ADA LAB WEEK 6

Yash Gupta
1BM21CS251
25/07/23

Q1) Knapsack problem using C.

```
#include <stdio.h>
int v[10][10];
int p[10],w[10];
int n,m;

int max(int x, int y){
    if(x>y)
        return x;
    else
        return y;
}
void knapsack(){
    int x[10];
    for(int i=0;i<=n;i++){
        for(int j=0;j<=m;j++){
            if(i==0 || j==0){
                v[i][j]=0;
            }
            else if(j-w[i]<0){
                v[i][j]=v[i-1][j];
            }
            else{
                v[i][j]=max(v[i-1][j],v[i-1][j-w[i]]+p[i]);
```

```
            }
        }
    }
    printf("Output is:\n");
    for(int i=0;i<=n;i++){
        for(int j=0;j<=m;j++){
            printf("%d ",v[i][j]);
        }
        printf("\n");
    }
    printf("Highest profit= %d\n",v[n][m]);
    int j=m;
    int i=n;
    printf("Optimal solution\n");
    while(j>0){
        if(v[i][j]!=v[i-1][j]){
            x[i]=1;
            j=j-w[i];
        }
        else{
            x[i]=0;
        }
    }
    for (i = 1; i <= n; i++){
        printf("%d\t", x[i]);
    }
}


int main(){
    printf("enter the number of items:\t");
    scanf("%d",&n);
    printf("enter the max capacity of knapsack:\t");
    scanf("%d",&m);
    printf("enter the weights of all items:\n");
    for(int i=1;i<n;i++){
```

```
        scanf("%d",&w[i]);
    }
    printf("enter the profits of all items:\n");
    for(int i=1;i<n;i++){
        scanf("%d",&p[i]);
    }
    knapsack();
    return 0;
}
```

Output:

```
enter the no. of items: 4

enter the weight of the each item:
1 3 2 4

enter the profit of each item:
10 20
32
16

enter the knapsack's capacity:  5

the output is:
0       0       0       0       0       0

0       10      10      10      10      10

0       10      10      20      30      30

0       10      32      42      42      52

0       10      32      42      42      52


the optimal solution is 52
the solution vector is:
0       1       1       0
```

Q2) Floyds algorithm to find shortest path among nodes.

```
#include<stdio.h>

int min(int a,int b) {
   if(a<b) return(a);
   else return(b);
}

void floyds(int p[10][10],int n) {
   int i,j,k;
   for (k=1;k<=n;k++)
   {
      for (i=1;i<=n;i++)
      {
         for (j=1;j<=n;j++)
               p[i][j]=min(p[i][j],p[i][k]+p[k][j]);
      }
   }
}

int main() {
   int p[10][10],w,n,e,u,v,i,j;
   printf("\n Enter the number of vertices and edges:");
   scanf("%d %d",&n,&e);
   for (i=1;i<=n;i++) {
      for (j=1;j<=n;j++)
      {
         if(i==j)
            p[i][j]=0;
```

```
            else
                p[i][j]=999;
        }
    }
    for (i=1;i<=e;i++) {
        printf("\nEnter the end vertices of edge %d with its weight:\n",i);
        scanf("%d %d %d",&u,&v,&w);
        p[u][v]=w;
    }
    printf("\n Matrix of input data:\n");
    for (i=1;i<=n;i++) {
        for (j=1;j<=n;j++)
            printf("%d \t",p[i][j]);
        printf("\n");
    }
    floyds(p,n);
    printf("\n Transitive closure:\n");
    for (i=1;i<=n;i++) {
        for (j=1;j<=n;j++)
            printf("%d \t",p[i][j]);
        printf("\n");
    }
    return 0;
}
```

Output:

```
 Enter the number of vertices and edges: 4 5

Enter the end vertices of edge 1 with its weight:
2 1 2

Enter the end vertices of edge 2 with its weight:
1 3 3

Enter the end vertices of edge 3 with its weight:
3 4 1

Enter the end vertices of edge 4 with its weight:
3 2 7

Enter the end vertices of edge 5 with its weight:
4 1 6

 Matrix of input data:
0       999     3       999
2       0       999     999
999     7       0       1
6       999     999     0

 Transitive closure:
0       10      3       4
2       0       5       6
7       7       0       1
6       16      9       0
PS D:\1>
```