```
/* 邻接矩阵存储 - 有权图的单源最短路算法 */

Vertex FindMinDist( MGraph Graph, int dist[], int collected[] )
{ /* 返回未被收录顶点中dist最小者 */
    Vertex MinV, V;
    int MinDist = INFINITY;

    for (V=0; V<Graph->Nv; V++) {
        if ( collected[V]==false && dist[V]<MinDist) {
            /* 若V未被收录，且dist[V]更小 */
            MinDist = dist[V]; /* 更新最小距离 */
            MinV = V; /* 更新对应顶点 */
        }
    }
    if (MinDist < INFINITY) /* 若找到最小dist */
        return MinV; /* 返回对应的顶点下标 */
    else return ERROR;  /* 若这样的顶点不存在，返回错误标记 */
}

bool Dijkstra( MGraph Graph, int dist[], int path[], Vertex S )
{
    int collected[MaxVertexNum];
    Vertex V, W;

    /* 初始化：此处默认邻接矩阵中不存在的边用INFINITY表示 */
    for ( V=0; V<Graph->Nv; V++ ) {
        dist[V] = Graph->G[S][V];
        if ( dist[V]<INFINITY )
            path[V] = S;
        else
            path[V] = -1;
        collected[V] = false;
    }
    /* 先将起点收入集合 */
    dist[S] = 0;
    collected[S] = true;

    while (1) {
        /* V = 未被收录顶点中dist最小者 */
        V = FindMinDist( Graph, dist, collected );
        if ( V==ERROR ) /* 若这样的V不存在 */
            break;      /* 算法结束 */
        collected[V] = true;  /* 收录V */
        for( W=0; W<Graph->Nv; W++ ) /* 对图中的每个顶点W */
            /* 若W是V的邻接点并且未被收录 */
            if ( collected[W]==false && Graph->G[V][W]<INFINITY ) {
                if ( Graph->G[V][W]<0 ) /* 若有负边 */
                    return false; /* 不能正确解决，返回错误标记 */
                /* 若收录V使得dist[W]变小 */
                if ( dist[V]+Graph->G[V][W] < dist[W] ) {
                    dist[W] = dist[V]+Graph->G[V][W]; /* 更新dist[W] */
                    path[W] = V; /* 更新S到W的路径 */
                }
            }
    } /* while结束*/
    return true; /* 算法执行完毕，返回正确标记 */
}
```