计算关键路径的算法

```
void Graph_List :: CriticalPath ( )
{
    int i , k , e , l ;
    int n = graphsize;
    int* ve = new int [n ] ;                    // 事件的最早发生时间
    int* vl = new int [n ] ;                    // 事件的最迟发生时间
    for ( i = 0 ; i < n ; i ++ )                // 初始化数组 ve[ ]
        ve [ i ] = 0 ;
    for ( i = 0 ; i < n ; i ++ )                // （1）按拓扑顺序计算各事件允许的最早发生时间
    {
        Edge *p = Head [ i ].adjacent ;
        while ( p ! = NULL )
        {
            k = p→ VerAdj ;
            if ( ve [ i ] + p→ cost > ve[k ] )
                ve[k] = ve[i] + p→ cost ;
            p = p→ link ;
        }
    }
    for ( i = 0 ; i < n ; i ++ )                // 数组 vl [ ]初始化
        vl [ i ] = ve[ n – 1 ] ;
    for ( i = n - 2 ; i >= 0 ; i -- )           // （2）逆序计算事件的最迟发生时间
    {
        Edge* p = Head [ i ].adjacent ;
        while ( p != NULL )
        {
            k = p→ VerAdj ;
            if ( vl [ k ] - p→ cost < vl[ i] )
                vl[i] = vl[k] - p→ cost ;
            p = p→ link ;
        }
    }
    // （3）求诸活动的最早开始时间和最迟开始时间
    for ( i = 0 ; i < n ; i ++ )
    {
        Edge* p = Head [ i ].adjacent ;
        while ( p != NULL )
        {
            k = p→ VerAdj ;
            e = ve [ i ] ;
            l = vl [ k ] - p→ cost ;
            if( l == e )
                cout<< "<" << i << "," << k << ">"<< " is Critical Activity! " <<endl ;
            p = p→ link ;
        }
    }
    delete[] ve;
    delete[] vl;
}
```