算法与数据结构
8.2: 图

郝家胜

hao@uestc.edu.cn

School of Automation Engineering,
University of Electronic Sci. & Tech. of China

图的遍历

- 从某个顶点出发，沿着某条**搜索路径**对图中所有顶点各作**一次访问**。
- 若给定的图是连通图，则从图中任一顶点出发，顺着边可以访问该图中所有的顶点。

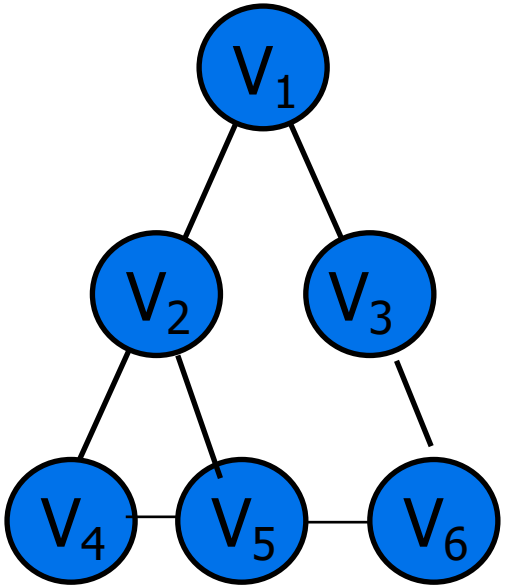
- 遍历过程中，可设置一个标志向量vistited[1…n]，说明哪些顶点已被访问（值为真）。

$$visited[i]=\begin{cases}0\\1 & \text{第i顶点访问过}\end{cases}$$

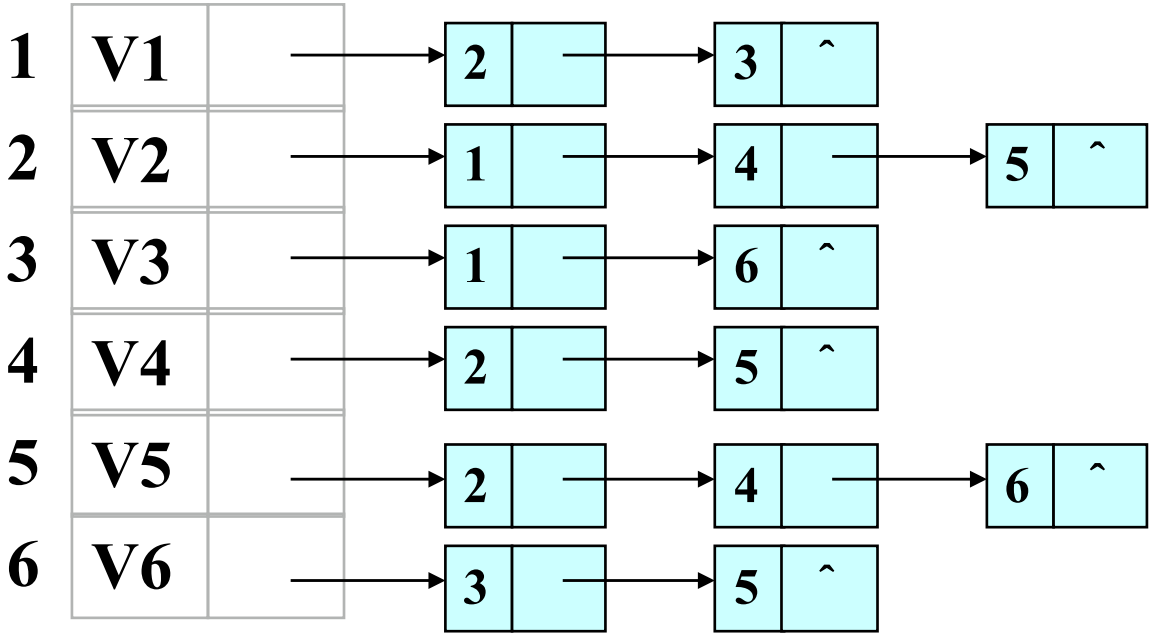
深度优先搜索遍历

- 假设图G中所有顶点均未访问
- 在G中任选一顶点V_i出发：
- 先访问出发顶点V_i，且标记为已访问
- 再搜索V_i的每一个邻接点V_j
- 若V_j未访问过以V_j为新的出发点继续深度优先搜索

递归方式



序号



深度搜索遍历结果：

{ V_1 V_2 V_4 V_5 V_6 V_3 }

（算法24）深度优先搜索遍历算法

- **接口描述**
  - **输入：图、顶点V_0　　输出：遍历的序列**
- **算法分析**
  - **访问顶点V_0**
  - **搜索V_0第一个邻接点V_1**
  - **若V_1存在且未访问过，深度优先搜索V_1**
  - **若V_1存在且访问过，搜索V_0下一个邻接点V_2**
  - **若V_2存在且未访问过，深度优先搜索V_2**

递归方式

搜索结束：所有与V_0有路径相通的顶点都已访问

```
typedef   struct
{ int n;
  char a;
} weighttype;

struct st_arc
{  int adivex;
  weighttype date;
 struct st_arc * nextarc;
 };
```

```
typedef   struct
{ int num;
  char name[10];
} nodetype;

typedef   struct
{  nodetype   vexdata;
struct  st_arc * firstarc;
}headnode;

headnode g[M];

visited[M];
```

```
void  dfs(headnode  g[], int v0 ,
                visited[])
{  int w;
   visited[v0]=1;
   printf("%d", v0);
   w=FIRST_VEX(g, v0);
   while ( w!=0 )
  {   if(visited[w]==0)
         dfs(g,w,visited);
    w=NEXT_VEX(G, v0 ,w );}}
```

应先建立图

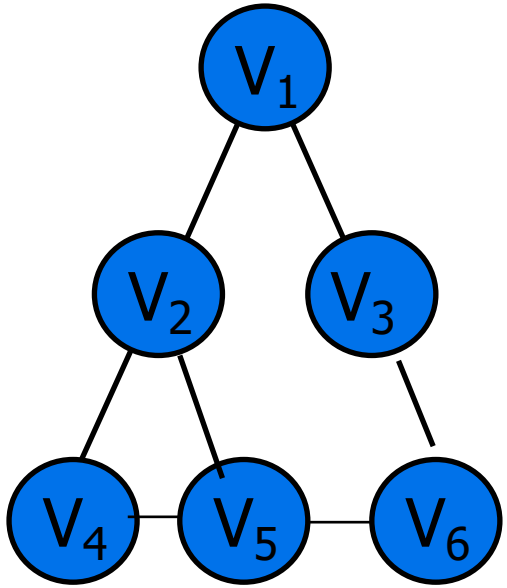
求第一个邻接点的函数

求下一个邻接点的函数

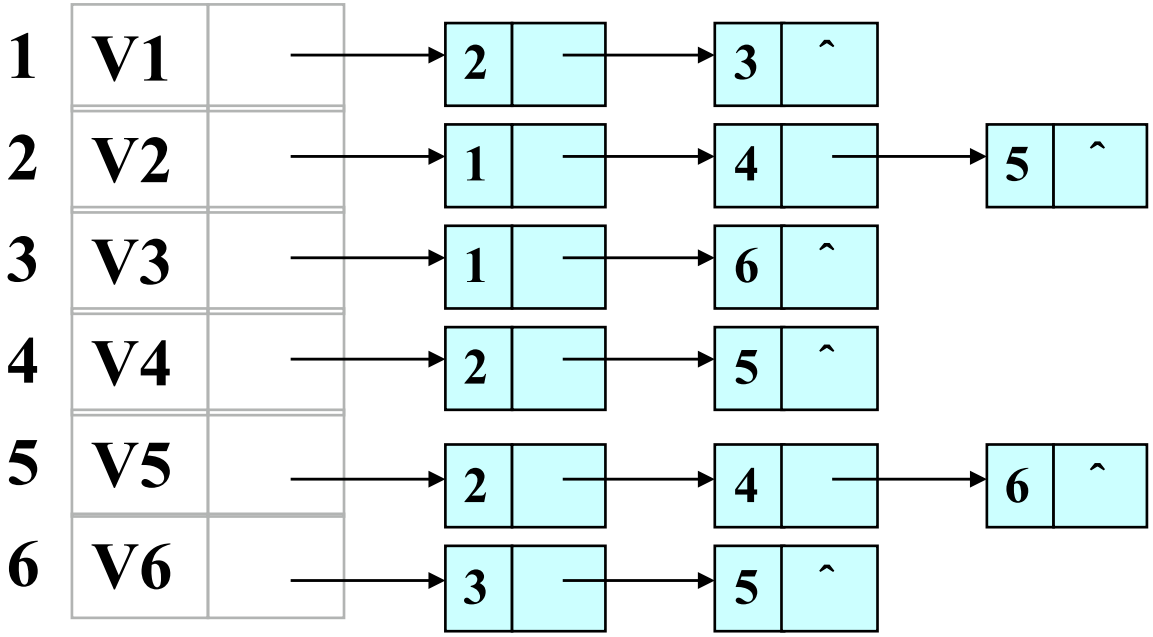
广度优先搜索遍历

- 假设图**G**中所有顶点均未访问
- 在**G**中任选一顶点V_i出发：
- 先访问出发顶点V_i，且标记为已访问
- 再依次访问V_i的所有邻接点$W_1,W_2,\ldots$
- 再依次广度优先搜索访问$W_1,W_2,\ldots$
- 依次类推

先访问的顶点其邻接点先被访问，需要用队列保存已访问过的顶点。



序号



广度搜索遍历结果：

{ V_1 V_2 V_3 V_4 V_5 V_6 }

（算法25）广度优先搜索遍历算法

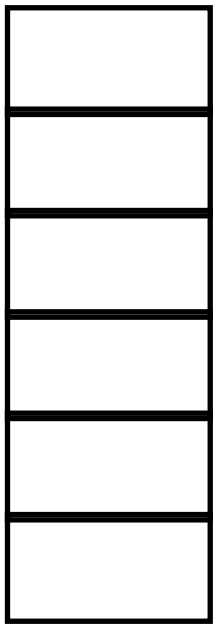
- **接口描述**
  - **输入：图、顶点k　　一输出：遍历的序列**
- **算法分析**
  - **访问顶点k, 并将它入队列**
  - **从队列中取出队头元素，访问它的所有邻接点，并依次入队列**
  - **重复上一步，直至队列为空**

搜索结束：所有与k有路径相通的顶点都已访问

```
void bfs(headnode g[],int k,
                visited[])
{  st_arc *p;
    int w;
    visited[k]=1;
   printf("%d", k);
   enqueue(g_queue, k);
while(empty(g_queue)==0)
{   w=dequeue(g_queue);
     p=g[w].firstarc;
```

队列要先定义

队列queue

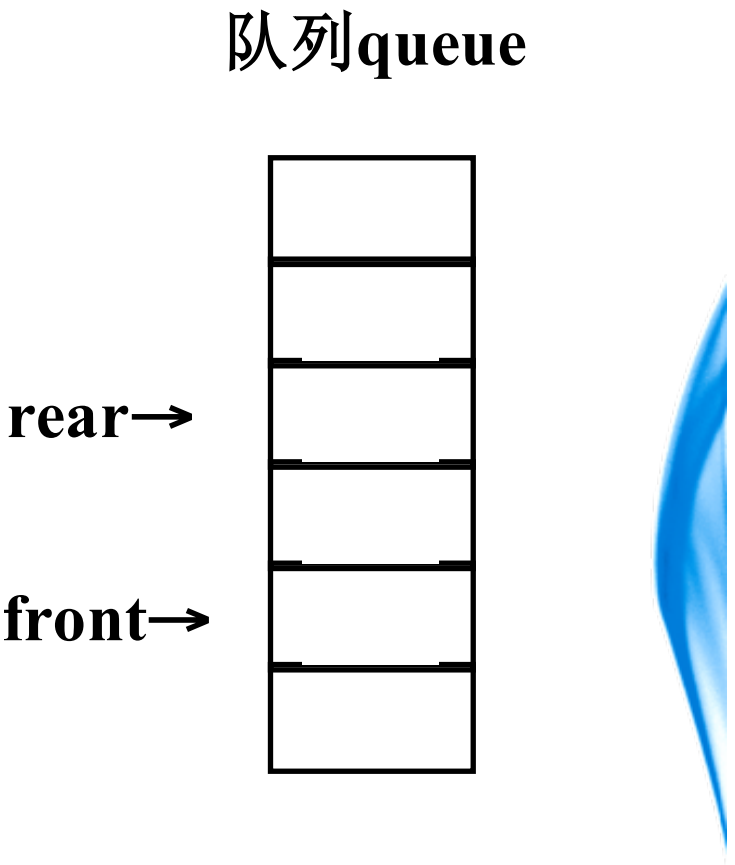


队列不空

rear→

front→

```
while(p!=NULL)
{  if(visited[p->adivex]==0);
   { printf("%d",p->adivex);
      visited[p->adivex]=1;
   enqueue(g_queue,p->adivex);
    }
   p=p->nextarc;
   }
  }
}
```



小结

- 1. 深刻理解图的定义和相关术语。
- 2. 理解图的存储存储；
- 3. 用图的存储存储实现图的遍历；