图

图的数据结构(邻接矩阵)

```
typedef struct GNode *PtrToGNode;
struct GNode{
    int Nv;  /* 顶点数 */
    int Ne;  /* 边数   */
    WeightType G[MaxVertexNum][MaxVertexNum]; /* 邻接矩阵, 边表*/
};
typedef PtrToGNode MGraph; /* 以邻接矩阵存储的图类型 */
```

深度优先遍历（递归）

```
void DFS(MGraph graph, int v){
    visited[v] = true;
    visit(v);
    for(int i = 0; i < graph->Nv; ++i){
        if(graph->G[v][i] == 1 && !visited[i])
            DFS(graph, i);
    }
}
```

深度优先遍历（迭代）

```
void DFS(MGraph graph, int v){
    stack<int> st;
    visit(v);
    visited[v];
    st.push(v);
    while(!st.empty()){
        int data, i;
        data = st.top();
        for(i = 0; i < graph->Nv; ++i){
            if(graph->G[data][i] == 1 && visited[i] == 1){
                visit(i);
                visited[i] = true;
                st.push(v);
                break;
            }
        }
        if(i == graph->Nv) st.pop();
    }
}
```

图的广度优先遍历(迭代)

```
void BFS(MGraph graph, int v){
    queue<int> que;
    int vertex;
    visit(v);
    visited[v] = true;
    que.push(v);
    while(!que.empty()){
        vertex = que.front();
        que.pop();
        for(int i = 0; i < graph->Nv; ++i){
            if(graph->G[v][i] == 1 && !visited[i]){
                visit(i);
                visited[i] = true;
                que.push(i);
            }
        }
    }
}
```