```
/* 邻接矩阵存储 - Prim最小生成树算法 */

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

    for (V=0; V<Graph->Nv; V++) {
        if ( dist[V]!=0 && 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-1作为标记 */
}

int Prim( MGraph Graph, LGraph MST )
{ /* 将最小生成树保存为邻接表存储的图MST，返回最小权重和 */
    WeightType dist[MaxVertexNum], TotalWeight;
    Vertex parent[MaxVertexNum], V, W;
    int VCount;
    Edge E;

    /* 初始化。默认初始点下标是0 */
       for (V=0; V<Graph->Nv; V++) {
        /* 这里假设若V到W没有直接的边，则Graph->G[V][W]定义为INFINITY */
           dist[V] = Graph->G[0][V];
           parent[V] = 0; /* 暂且定义所有顶点的父结点都是初始点0 */
    }
    TotalWeight = 0; /* 初始化权重和     */
    VCount = 0;      /* 初始化收录的顶点数 */
    /* 创建包含所有顶点但没有边的图。注意用邻接表版本 */
    MST = CreateGraph(Graph->Nv);
    E = (Edge)malloc( sizeof(struct ENode) ); /* 建立空的边结点 */

    /* 将初始点0收录进MST */
    dist[0] = 0;
    VCount ++;
    parent[0] = -1; /* 当前树根是0 */

    while (1) {
        V = FindMinDist( Graph, dist );
        /* V = 未被收录顶点中dist最小者 */
        if ( V==ERROR ) /* 若这样的V不存在 */
            break;   /* 算法结束 */

        /* 将V及相应的边<parent[V], V>收录进MST */
        E->V1 = parent[V];
        E->V2 = V;
        E->Weight = dist[V];
        InsertEdge( MST, E );
        TotalWeight += dist[V];
        dist[V] = 0;
        VCount++;

        for( W=0; W<Graph->Nv; W++ ) /* 对图中的每个顶点W */
            if ( dist[W]!=0 && Graph->G[V][W]<INFINITY ) {
            /* 若W是V的邻接点并且未被收录 */
                if ( Graph->G[V][W] < dist[W] ) {
                /* 若收录V使得dist[W]变小 */
                    dist[W] = Graph->G[V][W]; /* 更新dist[W] */
                    parent[W] = V; /* 更新树 */
                }
            }
    } /* while结束*/
    if ( VCount < Graph->Nv ) /* MST中收的顶点不到|V|个 */
       TotalWeight = ERROR;
    return TotalWeight;   /* 算法执行完毕，返回最小权重和或错误标记 */
}
```