求从初始顶点 v 到其他各顶点的最短路径。数组 path[ ]和 dist[ ]的含义与上节相同

```
void Graph_List :: DShortestPath(const int v )
{
        int u , k ;
        int max = 10000;
        Edge *p ;
        int n = graphsize;
        int* path = new int[graphsize];
        int* dist = new int[graphsize];
        int* s = new int[n] ;                                   // 数组 s[i]记录 i 是否被访问过
        for(int i = 0 ; i < n ; i ++ )                  // 数组 path, dist, s 初始化
          { path[i] = -1 ; dist[i] = max ; s[i] = 0 ; }
        dist[v] = 0 ;   s[v] = 1 ;                      // 初始顶点 v 的数组值
        p= Head[v].adjacent;
        u = v ;                                                 // u 为即将访问的顶点
        for(int j = 0 ; j < n ; j ++ )                  // 循环(1)
        {
                // 循环(2)：修改 u 邻接顶点的 s[]值、path[]值和 dist[]值
                while( p != NULL )
                {
                        k = p→ VerAdj ;
                         if( s[k] != 1 && dist[u] + p→cost < dist[k] )
                         {
                                dist[k] = dist[u] + p→cost ;
                                path[k] = u ;
                         }
                        p = p→link ;
                }
                // 循环(3)：确定即将被访问的顶点 u
                int ldist = max;
                for( i = 0 ; i < n ; i ++ )
                        if( dist[i] > 0 && dist[i] < ldist && s[i] == 0 )
                 { ldist = dist[i];    u = i ; }
                s[u] = 1 ;                                      // 访问 u
                p = Head[u].adjacent ;                  // p 为 u 的边链的头指针

        }
        for (i = 0;i<n;i++)  cout<<path[i]<<" ";
        for (i = 0;i<n;i++)      cout<<dist[i]<<" ";
        delete[] path;
        delete[] dist;
}
```