void dijkstra(int src)

{ //v为点数

//graph 为path[][] src为inputnode1

//double V=4000;

// int dist[V]; // The output array. dist[i] will hold the shortest

// distance from src to i(dist[V]=path[src][V].distance)

bool sptSet[NODENUM]; // sptSet[i] will be true if vertex i is included in shortest

// path tree or shortest distance from src to i is finalized(已知)

//V=NODENUM

// Initialize all distances as INFINITE and stpSet[] as false

for (int i = 0; i < NODENUM; i++)

sptSet[i] = false;// INT\_MAX = MAXLENGTH

// Distance of source vertex from itself is always 0

// dist[src] = 0;

// Find shortest path for all vertices

for (int count = 0; count < NODENUM -1; count++)

{

// Pick the minimum distance vertex from the set of vertices not

// yet processed. u is always equal to src in the first iteration.

int u = minDistance(sptSet);

// Mark the picked vertex as processed

sptSet[u] = true;

// Update dist value of the adjacent vertices of the picked vertex.

for (int v = 0; v < NODENUM; v++)

// Update dist[v] only if is not in sptSet, there is an edge from

// u to v, and total weight of path from src to v through u is

// smaller than current value of dist[v]

if (!sptSet[v] && path[u][v].distance && path[src][u].distance != MAXLENGTH

&& path[src][u].distance + path[u][v].distance < path[src][v].distance)

//dist[v] = dist[u] + graph[u][v];

path[src][v].distance= path[src][u].distance+ path[u][v].distance

}

// print the constructed distance array

printSolution(dist, V);

}

// A utility function to find the vertex with minimum distance value, from

// the set of vertices not yet included in shortest path tree

**int** minDistance(**bool** sptSet[])

{

   // Initialize min value

**int** min = MAXLENGTH, min\_index;

**for** (**int** v = 0; v < NODENUM; v++)

**if** (sptSet[v] == **false** && path[src][v].distance <= min)

         min = path[src][v].distance, min\_index = v;

**return** min\_index;

}