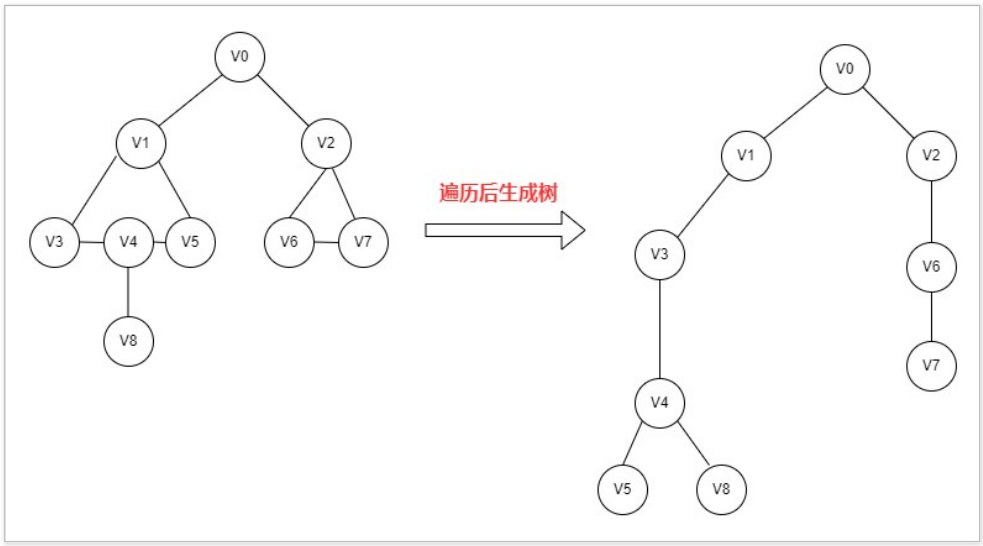
2.3 图的存储之深度优先遍历_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.3 图的存储之深度优先遍历涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

3. 图的存储之深度优先遍历

深度优先遍历的思想类似于我们树的先序遍历 **。初始的时候，图中个顶点都没有被访问，从图中某个顶点（假设为 V0）出发，访问 V0，然后搜索 V0 的一个领接点 Vi。若是 Vi 未被访问，在访问之。（深度优先），按照我们的顶点的大小，根据他在内存中存储，我们可以依次遍历图中所有的数据（规定规则为：从 V0 开始遍历，依次遍历 V0 所连接的点，根据序号大小依次遍历。）。若是所有的顶点都访问完毕。则回溯到它的上一个顶点，然后再从此顶点又按照深度优先的方法搜索下去... 知道所有的顶点都被访问完毕为止。**



首先把所有的结点全部初始值设置为 0，假设从 V0 开始，遍历之后把 V0 的值设置为 1.V0 连接 V1 和 V2。按照从小到大的原则。V0–>V1, 然后把 V1 的值置为 1，然后 V1 连接 V3 和 V4，。V1–>V3, 把 V3 的值设置为 1... 以此类推。V3–>V4...v4–>V5. 这个时候，我们 V5 连接的两个结点 V1 和 V4 都访问过了，然后再回溯到 V4,V4–>V8. 然后再回溯到 V3, 然后再回溯到 V1, 然后再回溯到 V0,V0–>V2, 然后再 V2–>V6, 然后再是 ****V6–>V7

** 最终我们深度遍历得到的结果为：**V0–>V1>V3–>V4–>V5–>V8–>V2–>V6–>V7

	V0	V1	V2	V3	V4	V5	V6	V7	V8
V0	0	1	1	0	0	0	0	0	0
V1	1	0	0	1	0	1	0	0	0
V2	1	0	0	0	0	0	1	1	0
V3	0	1	0	0	1	0	0	0	0
V4	0	0	0	1	0	1	0	0	1
V5	0	1	0	0	1	0	0	0	0
V6	0	0	1	0	0	0	0	1	0

	V0	V1	V2	V3	V4	V5	V6	V7	V8
V7	0	0	1	0	0	0	1	0	0
V8	0	0	0	0	1	0	0	0	0

- ## 类型设计的由来
    - 假设从 V0 开始遍历，遍历完毕之后就就把遍历过的顶点置为 1. 没有遍历过的记录为 0。

因此，我们在设计数据类型的时候，我们就需要定义一个数组 **，表示我们的顶点是否被 **

** 遍历过,** 遍历过则置为 1, 否则设置为 0.

- 假设我们的图用领接矩阵存储的话，我们就需要遍历邻接矩阵并且相连的最小值的编号，判断该编号的顶点是否被遍历过，没有遍历过则递归继续遍历。若是已经遍历了。找到下一个顶点继续遍历。

- ## 类型设计

```
#define  N  9

typedef  int  vertex_t;

typedef struct
{
     vertex_t  v[N];
     int matrix[N][N];
}graph_t;

int visited[N];
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef int vertex_t;

#define N 9

typedef struct
{
        vertex_t v[N];
        int matrix[N][N];
}grapht_t;

int visited[N];


grapht_t *create_graph()
{
```

```
    grapht_t *g = NULL;
    int i = 0;
    g = (grapht_t *)malloc(sizeof(grapht_t));
    if(NULL == g)
    {
            printf("%s : mallco is fail\n",__FUNCTION__);
            return NULL;
    }
    memset(g,0,sizeof(grapht_t));
```

```
    for(i = 0;i < N;i++)
    {
            g->v[i] = i;
    }
```

```
    return g;
}



void input_edge(grapht_t *g)
{
    int i = 0,j = 0;
    int ret = 0;
    printf("please input (V0,V1) (V0,V1).... :\n");



    while(scanf("(V%d,V%d)",&i,&j) == 2)
    {
            getchar();
            g->matrix[i][j] = g->matrix[j][i] = 1;
    }

    while(getchar() != '\n');
    return;
}




int print_matrix(grapht_t *g)
{
        int i = 0,j = 0;







        printf("%3c",' ');



        for(i = 0;i < N;i++)
        {
                printf("V%-2d",i);
        }
        putchar('\n');




        for(i = 0;i < N;i++)
        {
                printf("V%-2d",i);
                for(j = 0;j < N;j++)
                {

                        printf("%-3d",g->matrix[i][j]);
                }
                putchar('\n');
        }
        return 0;
}



int first_adj(grapht_t *g,int v)
{
        int i = 0;

        for(i = 0;i < N;i++)
        {
                if(g->matrix[v][i] != 0)
                        return i;
        }
        return -1;
}


int next_adj(grapht_t *g,int v,int u)
{
        int i = 0;

        for(i = u + 1;i < N;i++)
        {
```

```
            if(g->matrix[v][i] != 0)
                    return i;
        }
        return -1;
}



void DFS(grapht_t *g,int v)
{

        int u = 0;


        printf("V%-3d",v);

        visited[v] = 1;


        u = first_adj(g,v);


        while(u >= 0)
        {

                if(visited[u] == 0)
                {

                        DFS(g,u);
                }


                u = next_adj(g,v,u);
        }
        return ;
}

int main()
{
        int i = 0,j = 0;

        grapht_t *g = NULL;

        g =  create_graph();

        input_edge(g);

        print_matrix(g);


        DFS(g,0);
        putchar('\n');
        return 0;
}
```

全文完