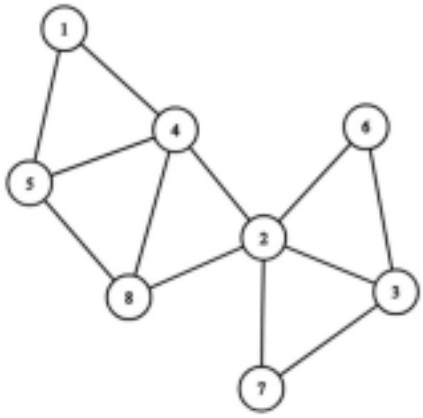
Duyệt đồ thị theo chiều sâu

```c
/* Khai báo CTDL Stack*/

#define MAX_SIZE 100
typedef int ElementType;
typedef struct {
      ElementType data[MAX_SIZE];
      int top_idx;
} Stack;

/* Hàm khởi tạo ngăn xếp rỗng */
void make_null_stack(Stack *pS) {
      pS->top_idx = -1;
}

/* Hàm thêm phần tử u vào đỉnh ngăn xếp */
void push(Stack *pS, ElementType u) {
      pS->top_idx++;
      pS->data[pS->top_idx] = u;
}

/* Hàm xem phần tử trên đỉnh ngăn xếp */
ElementType top(Stack *pS) {
      return pS->data[pS->top_idx];
}

/* Hàm xoá bỏ phần tử trên đỉnh ngăn xếp */
void pop(Stack *pS) {
      pS->top_idx--;
}

/* Hàm kiểm tra ngăn xếp rỗng */
int empty(Stack *pS) {
      return pS->top_idx == -1;
}
```

```
//Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
int mark[MAX_N];

void DFS(Graph *pG, int s) {
      //1. Khai báo ngăn xếp S, khởi tạo rỗng
      Stack S;
      make_null_stack(&S);

      //2. Đưa s vào S, bắt đầu duyệt từ đỉnh s */
      push(&S, s);

      //3. Vòng lặp chính dùng để duyệt
      while (!empty(&S)) {

            //3a. Lấy phần tử trên đỉnh S ra
            int u = top(&S); pop(&S);
            if (mark[u] != 0)              //u đã duyệt rồi, bỏ qua
                  continue;

            printf("Duyet %d\n", u);       //Làm gì đó trên u
            mark[u] = 1;                   //Đánh dấu nó đã duyệt

            //3b. Xét các đỉnh kề của u, đưa vào ngăn xếp S
            for (int v = 1; v <= pG->n; v++)
                  if (adjacent(pG, u, v))
                        push(&S, v);
      }
}
```

Bài tập: Chạy từng dòng code bằng tay và thực hiện duyệt đồ thị sau đây theo chiều sâu (hàm DFS). Lưu ý: mảng mark được khởi tạo bằng 0



Duyệt đồ thị theo chiều rộng

```c
/* Khai báo CTDL Queue*/

#define MAX_SIZE 100
typedef int ElementType;

typedef struct {
      ElementType data[MAX_SIZE];
      int front, rear;
} Queue;

/* Khởi tạo hàng đợi rỗng */
void make_null_queue(Queue *pQ) {
      pQ->front = 0;
      pQ->rear = -1;
}
/* Đưa phần tử u vào cuối hàng đợi */
void enqueue(Queue *pQ, ElementType u) {
      pQ->rear++;
      pQ->data[pQ->rear] = u;
}

/* Xem phần tử đầu hàng đợi */
ElementType front(Queue *pQ) {
      return pQ->data[pQ->front];
}

/* Xoá bỏ phần bỏ đầu hàng đợi */
void dequeue(Queue *pQ) {
      pQ->front++;
}

/* Kiểm tra hàng đợi rỗng */
int empty(Queue *pQ) {
      return pQ->front > pQ->rear;
}
```

```c
//Biến hỗ trợ dùng để lưu trạng thái của đỉnh: đã duyệt/chưa duyệt
int mark[MAX_N];

void BFS(Graph *pG, int s) {
	//1. Khai báo hàng đợi Q, khởi tạo rỗng
	Queue Q;
	make_null_queue(&Q);

	//2. Đưa s vào Q, bắt đầu duyệt từ đỉnh s */
	enqueue(&Q, s);

	//3. Vòng lặp chính dùng để duyệt
	while (!empty(&Q)) {

		//3a. Lấy phần tử ở đầu hàng đợi
		int u = front(&Q); dequeue(&Q);
		if (mark[u] != 0)			//u đã duyệt rồi, bỏ qua
			continue;

		printf("Duyet %d\n", u);	//Làm gì đó trên u
		mark[u] = 1;				//Đánh dấu nó đã duyệt

		//3b. Xét các đỉnh kề của u, đưa vào hàng đợi Q
		for (int v = 1; v <= pG->n; v++)
			if (adjacent(pG, u, v))
				enqueue(&Q, v);
	}
}
```

```c
//Duyệt đồ thị theo chiều rộng từ đỉnh 1
#include <stdio.h>

//Các khai báo và cài đặt
...

int main() {
	//1. Khai báo đồ thị G
	Graph G;

	//2. Đọc dữ liệu và dựng đồ thị
	...

	//3. Khởi tạo mảng mark[u] = 0, với mọi u = 1, 2, ..., n
	for (int u = 1; u <= G.n; u++)
		mark[u] = 0;

	//4. Gọi hàm BFS duyệt theo chiều rộng từ đỉnh 1
	BFS(&G, 1);

	return 0;
}
```

Bài tập: Chạy từng dòng code bằng tay và thực hiện duyệt đồ thị sau đây theo chiều rộng (hàm BFS). Lưu ý: mảng mark được khởi tạo bằng 0 (hàm main):