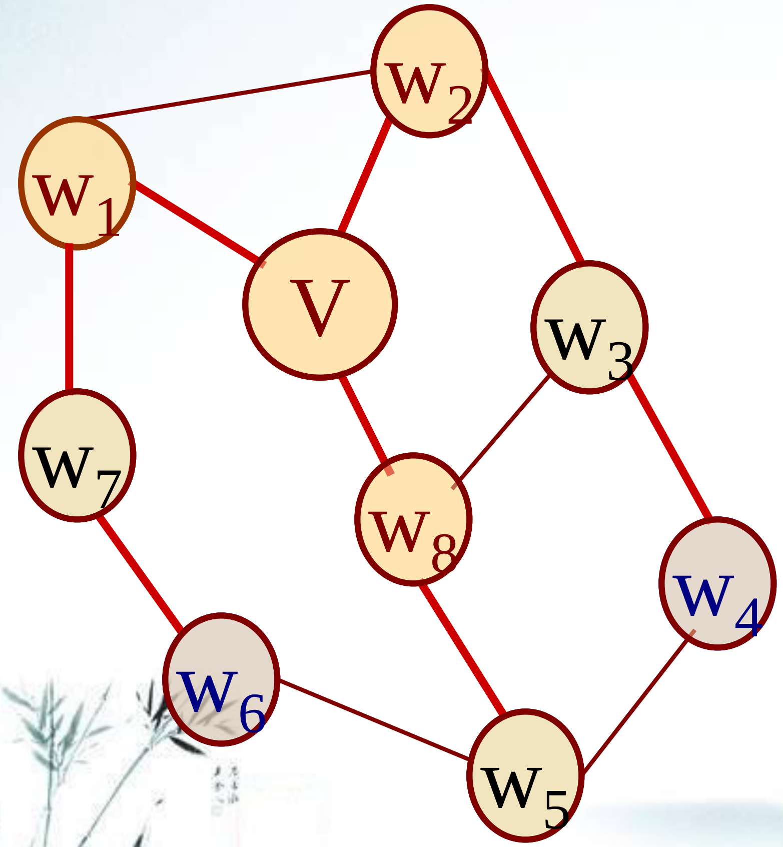
图的遍历

图的遍历

- *Depth-first traversal(深度优先遍历)* of a graph is roughly **analogous to preorder traversal of an ordered tree**. Suppose that the traversal has just visited a vertex *v*, and let *w*1;*w*2……*wk* be the vertices adjacent to *v*. Then we shall next visit *w*1 and keep *w*2……*wk* waiting. After visiting *w*1 , we traverse all the vertices to which it is adjacent before returning to traverse *w*2; ……;*wk* .
- *Breadth-first traversal(广度或宽度优先遍历)* of a graph is roughly analogous to level-by-level traversal*(层序遍历)* of an ordered tree. If the traversal has just visited a vertex *v*, then it next visits *all* the vertices adjacent to *v*, putting the vertices adjacent to these in a waiting list to be traversed after all vertices adjacent to *v* have been visited.

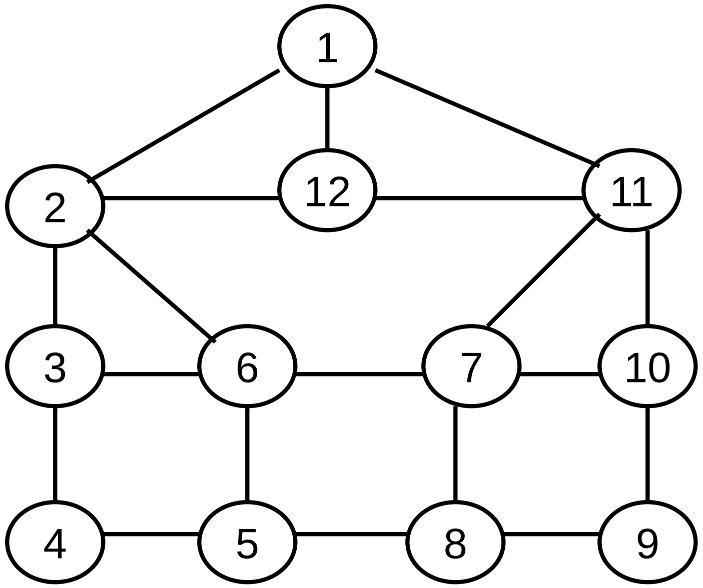
广度优先搜索遍历

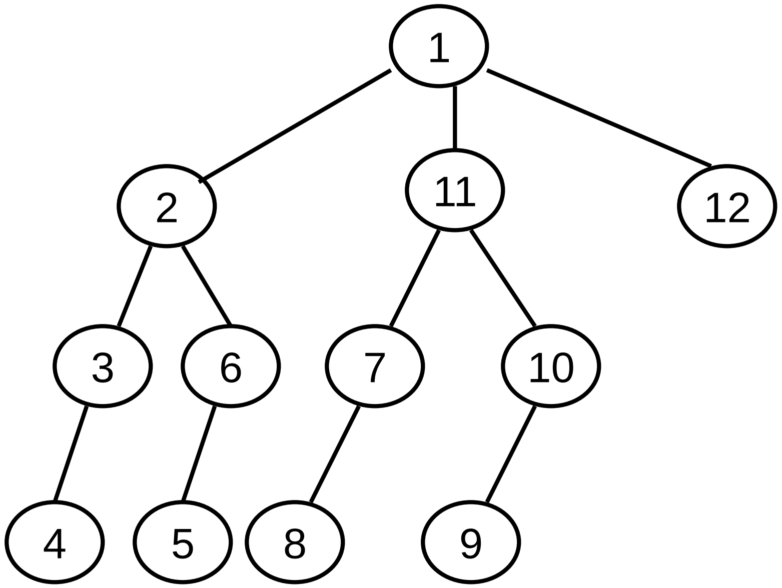


从图中的某个顶点V_0出发，并在访问此顶点之后依次访问V_0的所有未被访问过的邻接点，之后按这些顶点被访问的先后次序依次访问它们的邻接点，直至图中所有和V_0有路径相通的顶点都被访问到。

若此时图中尚有顶点未被访问，则另选图中一个未曾被访问的顶点作起始点，重复上述过程，直至图中所有顶点都被访问到为止。

- 无向图的实例：为了说明问题，邻接结点的访问次序以序号为准。序号小的先访问。如：结点 1 的邻接结点有三个 2、12、11，则先访问结点 2、11，再访问结点 12。

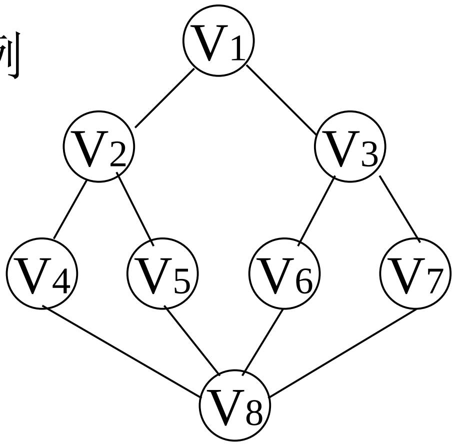




图的广度优先的访问次序：

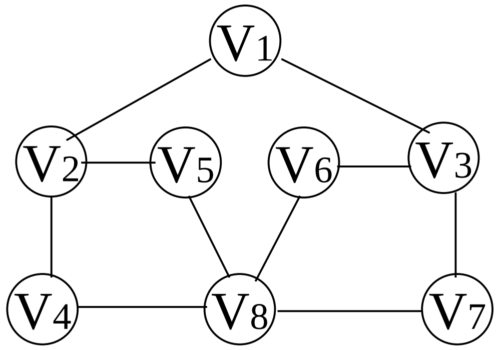
1、2、11、12、3、6、7、10、4、5、8、9

例



广度遍历：V1⇒ V2 ⇒V3 ⇒ V4 ⇒V5 ⇒V6 ⇒V7 ⇒V8

例



广度遍历：V1⇒ V2 ⇒V3 ⇒ V4 ⇒V5 ⇒V6 ⇒V7 ⇒V8

广度优先搜索遍历

❖ Breadth-First算法

```
template <int max_size>
void Digraph<max_size> :: breadth_first(void (*visit)(Vertex &))
        const
/* Post: The function *visit has been performed at each vertex of
        the Digraph in breadth-first order.
Uses: Methods of class Queue . */
{
      Queue q;
      bool visited[max_size];
      Vertex v, w, x;
      for (all v in G) visited[v] = false;
      for (all v in G)
      if (!visited[v]) {
            q.append(v);
```

養天地正氣 法古今完人

广度优先搜索遍历

❖ Breadth-First算法（续）

```
while (!q.empty( )){
q.retrieve(w);
      if (!visited[w]) {
            visited[w] = true;
            (*visit)(w);
            for (all x adjacent to w)
                        q.append(x);
      }
      q.serve( );
      }
  }
}
```

O(n + e)

广度遍历算法需要队列支撑