2.Swap two rows of Sparse Matrix\n");

printf("3.Display Triplet Representation of Sparse Matrix\n");

while(1)

{

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter row number which you want to delete: ");

scanf("%d",&x);

deleterow(trp,x);

break;

case 2:

printf("Enter row1 and row2 : ");

scanf("%d%d",&x,&y);

swap(trp,x,y);

break;

case 3:

display(trp);

break;

case 4:

exit(1);

}

}

return 0;

}

void create(int arr[][100],int trp[][3],int row,int col)

{

int i,j;

trp[0][0] = row;

trp[0][1] = col;

int x=1;

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

{

for(j=0;j<col;j++)

{

if(arr[i][j]!=0)

{

trp[x][0] = i;

trp[x][1] = j;

trp[x][2] = arr[i][j];

trp[0][2]++;

x++;

}

}

}

}

void deleterow(int trp[][3],int row)

{

int i,j,k;

int cnt=0;

for(i=1;i<=trp[0][2];i++)

{

if(trp[i][0]==row-1)

{

if(!cnt)

k = i;

cnt++;

}

else if(trp[i][0] > row-1)

break;

}

i = k + cnt;

while(i<=trp[0][2])

{

trp[k][0] = --trp[i][0];

trp[k][1] = trp[i][1];

trp[k][2] = trp[i][2];

k++;

i++;

}

trp[0][2] = trp[0][2] - cnt;

trp[0][0]--;

}

void swap(int trp[][3],int row1,int row2)

{

int i,j;

for(i=1;i<=trp[0][2];i++)

{

if(trp[i][0]==row1-1)

trp[i][0] = row2-1;

else if(trp[i][0]==row2-1)

trp[i][0] = row1-1;

}

int temp1,temp2,temp3;

for(i=1;i<trp[0][2];i++)

{

for(j=i+1;j<=trp[0][2];j++)

{

if(trp[i][0] > trp[j][0])

{

temp1 = trp[i][0];

temp2 = trp[i][1];

temp3 = trp[i][2];

trp[i][0] = trp[j][0];

trp[i][1] = trp[j][1];

trp[i][2] = trp[j][2];

trp[j][0] = temp1;

trp[j][1] = temp2;

trp[j][2] = temp3;

}

else if(trp[i][0] == trp[j][0])

{

if(trp[i][1] > trp[j][1])

{

temp1 = trp[i][1];

temp2 = trp[i][2];

trp[i][1] = trp[j][1];

trp[i][2] = trp[j][2];

trp[j][1] = temp1;

trp[j][2] = temp2;

}

}

}

}

}

void display(int trp[][3])

{

int cnt=0;

while(cnt<=trp[0][2])

{

printf("%d\t%d\t%d\n",trp[cnt][0],trp[cnt][1],trp[cnt][2]);

cnt++;

}

}

***OUTPUT :***

