1602-19-733-119

V.Venkateswarlu Gupta

CSE-B, 3rd SEM.

**Data Structures Lab**

***Prelab Questions:***

1.How many number of non zero elements are present in a tridiagonal matrix of size 65\*65?

A. N\*3-2

=65\*3-2

=195-2

=193

2. What do you mean by the terms overflow and under flow?

A. They describe the situation when something becomes too big or too small to be processed correctly

or stored in the space allocated to it correctly.

3. What is the efficient way for storing two symmetric matrices?

A. Storing either lower triangular or upper triangular matrix.

4. What is the minimum size of the array that has to be created for storing the elements of a lower triangular matrix?

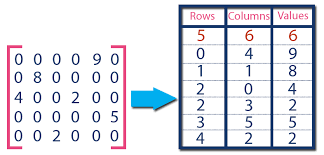
A. If N is the size of matrix, then N\*(N+1)/2 amount of memory is required.

5. Define Sparse matrix.

A. **Sparse matrix** is a **matrix** which contains very few non-zero elements

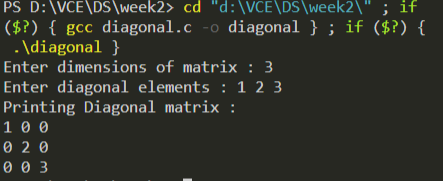
6. What is the efficient way to store the elements of a sparse matrix?

A. Storing only non-zero values along with its position.



***Prelab Questions:***

1. Implement a C program to store and retrieve the elements of a diagonal matrix.
2. #include<stdlib.h>
3. #include<stdio.h>
4. *void* store(*int* \**a*,*int* *n*)
5. {
6. printf("Enter diagonal elements : ");
7. for(*int* i=0;i<n;i++)
8. scanf("%d",&a[i]);
9. }
10. *void* display(*int* \*\**arr*,*int* *n*)
11. {
12. for(*int* i=0,j;i<n;i++)
13. {
14. for(j=0;j<n;j++)
15. printf("%d ",arr[i][j]);
16. printf("\n");
17. }
18. }
19. *void* main()
20. {
21. *int* \*\*arr,\*a,i,j,n;
22. printf("Enter dimensions of matrix : ");
23. scanf("%d",&n);
24. arr=(*int* \*\*)calloc(n,sizeof(*int* \*));
25. a=(*int* \*)calloc(n,sizeof(*int*));
26. store(a,n);
27. for(i=0;i<n;i++)
28. {
29. arr[i]=(*int* \*)malloc(n\*sizeof(*int*));
30. for(j=0;j<n;j++)
31. {
32. if(i!=j)
33. arr[i][j]=0;
34. else
35. arr[i][j]=a[i];
36. }
37. }
38. printf("Printing Diagonal matrix : \n");
39. display(arr,n);
40. }



2. Implement a C program to store and retrieve the elements of a tri-diagonal

matrix.

#include<stdio.h>

#include<stdlib.h>

*void* store(*unsigned* *short* *int*  \**a*,*unsigned* *short* *int*  *n*)

{

    printf("Enter elements of tridiagonal Matrix : \n");

    for(*unsigned* *short* *int*  i=0;i<*n*\*3-2;i++)

        scanf("%hu",&*a*[i]);

}

*void* display(*unsigned* *short* *int*  \*\**arr*,*unsigned* *short* *int*  *n*)

{

    for(*unsigned* *short* *int*  i=0,j;i<*n*;i++)

    {

        for(j=0;j<*n*;j++)

            printf("%hu ",*arr*[i][j]);

        printf("\n");

    }

}

*void* main()

{

*unsigned* *short* *int*  i,j,n,\*a,\*\*arr;

    printf("Enter dimensions of matrix :");

    scanf("%hu",&n);

    a=(*unsigned* *short* *int*  \*)calloc(n\*3-1,sizeof(*unsigned* *short* *int* ));

    arr=(*unsigned* *short* *int*  \*\*)malloc(n\*sizeof(*unsigned* *short* *int*  \*));

    store(a,n);

    for(i=0;i<n;i++)

    {

*unsigned* *short* *int*  jump=0,cur=(i+n-1)%n;

        arr[i]=(*unsigned* *short* *int*  \*)calloc(n,sizeof(*unsigned* *short* *int* ));

        for(j=0;j<n;j++)

        {

            if(j<=i+1&&j>=i-1)

            {

                arr[i][j]=a[jump+cur];

                jump+=n+cur;

                cur=0;

            }

            else

            {

                arr[i][j]=0;

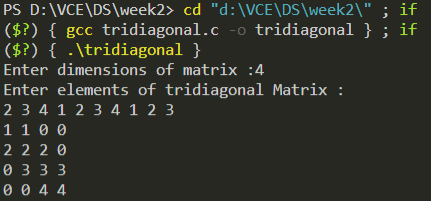
            }

        }

    }

    display(arr,n);

}



3. Implement a C program to store and retrieve the elements of a lower

triangular matrix.

#include<stdio.h>

#include<stdlib.h>

*void* store(*short* *int* \**a*,*short* *int* *n*)

{

    printf("Enter elements of lower trangular matrix : \n");

    for(*int* i=0;i<(*n*\*(*n*+1))/2;i++)

            scanf("%hi",&*a*[i]);

}

*void* display(*short* *int* \*\**arr*,*short* *int* *n*)

{

    for(*short* *int* i=0,j;i<*n*;i++)

    {

        for(j=0;j<*n*;j++)

            printf("%hi ",*arr*[i][j]);

        printf("\n");

    }

}

*void* main()

{

*short* *int* \*a,\*\*arr,i,j,n,k=0;

    printf("Enter dimensions of matrix : ");

    scanf("%hi",&n);

    arr=(*short* *int* \*\*)calloc(n,sizeof(*short* *int* \*));

    a=(*short* *int* \*)calloc((n\*(n+1))/2,sizeof(*short* *int*));

    store(a,n);

    for(i=0;i<n;i++)

    {

        arr[i]=(*short* *int* \*)calloc(n,sizeof(*short* *int*));

        for(j=0;j<n;j++)

        {

            if(j<=i)

            {

                arr[i][j]=a[k];

                k++;

            }

            else

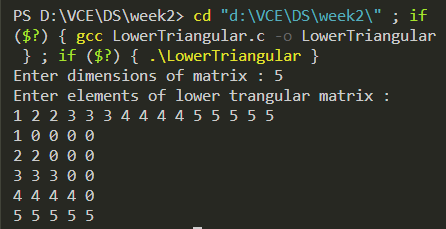
                arr[i][j]=0;

        }

    }

    display(arr,n);

}



4. Implement a C program to store and retrieve the elements of a Symmetric matrix.

#include<stdio.h>

#include<stdlib.h>

*void* store(*short* *int* \**a*,*short* *int* *n*)

{

    printf("Enter elements of upper trangular matrix row wise : \n");

    for(*short* *int* i=0;i<*n*\*(*n*+1)/2;i++)

        scanf("%hi",&*a*[i]);

}

*void* display(*short* *int* \*\**arr*,*short* *int* *n*)

{

    for(*int* i=0,j;i<*n*;i++)

    {

        for(j=0;j<*n*;j++)

            printf("%hi ",*arr*[i][j]);

        printf("\n");

    }

}

*void* main()

{

*short* *int* \*a,\*\*arr,i,j,n,k=0;

    printf("Enter dimensions of martix : ");

    scanf("%hi",&n);

    arr=(*short* *int* \*\*)malloc(n\*sizeof(*short* *int* \*));

    a=(*short* *int* \*)malloc((n\*(n+1)/2)\*sizeof(*short* *int*));

    store(a,n);

    for(i=0;i<n;i++)

    {

        arr[i]=(*short* *int* \*)malloc(n\*sizeof(*short* *int*));

        for(j=0;j<n;j++)

        {

            if(j>=i)

            {

                arr[i][j]=a[k];

                k++;

            }

            else

                arr[i][j]=arr[j][i];

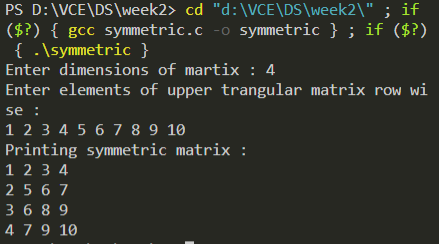
        }

    }

    printf("Printing symmetric matrix : \n");

    display(arr,n);

}



***Lab Programs:***

1.Implement c program to report all occurrences of an element in row wise and

column wise sorted matrix.

bns.h

*short* *int* BinarySearch(*int* \**a*,*int* *x*,*short* *int* *l*,*short* *r*)

{

    if(*l*==*r*) return (*short*)-1;

*short* *int* mid=(*l*+*r*)/2;

    if(*x*==*a*[mid]) return mid;

    if(*x*<*a*[mid]) return BinarySearch(*a*,*x*,*l*,mid);

    else return BinarySearch(*a*,*x*,mid+1,*r*);

}

*int* extreme(*int* \**a*,*int* *x*,*int* *left*,*short* *int* *size*)

{

*int* lo=0;

*int* hi=*size*;

    while(lo<hi)

    {

*int* mid=(lo+hi)/2;

        if(*a*[mid]>*x*||(*left*==1&&*x*==*a*[mid]))

        {

            hi=mid;

        }

        else

        {

            lo=mid+1;

        }

    }

    return lo;

}

*int* BNSrange(*int* \**a*,*int* *x*,*short* *int* *tr*[2],*short* *int* *size*)

{

*tr*[0]=-1;*tr*[1]=-1;

*tr*[0]=extreme(*a*,*x*,1,*size*);

    if(*tr*[0]==*size*||*a*[*tr*[0]]!=*x*)

    {

        return 0;

    }

*tr*[1]=extreme(*a*,*x*,0,*size*)-1;

    return 0;

}

Occurrences

#include<stdio.h>

#include<stdlib.h>

#include"bns.h"

*void* main()

{

*short* *int* m,n,i,j,c;

    printf("Enter dimensions of matrix : ");

    scanf("%hi%hi",&m,&n);

    printf("Enter elements of the array \n");

*int* \*\*arr=(*int* \*\*)malloc(m\*sizeof(*int* \*)),x;

    for(i=0;i<m;i++)

    {

        arr[i]=(*int* \*)malloc(n\*sizeof(*int*));

        for(j=0;j<n;j++)

            scanf("%d",&arr[i][j]);

    }

    printf("Enter key : ");

    scanf("%d",&x);

    for(i=0;i<m;i++)

    {

        /\* c=BinarySearch(arr[i],x,0,m);

        if(c!=-1)

        {

            printf("%d found at position ( %hi , %hi )\n",x,i,c);

        } \*/

*short* *int* tr[2];

        BNSrange(arr[i],x,tr,n);

        if(tr[0]!=-1&&tr[1]!=-1)

        {

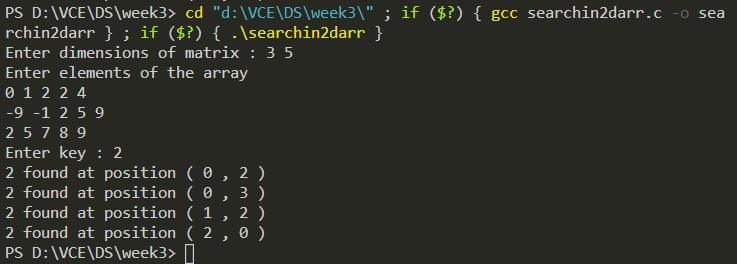
            for(*int* j=tr[0];j<=tr[1];j++)

                printf("%d found at position ( %hi , %hi )\n",x,i,j);

        }

    }

}

2.Upper Triangluar matrix.

#include<stdio.h>

#include<stdlib.h>

*void* store(*short* *int* \**a*,*short* *int* *n*)

{

    printf("Enter elements of upper trangular matrix row wise : \n");

    for(*short* *int* i=0;i<*n*\*(*n*+1)/2;i++)

        scanf("%hi",&*a*[i]);

}

*void* display(*short* *int* \*\**arr*,*short* *int* *n*)

{

    for(*int* i=0,j;i<*n*;i++)

    {

        for(j=0;j<*n*;j++)

            printf("%hi ",*arr*[i][j]);

        printf("\n");

    }

}

*void* main()

{

*short* *int* \*a,\*\*arr,i,j,n,k=0;

    printf("Enter dimensions of martix : ");

    scanf("%hi",&n);

    arr=(*short* *int* \*\*)malloc(n\*sizeof(*short* *int* \*));

    a=(*short* *int* \*)malloc((n\*(n+1)/2)\*sizeof(*short* *int*));

    store(a,n);

    for(i=0;i<n;i++)

    {

        arr[i]=(*short* *int* \*)malloc(n\*sizeof(*short* *int*));

        for(j=0;j<n;j++)

        {

            if(j>=i)

            {

                arr[i][j]=a[k];

                k++;

            }

            else

                arr[i][j]=0;

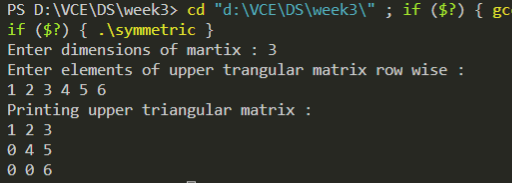
        }

    }

    printf("Printing upper triangular matrix : \n");

    display(arr,n);

}



3.Sparse Matrix storing & retrieval

#include<stdio.h>

#include<stdlib.h>

*struct* sparse{

*short* *int* r;

*short* *int* c;

*int* v;

};

*int* store(*struct* sparse \**lst*,*short* *int* *m*,*short* *int* *n*)

{

*short* *int* i,j,cnt=0;

*int* val;

    for(i=0;i<*m*;i++)

    {

        for(j=0;j<*n*;j++)

        {

            scanf("%d",&val);

            if(val!=0)

            {

                cnt++;

*lst*=(*struct* sparse \*)realloc(*lst*,cnt\*sizeof(*struct* sparse));

*lst*[cnt-1].r=i;

*lst*[cnt-1].c=j;

*lst*[cnt-1].v=val;

            }

        }

    }

    return cnt;

}

*void* print(*struct* sparse \**lst*,*short* *int* *cnt*)

{

*short* *int* i;

    printf("Row \tColumn\t Value\n");

    for(i=0;i<*cnt*;i++)

    {

        printf(" %hi\t   %hi\t   %d\n",*lst*[i].r,*lst*[i].c,*lst*[i].v);

    }

}

*void* main()

{

*short* *int* i,j,m,n,cnt,m2,n2,cnt2,size[4];

*int* val;

    printf("Enter dimensions of Sparse Matrix : ");

    scanf("%hi%hi",&m,&n);

*struct* sparse \*lst=(*struct* sparse \*)malloc(0\*sizeof(*struct* sparse));

    cnt=store(lst,m,n);

    print(lst,cnt);

}

