Advanced C++ Program

Paper code:17PCSP02

/ Merge Sorting/

```
#include<iostream.h>
#include<conio.h>
void Merge(int*a, int low, int high, int mid)
{
int i,j,k,temp[50];
temp[high-low+1];
i=low;
k=0;
j=mid+1;
while(i<=mid && j<=high)
{
if(a[i]<a[j])
{
temp[k]=a[i];
k++;
i++;
}
else
{
temp[k]=a[j];
k++;
```

```
j++;


}
}
while(i<=mid)
{
temp[k]=a[i];
k++;
i++;
}
while(j<=high)
{
temp[k]=a[j];
k++;
j++;
}
for(i=low;i<=high;i++)
{
a[i]=temp[i-low];
}
}
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);
merges(a,low,high,mid);
}
}
void main()
{
int n,i,arr[50];
clrscr();
cout<<"\n Enter the number of data element to be sorted:";
cin>>n;
arr[n];
for(i=0;i<n;i++)
{
cout<<"enter element"<<i+1<<":";
cin>>arr[i];
}
mergesort(arr,0,n-1);
cout<<"\n sorted data";
for(i=0;i<n;i++)
{
cout<<endl<<arr[i];
```

```
getch();
}
}
```

Output:

Enter the number of data element to be sorted:7

Enter element 1:34

Enter element 2:67

Enter element 3:23

Enter element 4:56

Enter element 5:12

Enter element 6:89

Enter element 7:45

Sorted data

12

23

34

45

56

67

89

/ Strassen's matrix multiplication/

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a[2][2],b[2][2],c[2][2];
int m1,m2,m3,m4,m5,m6,m7,j,i;
clrscr();
cout<<"matrix multiplication strassen's method\n";
cout<<"enter the element of 2*2 matrix 1:\n";
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
cin>>a[i][j];
}
}
cout<<"enter the element of 2*2 matrix 2:\n";
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
```

```
cin>>b[i][j];
}
}
cout<<"\n First matrix is:\n";
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
cout<<a[i][j]<<"\t";
}
cout<<"\n";
}
cout<<"\n Second matrix is:\n";
for(i=0;i<2;i++)
{
for(j=0;j<2;j++)
{
cout<<b[i][j]<<"\t";
}
cout<<"\n";
}
m1=(a[0][0]+a[1][1])*(b[0][0]+b[1][1]);
m2=(a[1][0]+a[1][1])*b[0][0];
m3=a[0][0]*(b[0][1]-b[1][1]);
```

```
m4=a[1][1]*(b[1][0]-b[0][0]);

m5=(a[0][0]+a[0][1])*b[1][1];

m6=(a[1][0]-a[0][0])*(b[0][0]+b[0][1]);

m7=(a[0][1]-a[1][1])*(b[1][0]+b[1][1]);

c[0][0]=m1+m4-m5+m7;

c[0][1]=m3+m5;

c[1][0]=m2+m4;

c[1][1]=m1+m3-m2+m6;

cout<<"\n Product of both is:\n";

for(i=0;i<2;i++)

{

for(j=0;j<2;j++)

{

cout<<c[i][j]<<"\t";

}

cout<<"\n";

}

getch();

}
```

Output:

Matrix multiplication Strassen's method

Enter the element of 2*2 matrix 1:

8

6

5

4

Enter the element of 2*2 matrix 2:

9

10

12

15

First matrix is:

8 6

5 4

Second matrix is:

9 10

12 15

Product of both is:

144 170

93 110

/ Knapsack problem using Dynamic programming/

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
void knapsack(int n,int c);
int n,i,w,c,k;
int weight[50],v[50];
void main()
{
clrscr();
cout<<"Enter number of items:";
cin>>n;
cout<<"Enter capacity:";
cin>>c;
cout<<"Enter weights:";
for(i=0;i<n;i++)
{
cin>>weight[i];
}
cout<<"Enter values:";
for(i=0;i<n;i++)
{
```

```
cin>>v[i];
}
knapsack(n,c);
getch();
}
void knapsack(int n,int c)
{
for(i=0;i<n;i++)
{
for(int k=i+1;k<n;K++)
{
if(weight[i]<weight[k])
{
int temp,temp1;
temp=weight[k];
temp1=v[k];
weight[k]=weight[i];
v[k]=v[i];
weight[i]=temp;
v[i]=temp1;
}
}
}
int sum=0,j=0;
```

```
while(weight[j]<=c&&j<n)
{
sum=sum+v[j];
c=c-weight[j];
if(weight[j]>=c)
j++;
}
cout<<"knapsack value:";
cout<<sum;
}
```

Output:

Enter number of items:5

Enter capacity:300

Enter weights:

111

121

131

141

151

Enter Value:

11

22

33

44

55

knapsack value:99

/ Minimum Spanning tree/

```
#include<iostream.h>
#include<conio.h>
struct node
{
int fr,to,cost;
}
p[6];
int c,temp1=0;temp=0;
void prims(int*a,int b[][7],int i,int j)
{
a[i]=1;
while(c<6)
{
int min=999
for(int i=0;i<7;i++)
{
if(a[i]==1)
{
for(int j=0;j<7;)
{
if(b[i][j]>=min||b[i][j]==0)
{
```

```
j++;
}
else if(b[i][j]<min)
{
min=b[i][j];
temp=i;
temp1=j;
}
}
}
}
a[temp1]=1;
p[c].fr=temp;
p[c].to=temp1;
p[c].cost=min;
c++;
b[temp][temp1]=b[temp1][temp]=1000;
}
for(int k=0;k<6;k++)
{
cout<<"Source node:"<<p[k].fr<<endl;
cout<<"Destination node:"<<p[k].to<<endl;
cout<<"Weight of node:"<<p[k].cost<<endl;
}
}
```

```
void main()
{
int a[7];
clrscr();
for(int i=0;i<7;i++)
{
a[i]=0;
}
int b[7][7];
for(i=0;i<7;i++)
{
cout<<"Minimum spanning tree and order of traversal\n";
cout<<"Enter value for"<<" "<<(i+1)<<"row"<<endl;
for(int j=0;j<7;j++)
{
cin>>b[i][j];
}
}
prims(a,b,0,0);
getch();
}
```

Output:

Minimum spanning tree and order of traversal:

Enter value for 1 row

0

3

6

0

0

0

0

Enter value for 2 row

3

0

2

4

0

0

0

Enter value for 3 row

6

2

0

1

4

2

0

Enter value for 4 row

0

4

1

0

2

0

4

Enter value for 5 row

0

0

4

2

0

2

1

Enter value for 6 row

0

0

2

0

2

0

1

Enter value for 7 row

0

0

0

4

1

1

0

Source node:0

Destination node:1

Weight of node:3

Source node:1

Destination node:2

Weight of node:2

Source node:2

Destination node:3

Weight of node:1

Source node:2

Destination node:5

Weights of node:2

Source node:5

Destination node:6

Weight of node:1

Source node:6

Destination node:4

Weight of node:1

/ Warshall's Algorithm /

```
#include<iostream.h>
#include<conio.h>
void floyds(int b[][7])
{
int i,j,k;
for(k=0;k<7;k++)
{
for(i=0;i<7;i++)
{
for(j=0;j<7;j++)
{
if((b[i][k]*b[k][j]!=0)&&i!=j
{
if((b[i][k]+b[k][j]<b[i][j]||b[i][j]==0))
{
b[i][j]=b[i][k]+b[k][j];
}
}
}
}


}
for(i=0;i<7;i++)
```

```
{
cout<<"\n Minimum cost with respect to node:"<<i<<endl;
for(j=0;j<7;j++)
{
cost<<b[i][j]<<"\t";
}
}
}
void main()
{
int b[7][7];
clrscr();
cout<<"Enter value of adjacency matrix:\n\n";
for(int i=0;i<7;i++)
{
cout<<"Enter value for"<<" "<<(i+1)<<"row"<<endl;
for(int j=0;j<7;j++)
{
cin>>b[i][j];
}
}
floyds(b);
getch();
}
```

Output:

Enter value of adjacency matrix

Enter value for 1 row

0

3

6

0

0

0

0

Enter value for 2 row

3

0

2

4

0

0

0

Enter value for 3 row

6

2

0

1

4

2

0

Enter value for 4 row

0

4

1

0

2

0

4

Enter value for 5 row

0

0

4

2

0

2

1

Enter value for 6 row

0

0

2

0

2

0

1

Enter value for 7 row

0

0

0

4

1

1

0

Minimum cost with respect to node:0

0 3 5 6 8 7 8

Minimum cost with respect to node:1

3 0 2 3 5 4 5

Minimum cost with respect to node:2

5 2 0 1 3 2 3

Minimum cost with respect to node:3

6 3 1 0 2 3 3

Minimum cost with respect to node:4

8 5 3 2 0 2 1

Minimum cost with respect to node:5

7 4 2 3 2 0 1

Minimum cost with respect to node:6

8 5 3 3 1 1 0

/ Dijikstraris Algorithm /

```
#include<iostream.h>
#include<conio.h>
int shortest(int,int);
int cost[10][10],dist[20],i,j,n,k,m,s[20],v,totcost,path[20],p;
void main()
{
int c;
clrscr();
cout<<"Enter number of vertices:";
cin>>n;
cout<<"Enter number of edges:";
cin>>m;
cout<<"\n Enter \n EGDE cost \n";
for(k=1;k<=m;k++)
{
cin>>i>>j>>c;
cost[i][j]=c;
}
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(cost[i][j]==0)
```

```
cost[i][j]=31999;
cout<<"Enter initial vertex:";
cin>>v;
cout<<v<<"\n";
shortest(v,n);
getch();
}
int shortest(intv,intn)
{
int min;
for(i=1;i<=n;i++)
{
s[i]=0;
dist[i]=cost[v][i];
}
path[++p]=v;
s[v]=1;
dist[v]=0;
for(i=2;i<=n-1;i++)
{
k=-1;
min=31999;
for(j=1;j<=n;j++)
{
if(dist[j]<min&&s[j]!=1)
```

```
{
min=dist[j];
k=j;
}
}
if(cost[v][k]<=dist[k])
p=1;
path[++P]=k;
for(j=1;j<=p;j++)
cout<<path[j];
cout<<"\n";
s[k]=1;
for(j=1;j<=n;j++)
if(cost[k][j]!=31999&&dist[j]>=dost[k]+cost[k][j]&&s[j]!=1)
dist[j]=dist[k]+cost[k][j];
}
}
```

Output:

Enter number of vertices:6

Enter number of edges:11

Enter

EGDE

1 2 50

1 3 45

1 4 10

2 3 10

2 4 15

3 5 30

4 1 10

4 5 15

5 2 20

5 3 30

6 5 3

Enter initial vertex:

1

14

145

1452

13

/ Subset Sum Problem /

```
#include<iostream.h>
#include<conio.h>
int m,n,w[10],x[10];
void sum_subset(int s,int k,int r)
{
int i;
x[k]=1;
if(st w[k]==m)
{
cout<<"Subset with the given sum found \n";
for(i=1;i<=n;i++)
{
cout<<"x["<<i<<"]:"<<x[i]<<"\t";
}
cout<<"\n";
}
else
{
if((s+w[k]+w[k-1])<=m)
{
sum-subset(s+w[k],k+1,r-w[k]);
}
```

```
}
if((s+r-w[k]>=m)&&(s+w[k+1]<=m))
{
x[k]=0;
sum-subset(s,k+1,r-w[k]);
}
}
void main()
{
clrscr();
cout<<"Enter the number of value:";
cin>>n;
cout<<"Enter the sum:";
cin>>m;
cout<<"Enter the value:"<<endl
int i;
for(i=1;i<=n;i++)
{
cin>>w[i];
}
int r;
r=0;
for(i=0;i<=n;i++)
{
r+=w[i];
```

```
}
sum-subset(0,1,r);
getch();
}
```

Output:

Enter the number of value:4

Enter the sum:25

Enter the value:

7

8

5

10

Subset with the given sum

found

x[1]:1 x[2]:1 x[3]:0 x[4]:1

/ Eight queen problem /

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
void print(int board[8][8])
{
for(int i=0;i<8;i++)
{
for(int j=0;j<8;j++)
cout<<setw(4)<<board[i][j];
cout<<endl;
}
}
int back(int board[8][8],int row,int col)
{
int i,j;
for(i=0;i<col;i++)
if(board[row][i])
return 0;
```

```
for(i=row,j=col;i>=0&&i>=0;i--,j--)
if(board[i][j])
return 0;
for(i=row,j=col;j>=0&&i<8;i++,j--)
{
if(board[i][j])
return 0;
}
return 1;
}
int track(int board[8][8],int col)
{
if(col>=8)
return 1;
for(int i=0;i<8;i++)
{
if(back(board,i,col))
{
board[i][col]=1;
if(track(board,col+1))
return 1;
board[i][col]=0;
}
}
return 0;
```

```
}
int queen()
{
int board[8][8]={0};
if(track(board,0)==0)
{
cout<<"Solution does not exist";
return 0;
}
print(board);
return 1;
}
void main()
{
clrscr();
cout<<"\n 8-queens problem solution\n\n";
queen();
getch();
}
```

Output:

8-queens problem solution

1 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0

0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1

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 1 0 0 0 0 0

/ Knapsack problem using backtracking /

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
float p[10]={0},w[10]={0},y[10]={0},x[10]={0};
int i,n,max,k,cp=0,cw=0,fp,fw;
class back
{
public:
void get();
void knapsack(int,int,int);
int bound(int,int,int);
};
void back::get()
{
int i;
cout<<"\n Enter the capacity:";
cin>>max;
cout<<"\n Enter the number of object:";
cin>>n;
```

```
for(i=1;i<=n;i++)
{
cout<<"Enter the weight of the object"<<i<<":";
cin>>w[i];
cout<<"Enter the profit of the object"<<i<<":";
cin>>p[i];
}
}
void back::knapsack(int k,int co,int cw)
{
int j;
if(cw+w[k],+max)
{
y[k]=1;
if(k<n)
knapsack(k+1,cp+p[k],cw+w[k]);
if((cp+p[k]>fp)&&(k==n))
{
fp=cp+p[k];
fw=cw+w[k];
for(j=1;j<=n;++j)
x[i]=y[i];
}
}
if(bound(cp,cw,k)>=fp)
```

```
{
y[k]=0;
if(k<n)
knapsack(k+1,cp,cw);
if((cp>fp)&&(k==n))
{
fp=cp;
fw=cw;
for(j=1;j<=n;j++)
x[i]=y[i];
}
}
}
int back::bound(int cp,int cw,int k)
{
int i,b,c;
b=cp;
c=cw;
for(i=k+1;i<=n;i++)
{
c=c+w[i];
if(c<max)
b=b+p[i];
else
{
```

```
return(b+(1-(c-max)/w[i])*p[i]);
}
}
return b;
}
void main()
{
clrscr();
cout<<"\n Knapsack using backtracking \n";
back obj;
obj.get();
k=1;
cp=0;
cw=0;
obj.knapsack(k,cp,cw);
cout<<"\n Maximum profit of the knapsack is:"<<fp;
cout<<"\n Total weight of the knapsack is:"<<fw;
getch();
}
```

Output:

Knapsack using backtracking

Enter the capacity: 20

Enter the number of object : 3

Enter the weight of the object 1: 20

Enter the profit of the object 1: 34

Enter the weight of the object 2: 25

Enter the profit of the object 2: 39

Enter the weight of the object 3: 13

Enter the profit of the object 3: 44

Maximum profit of the knapsack is: 44

Total weight of the knapsack is: 58

/ Travelling Salesman problem /

```
#include<iostream.h>
#include<conio.h>
#include<iomanip.h>
int a[10][10],visited[10],n,cost=0,least(int);
void get()
{
int i,j;
cout<<"Enter number of cities:";
cin>>n;
cout<<"Enter cost matrix:\n";
for(i=0;i<n;i++)
{
cout<<"Enter element of row:"<<i+1<<"\n";
for(j=0;j<n;j++)
cin>>a[i][j];
visited[i]=0;
}
cout<<"The cost list is:\n";
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
cout<<setw(4)<<a[i][j];
cout<<"\n";
```

```
}
}
void mincost(int city)
{
int i,n;
visited[city]=1;
cout<<city+1<<"->";
int ncity=least(city);
if(ncity==999)
{
ncity=0;
cout<<ncity+1;
cost+=a[city][ncity];
return;
}
mincost(ncity);
}
int least(int c)
{
int i,nc=999;
int min=999,kmin;
for(i=0;i<n;i++)
{
if((a[c][i]!=0)&&(visited[i]==0))
if(a[c][i]<min)
```

```
{
min=a[i][0]+a[c][i];
kmin=a[c][i];
nc=i;
}
}
if(min!=999)
cost+=kmin;
retrun nc;
}
void put()
{
cout<<"\n Minimum cost:";
cout<<cost;
}
void main()
{
clrscr();
get();
cout<<"\n The path is:\n";
mincost(0);
put();
getch();
}
```

Output:

Enter number of cities:3

Enter cost matrix:

Enter element of row:1

0

20

12

Enter element of row:2

33

0

23

Enter element of row:3

34

12

0

The cost list is:

0 20 12

33 0 23

34 12 0

The path is

1->3->2->1

Minimum cost:57