DAA LAB PROGRAM

Rajnish Kumar(17100042)

PROGRAM 1.1

```
#include<stdio.h>
#include<stdlib.h>
void main(){
	int n,l,h,mid,key;
	printf("Program for Binary Search\n");
	printf("Enter the length of the array\n");
	scanf("%d",&n);
	int a[n],i;
	printf("Enter the element of the increasing sorted array\n");
	for(i=0;i<n;i++)
	scanf("%d",&a[i]);
	l=1;
	h=n;
	mid=(l+h)/2;
	printf("Enter the key which you want to search\n");
	scanf("%d",&key);
	while(l<=h){
		if(a[mid]==key){
			printf("Element Found\n");
			exit(0);
		}
```

```
                else if(a[mid]<key){
                        l=mid+1;
                }
                else{
                        h=mid-1;
                }
        mid=(l+h)/2;
}
        if(l>h)
        printf("Element not found\n");
}
```

PROGRAM 1.2

```
#include<stdio.h>
#include<stdlib.h>
void merging(int a[],int low,int mid,int high){
  int c[100],i,j,k;
  i=low;
     j=mid+1;
     k=low;
     while(i<=mid && j<=high){
                if(a[i]<=a[j]){
                c[k]=a[i];
                  i++;
```

```
                    k++;
            }
                    else{
                            c[k]=a[j];
                            k++;
                            j++;
                    }
        }

            while(i<=mid){

                            c[k]=a[i];
                            k++;
                            i++;


            }

            while(j<=high){

                            c[k]=a[j];
                            k++;
                            j++;
            }



            for(i=0;i<high+1;i++)
            a[i]=c[i];


    }


    void mergeSort(int a[],int low,int high){


                int mid;
                    if(low<high){


                 mid=(low+high)/2;


                 mergeSort(a,low,mid);
                 mergeSort(a,mid+1,high);
                 merging(a,low,mid,high);
        }
    }


                    void main(){
                        int low,high,n,i;
                        printf("Program of merge sort\n");
                        printf("Enter the size of the array\n");
        scanf("%d",&n);


        low=0;high=n-1;

        int a[n];

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

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

        scanf("%d",&a[i]);

        printf("Original array is :\n");

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

        printf(" %d ",a[i]);


        printf("\n");

        printf("Sorted array is\n");


        mergeSort(a,low,high);




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

        printf(" %d ",a[i]);


        printf("\n");


}
```

PROGRAM 1.3

```
#include <stdio.h>

#include <stdlib.h>



int main() {

  int n;

  printf("enter the no of elements\n");

  scanf("%d",&n);

  int a[100],i,low=0,high=n-1;

  printf("enter the elements\n");

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

    scanf("%d",&a[i]);

  }

  quicksort(a,low,high);

  printf("sorted elements are\n");

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

    printf("%d\n",a[i]);

  }

  return 0;

}

int quicksort(int a[],int low,int high){

  int k;

  if(low<high){

    k=partition(a,low,high);
    quicksort(a,low,k-1);
    quicksort(a,k+1,high);
  }
  return 0;
}
int partition(int a[],int low,int high){
  int pivot,i,j,k,t;
  pivot=a[low];
  i=low;
  j=high+1;
  while(i<=j){
    do{
      i=i+1;
    }while(pivot>=a[i]);
    do{
      j=j-1;
    }while(pivot<a[j]);
    if(i<j){
      k=a[i];
      a[i]=a[j];
      a[j]=k;
    }
  }
  t=a[j];
  a[j]=a[low];
  a[low]=t;

  return j;

}
```

PROGRAM 1.4

```
#include<stdio.h>

#include<stdlib.h>>

int main(){

        int n,l,h,mid;

        printf("Program for index Search\n");

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

        scanf("%d",&n);

        int a[n],i;

        printf("Enter non decreasing element of array \n");

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

        scanf("%d",&a[i]);

        l=1;

        h=n;

        mid=(l+h)/2;



                while(l<=h){

                        if(a[mid]==mid){

                                printf("index Found\n");
```

```
                                        exit(0);

                                }

                                else if(a[mid]<mid){

                                        l=mid+1;

                                }

                                else{


                                        h=mid-1;

                            }

                            mid=(l+h)/2;



                }

                if(l>h)

                printf("index not found\n");





  return 0;

}
```

PROGRAM 3.1

```
#include<stdio.h>
```

```
#include<stdlib.h>

void main(){

        int n,l,h,mid,key,new_key;

                printf("checking sum of two elements\n");

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

                scanf("%d",&n);

        int a[n],i;

                printf("Enter the element of the increasing sorted array\n");

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

        scanf("%d",&a[i]);


                printf("Enter the sum you want to search\n");

                scanf("%d",&key);

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

              new_key=key-a[i];

                l=1;

                h=n;

                mid=(l+h)/2;

        while(l<=h){

                if(a[mid]==new_key){

                        printf("sum exist\n");


                        exit(0);

                }

                else if(a[mid]<new_key){
                        l=mid+1;
                }
                else{
                        h=mid-1;
                }
        mid=(l+h)/2;
}
}
        if(l>h)
        printf("sum not found\n");
}
```

PROGRAM 3.2

```
#include<stdio.h>
#include<stdlib.h>
void main(){
        int n,key;
        printf("searching sum of three elements\n");
        printf("Enter the length of the array\n");
        scanf("%d",&n);
        int a[n],i,j,k;
        printf("Enter the element of non decreasing sorted array\n");
        for(i=0;i<n;i++)
```

```
    scanf("%d",&a[i]);

    printf("Enter the sum you want to search\n");

    scanf("%d",&key);

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

        for(j=i+1,k=n-1;j<k;){

       if(a[i]+a[j]+a[k]==key){


                printf("Sum exist\n");

                exit(0);

        }

            else if(a[i]+a[j]+a[k]<key)

            j=j+1;


            else

            k=k-1;




        }


    }

    if(j>k)

    printf("sum not exist\n");

}
```

PROGRAM 3.3

```
#include<stdio.h>

#include<stdlib.h>

void main(){

	int n,l,h,mid,key,new_key;

	printf(" Searching for the sum\n");

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

	scanf("%d",&n);

	int a[n],i,b[n];

	printf("Enter the element of the increasing 1st sorted array\n");

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

	scanf("%d",&a[i]);

		printf("Enter the element of the increasing 2nd sorted array\n");

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

	scanf("%d",&b[i]);


	printf("Enter the key which you want to search\n");

	scanf("%d",&key);


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

		l=1;

	h=n;

	mid=(l+h)/2;

	new_key=key-a[i];

		while(l<=h){
```

```
                if(b[mid]==new_key){
                        printf("sum Found\n");

                        exit(0);
                }
                else if(b[mid]<new_key){
                        l=mid+1;
                }
                else{
                        h=mid-1;
                }
        mid=(l+h)/2;

}
}
        if(l>h)
        printf("sum not found\n");

}
```

PROGRAM 3.4

```
#include<stdio.h>
#include<stdlib.h>
void main(){
        int n,l,h,mid,key;

        printf("Program for Binary Search\n");

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

        scanf("%d",&n);

        int a[n],i;

        printf("Enter the element of the increasing sorted array\n");

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

        scanf("%d",&a[i]);


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

            if(a[i]==a[i+1]){


           printf("Duplicate Found\n");

           exit(0);


            }

          else

         continue;

}

}
```

PROGRAM 4.1

```
#include<stdio.h>

#include<stdlib.h>
```

```
void main(){
    int n,m,i,j,k;



    float sum3=0;
    printf("                    Program for solving Knapsack problem                    for OPTIMAL
SOLUTION        \n");
    printf("Enter the capacity of knapsack\n");
    scanf("%d",&m);
    printf("Enter the weights in knapsack\n");
    scanf("%d",&n);
    int w[n],p[n],pr[n];
    float x[n],pw[n],max;
    printf("Enter each weights \n");
    for(i=0;i<n;i++)
    scanf("%d",&w[i]);
    printf("Enter each profits\n");
    for(i=0;i<n;i++)
    scanf("%d",&p[i]);





    for(i=0;i<n;i++)
       x[i]=0;
```

```
            for(i=0;i<n;i++){
                    pw[i]=(float)p[i]/w[i];


            }




for(i=0;i<n;i++)
    pr[i]=-1;


        for(i=0;i<n;i++){
          max=pw[0];
          for(j=0,k=0;j<n;j++){


              if(pw[j]>max){
            max=pw[j];
            k=j;
        }
              else
          continue;
            }
          pr[i]=k;
          pw[k]=0;


    }
```

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


                   if(m > w[pr[i]]){

                        x[pr[i]] = 1;

                        m = m - w[pr[i]];


                   }

                   else {


                      x[pr[i]] = (float)m/w[pr[i]];

                       break;

               }


             }

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

          sum3=sum3+(p[i]*x[i]);


          printf("Final profit is:%f\n",sum3);


}
```

PROGRAM 4.2

```
#include<stdio.h>

#include<stdlib.h>
```

```
void main(){

        int n,i,j,k,max,max1;




        int sum=0;

        printf("                                        Program for solving job scheduling Algorithm     problem
\n");

        printf("Enter the total jobs \n");

        scanf("%d",&n);


        int p[n],pr[n],p1[n],d[n];


        printf("Enter each profits\n");

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

        scanf("%d",&p[i]);


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

                  p1[i]=p[i];


            printf("Enter deadline of the jobs corresponding to each jobs\n");

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

        scanf("%d",&d[i]);


        max1=d[0];

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

            if(d[i]>max1)
```

```
         max1=d[i];

     else

       continue;


   }


 int job[n];

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

  job[i]=-100;


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

  pr[i]=-1;


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

  max=p[0];

      k=0;

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


  if(p[j]>max){

  max=p[j];

  k=j;

  }

  else

  continue;

  }
```

```
        pr[i]=k;

        p[k]=0;


        }


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

          if(job[d[pr[i]]]==-100)

              job[d[pr[i]]]=pr[i];

          else{

            for(j=d[pr[i]]-1;j>=0;j--)

                  if(job[j]==-100){

                      job[j]=pr[i];

                      break;

                  }

          }


    }


for(i=1;i<=max1;i++)

      sum+=p1[job[i]];


printf("\nNet Profit is: %d \n",sum);

}
```

PROGRAM 4.3

```
#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#include<stdbool.h>


int main(int argc, char const *argv[])

{

        printf("Enter number of nodes in graph: ");

        int n;

        scanf("%d",&n);

        int key[n];

        int weight[n][n];


        int i, j, k;


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

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

                        scanf("%d",&weight[i][j]);

                }

        }




        bool MST[n];

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

                MST[i]=false;
                key[i]=true;
        }



        int min_i=0,min_j=0;
        int min =INT_MAX;

        for(i=0;i<n;i++){
                for(j=0;j<n;j++){
                        if( min>weight[i][j] && weight[i][j]!=0 ){
                                min_i = i; min_j = j;
                                min = weight[i][j];
                        }
                }
        }



        MST[min_i]=true;
        MST[min_j]=true;
        key[min_i]=false;
        key[min_j]=false;

        printf("%d - %d        -> %d\n", min_i+1,min_j+1,min);

        weight[min_i][min_j]=0;
```

```
for(i=0;i<n-2;i++){


    int l,r;

    min=INT_MAX;


    for(l=0;l<n;l++){

        for(r=0;r<n;r++){

            if(MST[l]==false && MST[r]==true && weight[l][r]!=0){

                if(min>weight[l][r]){

                    min=weight[l][r];

                    min_i=l;

                    min_j=r;

                }

            }

        }

    }


    MST[min_i]=true;

    key[min_i]=false;

    printf("%d - %d      -> %d\n", min_j+1,min_i+1,min);


}
```

```
	return 0;
}
```

PROGRAM 4.4

```
#include<stdio.h>

void main()
{
	int n,i,j,im=0,v=0;
	printf("Enter no of vertices:");
	scanf("%d",&n);
	int G[n][n],MST[3][n-1];
	printf("Enter graph:\n\t");
	for(i=0;i<n;i++)
	printf("%c\t",(97+i));
	printf("\n");
	for(i=0;i<n;i++)
	{
		printf("%c\t",(97+i));
		for(j=0;j<n;j++)
		{
			scanf("%d",&G[i][j]);
```

```
        }
}

while(im<n-1)
{
        int min=10000,m_i,m_j,q_i=-1,q_j=-2;
        for(i=0;i<n;i++)
        {
                for(j=0;j<n;j++)
                {
                        if(G[i][j]<min && G[i][j]>0)
                        {
                                min=G[i][j];
                                m_i=i;
                                m_j=j;
                        }
                }
        }
        G[m_i][m_j]=-1;
        G[m_j][m_i]=-1;
        printf("%d %d\n",m_i,m_j);
        for(i=0;i<2;i++)
        {
                for(j=0;j<im;j++)
                {
                    if(MST[i][j]==m_i)
                    {
                            q_i=MST[2][j];
                    }
                    if(MST[i][j]==m_j)
                    {
                            q_j=MST[2][j];
                    }
            }
    }
    printf("%d %d\n",q_i,q_j);
    if(q_i!=q_j)
    {
            if(q_i==-1 && q_j==-2)
            {
                    MST[2][im]=v;
                    v++;
            }

            if(q_i!=-1 && q_j!=-2)
            {
                    if(q_i!=q_j)
                    {
                            for(i=0;i<im;i++)
                            {
```

```
                    if(MST[2][i]==q_j)
                    {
                        MST[2][i]=q_i;
                    }
                }
                MST[2][im]=q_i;
            }
        }

        if(q_i==-1 && q_j!=-2)
        {
            MST[2][im]=q_j;
        }
        if(q_j==-2 && q_i!=-1)
        {
            MST[2][im]=q_i;
        }

        MST[0][im]=m_i;
        MST[1][im]=m_j;
        im++;
    }
}
printf("\nMST is:\n");
for(i=0;i<n-1;i++)
```

```
	{
		printf("%c->%c\n",(97+MST[0][i]),(97+MST[1][i]));
	}
}
```

PROGRAM 4.5

```
#include<stdio.h>
#include<conio.h>

void main()
{
	int n,i,j,k=1,r,cv;
	char p;
	printf("Enter no of vertices:");
	scanf("%d",&n);
	int G[n][n];
	printf("Enter graph:\n\t");
	for(i=0;i<n;i++)
	printf("%c\t",(97+i));
	printf("\n");
	for(i=0;i<n;i++)
	{
		printf("%c\t",(97+i));
		for(j=0;j<n;j++)
```

```
        {
                scanf("%d",&G[i][j]);
        }
}

printf("\nEnter source vertex(character):");
p=getch();
printf("%c\n",p);
r=(int)p-97;

int VIS[n],D[n];

for(i=0;i<n;i++)
{
        VIS[i]=0;
}

VIS[r]=1;D[r]=0;
for(i=0;i<n;i++)
{
        if(VIS[i]==0)
        {
                D[i]=G[r][i];
        }
}
```

```
int min=100;
for(i=0;i<n;i++)
{
        if(min>D[i]&&D[i]!=0)
        {
                min=D[i];
                cv=i;
        }
}

while(k<n)
{
        VIS[cv]=1;
        min=100;
        for(i=0;i<n;i++)
        {
                if(VIS[i]==0)
                {
                        D[i]=(D[i]>(D[cv]+G[cv][i])?(D[cv]+G[cv][i]):D[i]);
                }
        }
        for(i=0;i<n;i++)
        {
                if(VIS[i]==0)
                {
```

```
                                        if(min>D[i] && D[i]!=0)
                                        {
                                                min=D[i];
                                                cv=i;
                                        }
                                }
                        }
                        k++;
                }
                printf("Shotest path lengths are:\n");
                for(i=0;i<n;i++)
                {
                        printf("%c->%c: %d\n",(97+r),(97+i),D[i]);
                }
}
```

PROGRAM 5.1

```
#include <stdio.h>


int main()
{
int i,j,k=0,l=10000,n,s=0,t;
printf("Enter number of matrices");
scanf("%d",&n);
```

```
int a[2*n],c[n][n];

printf("Enter order of matrices");

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

{

scanf("%d %d",&a[i],&a[n+i]);

}

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

{

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

{

c[i][j]=0;

}

}

while(1>0)

{

k=k+1;

s=k;

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

{

s=k+i;

for(j=i;j<s;j++)

{

if(l>c[i][j]+c[j+1][s]+a[i]*a[n+j]*a[s+n])

l=c[i][j]+c[j+1][s]+a[i]*a[n+j]*a[s+n];

t=j;
}
c[i][s]=l;
l=10000;
}
if(c[0][n-1]!=0)
{
break;
}
}

printf("Minimum cost of matrix chain multiplication is %d",c[0][n-1]);
return 0;
}
```

PROGRAM 5.2

```
#include<stdio.h>
#include<stdlib.h>

int W(int i,int j,int *P,int *Q)
{
        if(i==j)
        return Q[j];
        else
        {
```

```
                    return(P[j]+Q[j]+W(i,j-1,P,Q));
          }
}

int C(int i,int j,int* P,int* Q)
{
          if(i==j)
          {
                    return 0;
          }
          else
          {
                    int k,min=1000;
                    for(k=i+1;k<=j;k++)
                    {
                              if(min>(C(i,k-1,P,Q)+C(k,j,P,Q)))
                              {
                                        min=(C(i,k-1,P,Q)+C(k,j,P,Q));
                              }
                    }
                    return (min+W(i,j,P,Q));
          }
}
void main()
{
          int i,j,n;

          printf("Enter no. of nodes:");

          scanf("%d",&n);


          n++;

          int* P;

          P=(int *)malloc(n*sizeof(int));


          P[0]=0;

          printf("P:");

          for(i=1;i<n;i++)

          scanf("%d",&P[i]);


          int* Q;

          Q=(int *)malloc(n*sizeof(int));


          printf("Q:");

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

          scanf("%d",&Q[i]);


          int res=C(0,n-1,P,Q);

          printf("res:%d",res);

}
```

PROGRAM 5.4

```
#include<stdio.h>

#include<stdlib.h>


int** APSP(int** A,int n,int iv,int** P)

{

        int i,j;

        int** N;

        N=(int**)malloc(n*sizeof(int*));

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

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

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

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

        {

                if(A[i][j]>(A[i][iv]+A[iv][j]))

                {

                        N[i][j]=A[i][iv]+A[iv][j];

                        P[i][j]=iv;

                }

                else

                {

                        N[i][j]=A[i][j];

                }
        }
        if(iv==n-1)
        {
                return N;
        }
        else
        {
                iv++;
                APSP(N,n,iv,P);
        }
}

void PATH(int **P,int i,int j)
{
        printf("%d->",(i+1));
        if(P[i][j]==-1)
        {
                printf("%d\n",(j+1));
        }
        else
        {
                PATH(P,P[i][j],j);
        }
}
```

```
void main()

{

        int i,j,n;

        printf("Enter no. of vertices:");

        scanf("%d",&n);

        int** C;

        C=(int**)malloc(n*sizeof(int*));

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

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


        printf("Enter cost adj. matrix:\n");

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

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

        scanf("%d",&C[i][j]);


        int** Path;

        Path=(int**)malloc(n*sizeof(int*));

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

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


        for(i=0;i<n;i++)        //'-1'-> direct path

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

        Path[i][j]=-1;


        int** Res;
```

```
{

        Res=(int**)malloc(n*sizeof(int*));

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

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


        Res=APSP(C,n,0,Path);


        printf("\nResult:\nS->D\tCost\tPath\n");

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

        {

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

                {

                        printf("%d->%d\t%d\t",(i+1),(j+1),Res[i][j]);

                        PATH(Path,i,j);

                }

        }

}
```

PROGRAM 5.5

```
#include<stdio.h>

#include<stdlib.h>


int G(int h,int n,int **C,int *S,int *P)

{
```

```
int i,d=0,count=0;
S[h]=0;

for(i=0;i<n;i++)
if(S[i]==1)
{
        count++;
        d=1;
}

if(d==0)
{
        return C[h][0];
}
else
{
        int min=1000,m_in;
        for(i=0;i<n;i++)
        {
                if(S[i]==1)
                {
                        S[i]=0;
                        if(min>(C[h][i]+G(i,n,C,S,P)))
                        {
                                min=(C[h][i]+G(i,n,C,S,P));
```

```
                                                        m_in=i;
                                                }
                                                S[i]=1;
                                        }
                                }
                                S[m_in]=0;
                                if(min==(C[h][m_in]+G(m_in,n,C,S,P)))
                                P[n-count]=m_in;
                                S[m_in]=1;
                                return min;
                }
}

void main()
{
                int i,j,n;
                printf("Enter no. of vertices:");
                scanf("%d",&n);

                int** C;
                C=(int**)malloc(n*sizeof(int*));
                for(i=0;i<n;i++)
                C[i]=(int*)malloc(n*sizeof(int));

                printf("Enter cost adj. matrix:\n");
```

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

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

        scanf("%d",&C[i][j]);


        int* S;

        S=(int*)malloc(n*sizeof(int));

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

        S[i]=1;


        int* Path;

        Path=(int*)malloc(n*sizeof(int));

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

        Path[i]=0;


        int res=G(0,n,C,S,Path);


        printf("\nOptimal Path will be:\n");

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

        printf("%d-",(Path[i]+1));

        printf("1\nProfit of this tour:%d",res);

}
```