KnapSack Problem

```
#include <stdio.h>

int max(int a, int b)
{
    return (a > b) ? a : b;
}

int knapSack(int W, int wt[], int val[], int n)
{
    if (n == 0 || W == 0)
        return 0;

    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);
    else
        return max(
            val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),
            knapSack(W, wt, val, n - 1));
}

int main()
{
    int n;
    printf("\nEnter the number of items: ");
    scanf("%d", &n);
    int profit[n], weight[n];
    for (int i = 0; i < n; i++)
    {
        printf("\nEnter the Profit and Weights of Item %d: ", i + 1);
        scanf("%d", &profit[i], weight[i]);
    }
    int W;
    printf("\nEnter the Knapsack Capacity: ");
    scanf("%d", &W);
    printf("\nOptimal Solution: ");
```

```
    printf("%d", knapSack(W, weight, profit, n));
    return 0;
}
```

```
Enter the number of items: 3

Enter the Profit and Weights of Item 1: 1 4

Enter the Profit and Weights of Item 2: 2 5

Enter the Profit and Weights of Item 3: 3 1

Enter the Knapsack Capacity: 4

Optimal Solution: 3
```

Prims Algorithm

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

int cost[10][10], vt[10], et[10][10], vis[10], j, n;
int sum = 0;
int x = 1;
int e = 0;
void prims();

void main()
{
    int i;
```

```
    printf("enter the number of vertices\n");
    scanf("%d", &n);
    printf("enter the cost adjacency matrix\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            scanf("%d", &cost[i][j]);
        }
        vis[i] = 0;
    }
    prims();
    printf("edges of spanning tree\n");
    for (i = 1; i <= e; i++)
    {
        printf("%d,%d\t", et[i][0], et[i][1]);
    }
    printf("weight=%d\n", sum);
    getch();
}

void prims()
{
    int s, min, m, k, u, v;
    vt[x] = 1;
    vis[x] = 1;
    for (s = 1; s < n; s++)
    {
        j = x;
        min = 999;
        while (j > 0)
        {
            k = vt[j];
            for (m = 2; m <= n; m++)
            {
                if (vis[m] == 0)
                {
```

```
                if (cost[k][m] < min)
                {
                    min = cost[k][m];
                    u = k;
                    v = m;
                }
            }
        }
        j--;
    }
    vt[++x] = v;
    et[s][0] = u;
    et[s][1] = v;
    e++;
    vis[v] = 1;
    sum = sum + min;
  }
}
```

```
enter the number of vertices
5
enter the cost adjacency matrix
0 2 999 6 999
2 0 3 8 5
999 3 0 999 7
6 8 999 0 9
999 5 7 9 0
edges of spanning tree
1,2     2,3     2,5     1,4     weight=16
```

Kuskal's Algorithm

```
#include <stdio.h>

int find(int v, int parent[10])
{
   while (parent[v] != v)
   {
      v = parent[v];
   }
   return v;
}

void union1(int i, int j, int parent[10])
{
   if (i < j)
      parent[j] = i;
   else
      parent[i] = j;
}

void kruskal(int n, int a[10][10])
{
   int count, k, min, sum, i, j, t[10][10], u, v, parent[10];
   count = 0;
   k = 0;
   sum = 0;
   for (i = 0; i < n; i++)
      parent[i] = i;
   while (count != n - 1)
   {
      min = 999;
      for (i = 0; i < n; i++)
      {
         for (j = 0; j < n; j++)
         {
```

```
            if (a[i][j] < min && a[i][j] != 0)
            {
                min = a[i][j];
                u = i;
                v = j;
            }
        }
    }
    i = find(u, parent);
    j = find(v, parent);
    if (i != j)
    {
        union1(i, j, parent);
        t[k][0] = u;
        t[k][1] = v;
        k++;
        count++;
        sum = sum + a[u][v];
    }
    a[u][v] = a[v][u] = 999;
}
if (count == n - 1)
{
    printf("spanning tree\n");
    for (i = 0; i < n - 1; i++)
    {
        printf("%d %d\n", t[i][0], t[i][1]);
    }
    printf("cost of spanning tree=%d\n", sum);
}
else
    printf("spanning tree does not exist\n");
}

int main()
{
```

```
    int n, i, j, a[10][10];
    printf("enter the number of nodes\n");
    scanf("%d", &n);
    printf("enter the adjacency matrix\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);
    kruskal(n, a);
    return 0;
}
```

```
enter the number of nodes
5
enter the adjacency matrix

0 2 999 6 999
2 0 3 8 5
999 3 0 999 7
6 8 999 0 9
999 5 7 9 0
spanning tree
0 1
1 2
1 4
0 3
cost of spanning tree=16
```