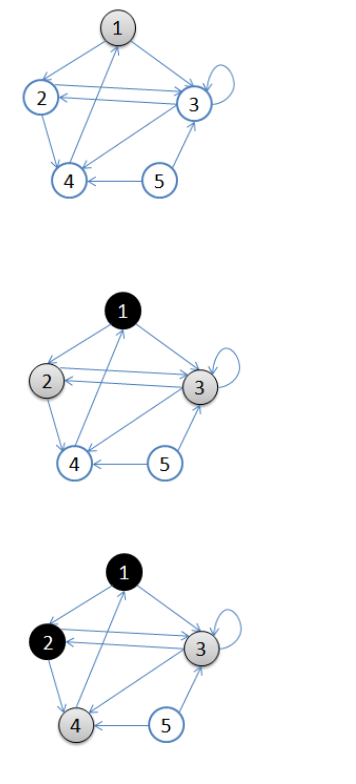
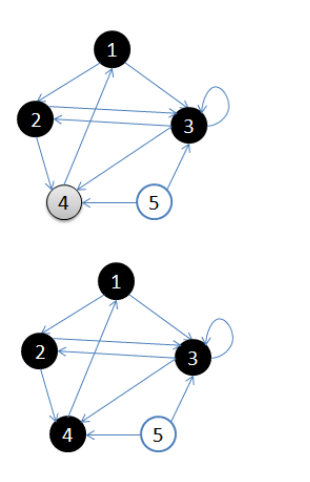
**AI课上代码实现 第五次试验 硬件一班 王倩倩 171491121**

1. BFS

****广度优先搜索（BFS）****  
广度优先搜索在进一步遍历图中顶点之前，先访问当前顶点的所有邻接结点。  
a .首先选择一个顶点作为起始结点，并将其染成灰色，其余结点为白色。  
b. 将起始结点放入队列中。  
c. 从队列首部选出一个顶点，并找出所有与之邻接的结点，将找到的邻接结点放入队列尾部，将已访问过结点涂成黑色，没访问过的结点是白色。如果顶点的颜色是灰色，表示已经发现并且放入了队列，如果顶点的颜色是白色，表示还没有发现  
d. 按照同样的方法处理队列中的下一个结点。  
基本就是出队的顶点变成黑色，在队列里的是灰色，还没入队的是白色。





从顶点1开始进行广度优先搜索：

1. 初始状态，从顶点1开始，队列={1}
2. 访问1的邻接顶点，1出队变黑，2,3入队，队列={2,3,}
3. 访问2的邻接结点，2出队，4入队，队列={3,4}
4. 访问3的邻接结点，3出队，队列={4}
5. 访问4的邻接结点，4出队，队列={ 空}  
   结点5对于1来说不可达。

实验代码：

#include <iostream>

#include <queue>

#define N 5

using namespace std;

int maze[N][N] = {

{ 0, 1, 1, 0, 0 },

{ 0, 0, 1, 1, 0 },

{ 0, 1, 1, 1, 0 },

{ 1, 0, 0, 0, 0 },

{ 0, 0, 1, 1, 0 }

};

int visited[N + 1] = { 0, };

void BFS(int start)

{

queue<int> Q;

Q.push(start);

visited[start] = 1;

while (!Q.empty())

{

int front = Q.front();

cout << front << " ";

Q.pop();

for (int i = 1; i <= N; i++)

{

if (!visited[i] && maze[front - 1][i - 1] == 1)

{

visited[i] = 1;

Q.push(i);

}

}

}

}

int main()

{

for (int i = 1; i <= N; i++)

{

if (visited[i] == 1)

continue;

BFS(i);

}

return 0;

}

运行结果：

