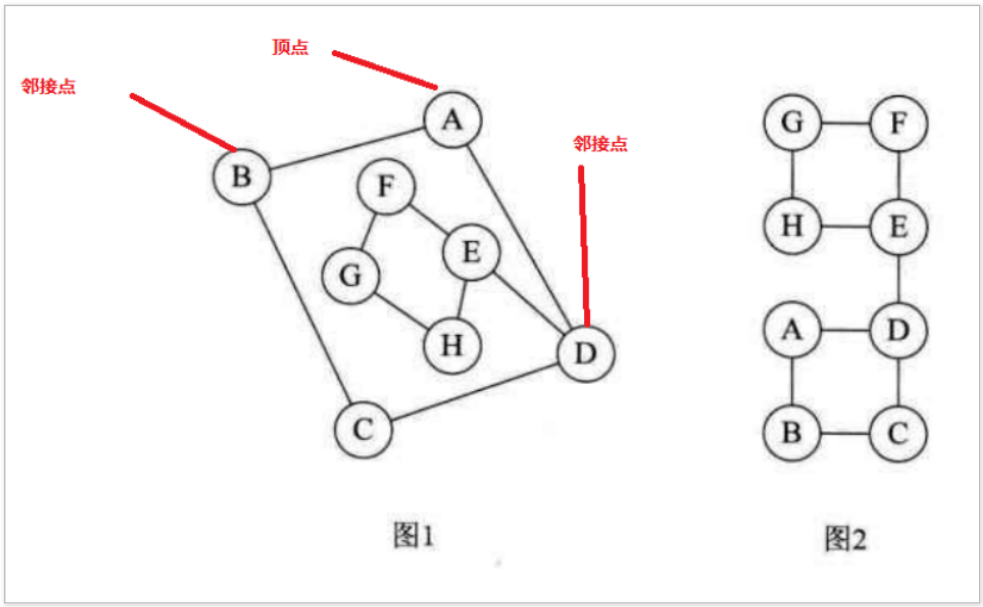
2.2 图的存储 --- 邻接矩阵_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 2.2 图的存储 ––– 邻接矩阵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。

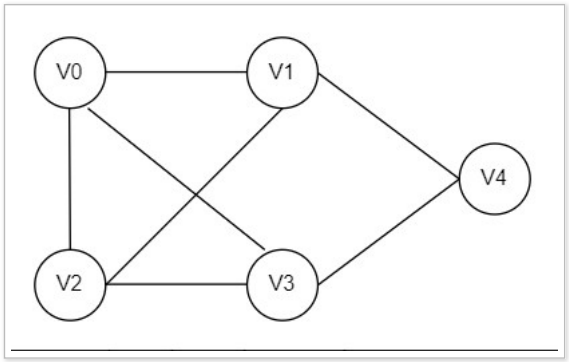
2. 图的存储—邻接矩阵

图的存储相对于我们前面学习的线性表和树来说就显得更为的复杂了。首先，我们需要了解两个概念，叫做“顶点的位置”或者“邻接点的位置”。从图的逻辑上说，图中任何一个顶点都可以被看成是第一个顶点，任何一个顶点的邻接点也不存在层次关系。如下图。



上面的 2 个图，若是我们仔细的观察可以发现，他们其实是一个图，只不过是顶点的位置不一样而已。所以，给人感觉是两张图。既然，我们图形的变化可能存在多种多样的！下面我们就来介绍邻接矩阵的用法。

对于一个具有 n 个节点的图，我们可以使用 n * n 的矩阵来表示他们之间的邻接关系，既然是矩阵，我们肯定就会想到我们 C 语言之中的二维数组。如下图，若是我们想要存储下面这个图案该怎么办呢?



我们的二维数组来表示，把我们把有两个顶点相连的置为 1，没有相连接的顶点置为 0(自己和自己也表示没有关系)，

	V0	V1	V2	V3	V4
V0	0	1	1	1	0
V1	1	0	1	0	1
V2	1	1	0	1	0
V3	1	0	1	0	1
V4	0	1	0	1	0

图的邻接矩阵（Adjacency Matrix）存储方式就是用两个数组来表示图，一个一维数组存储图中的顶点信息，一个二维数组（称为邻接矩阵）存储图中的边的信息。

```
typedef int vertex_t;

#define N 5
typedef struct
{

    vertex_t V[N];

    int matrix[N][N];
}grapht_t;



graph_t *create_graph()
{
    graph_t *g = NULL;
    int i = 0;


    g = (graph_t *)malloc(sizeof(graph_t));
    memset(g,0,sizeof(graph_t));

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

    return g;
}



void input_edge(graph_t *g)
{
    int i = 0,j = 0;


    printf("Input edge like (V0,V1) (V0,V2) ... a

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

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

    return ;
}
```

```
int print_matrix(graph_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 main()
{
    graph_t *g = NULL;

    g = create_graph();

    input_edge(g);

    print_matrix(g);

    return 0;
}



Input edge link (V0,V1) (V0,V2) ...
(V0,V1) (V0,V2) (V0,V3) (V1,V2) (V1,V4) (V2,V3) (V3,V4)
     V0    V1    V2    V3    V4
V0   0     1     1     1     0
V1   1     0     1     0     1
V2   1     1     0     1     0
V3   1     0     1     0     1
V4   0     1     0     1     0
```

全文完