### 2.1.1 Duyệt theo chiều sâu

Ta sử dụng một ngăn xếp để lưu trữ L và một mảng mark[] để kiểm tra xem một đỉnh đã được duyệt chưa, khởi tạo tất cả các đỉnh đều chưa duyệt. Bạn cần phải định nghĩa một cấu trúc dữ liệu Ngăn xếp (Stack) với các phép toán:

- make_null_stack(S): tạo ngăn xếp rỗng.
- push(S, x): đưa x vào ngăn xếp.
- top(S): trả về phần tử đầu trên ngăn xếp.
- pop(S): loại bỏ phần tử đầu danh sách.
- empty(S): kiểm tra ngăn xếp có rỗng hay không.

```
/* Khai bao Stack*/


/* Duyet do thi theo chieu sau */
void depth_first_search(Graph* G) {
  Stack L;
  int mark[MAX_VERTEXES];
  make_null_stack(&L);

  /* Khởi tạo mark, chưa đỉnh nào được duyệt */
  int j;
  for (j = 1; j <= G->n; j++)
    mark[j] = 0;

  /* Đưa 1 vào L, bắt đầu duyệt từ đỉnh 1 */
  push(&L, 1);

  /* Vòng lặp chính dùng để duyệt */
  while (!empty(&L)) {
    /* Lấy phần tử đầu tiên trong L ra */
    int x = top(&L); pop(&L);
    if (marl[x] != 0) // Đã duyệt rồi, bỏ qua
      continue;
    printf("Duyet %d\n", x);
    mark[x] = 1; //Đánh dấu nó đã duyệt
    /* Lấy các đỉnh kề của nó */
    List list = neighbors(G, x);
    /* Xét các đỉnh kề của nó */
    for (j = 1; j <= list.size; j++) {
      int y = element_at(&list, j);
      push(&L, y);
    }
  }
}
```

Chú ý:

- Các đỉnh trong stack là các đỉnh sẽ được duyệt.
- Một đỉnh có thể có mặt trong stack nhiều lần. Vì thế, khi lấy nó ra cần phải kiểm tra nó đã được duyệt chưa.
- Nếu đồ thị được cài đặt bằng phương pháp ma trận kề, bạn có thể không cần tới hàm **neighbors** làm gì.

Bạn có thể sử dụng khai báo ngăn xếp như thế này:

```
/* Khai bao Stack*/
#define MAX_ELEMENTS 100
typedef struct {
  int data[MAX_ELEMENTS];
  int size;
} Stack;

void make_null_stack(Stack* S) {
  S->size = 0;
}

void push(Stack* S, int x) {
  S->data[S->size] = x;
  S->size++;
}

int top(Stack* S) {
  return S->data[S->size - 1];
}

void pop(Stack* S) {
  S->size--;
}

int empty(Stack* S) {
  return S->size == 0;
}
```

### 2.1.2 Bài tập 2.1

Viết chương trình đọc đồ thị từ tập tin và duyệt đồ thị theo chiều sâu.

### 2.1.3 Duyệt theo chiều rộng

Tương tự như duyệt theo chiều sâu, ta cần một hàng đợi để lưu trữ L. Với phương pháp duyệt theo chiều sâu, ta cần chú ý điều chỉnh 1 chút ở chỗ duyệt đỉnh x (hơi khác 1 chút so với duyệt theo chiều sâu – xem thêm trong slides bài giảng)

```
/* Khai bao Queue */

...

/* Duyệt đồ thị theo chiều rộng */
void breath_first_search(Graph* G) {
  Queue L;
  int mark[MAX_VERTEXES];
  make_null_queue(&L);

  /* Khởi tạo mark, chưa đỉnh nào được xét */
  int j;
  for (j = 1; j <= G->n; j++)
    mark[j] = 0;

  /* Đưa 1 vào frontier */
  push(&frontier, 1);

  /* Duyệt 1 */
  printf("Duyet 1\n");
  mark[1] = 1;

  /* Vòng lặp chính dùng để duyệt */
  while (!empty(&L)) {
    /* Lấy phần tử đầu tiên trong L ra */
    int x = top(&L); pop(&L);
    /* Lấy các đỉnh kề của nó */
    List list = neighbors(G, x);
    /* Xét các đỉnh kề của nó */
    for (j = 1; j <= list.size; j++) {
      int y = element_at(&list, j);
      if (marl[y] == 0) { // y chưa duyệt, duyệt nó
        printf("Duyet %d\n", y);
        mark[y] = 1; //Đánh dấu y đã duyệt
        push(&L, y); //Đưa vào hàng đợi để lát nữa xét
      }
    }
  }
}
```

Có thay đổi chút ít so với phần duyệt theo chiều sâu.

Cài đặt Queue có thể như thế này.

```
/* Khai bao Queue */
#define MAX_ELEMENTS 100
typedef struct {
  int data[MAX_ELEMENTS];
  int front, rear;
} Queue;

void make_null_queue(Queue* Q) {
  Q->front = 0;
  Q->rear = -1;
}

void push(Queue* Q, int x) {
  Q->rear++;
  Q->data[Q->rear] = x;
}

int top(Queue* Q) {
  return Q->data[Q->front];
}

void pop(Queue* Q) {
  Q->front++;
}

int empty(Queue* Q) {
  return Q->front > Q->rear;
}
```

### 2.1.4 Bài tập 2.2

Viết chương trình đọc đồ thị từ tập tin và duyệt đồ thị theo chiều rộng. Quan sát thứ tự các đỉnh được in ra màn hình, so sánh với bài tập 2.1 (thứ tự có giống nhau không, giải thích tại sao ?).

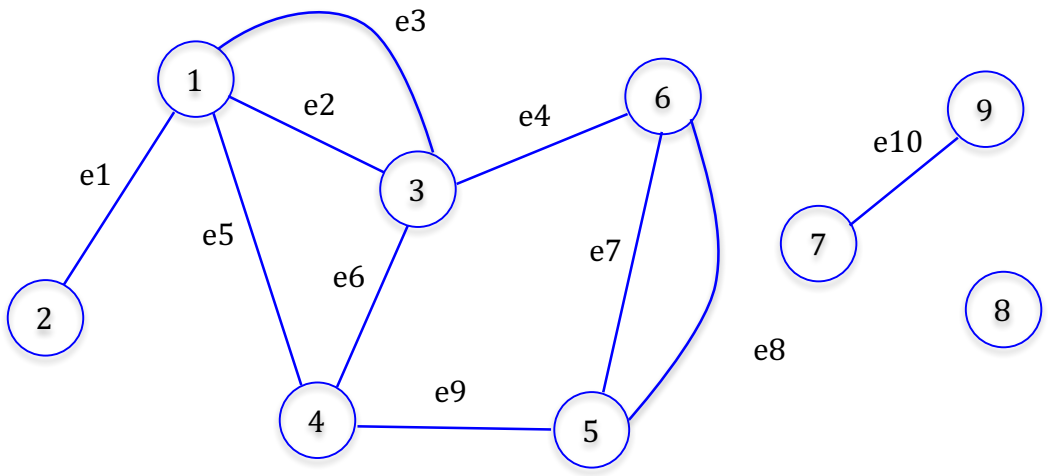
### 2.1.5 Bài tập 2.3

Cần phải chú ý rằng cả 2 phép duyệt này (rộng và sâu) chỉ có thể duyệt được một bộ phận liên thông (chứa đỉnh 1) của đồ thị. Để duyệt toàn bộ các đỉnh của đồ thị ta cần phải:

- Bổ sung thêm một tham số x vào hàm duyệt:
  - `depth_first_search(Graph* G, int x)` hoặc
  - `breath_first_search(Graph* G, int x)`
- Gọi lại hàm duyệt này nhiều lần, mỗi lần duyệt một phận liên thông

```
for (j = 1; j <= G->n; j++)
  /* Nếu đỉnh j chưa được duyệt, duyệt nó */
  if (mark[j] == 0)
    depth_first_search(G, j);
```

Làm lại bài tập 2.1 và 2.2 theo cách này. Sử dụng đồ thị bên dưới để kiểm tra.




### 2.1.6 Duyệt theo chiều sâu bằng phương pháp đệ quy

Ta có thể áp dụng kỹ thuật đệ quy để duyệt đồ thị theo chiều sâu mà không cần đến Stack L. Cách này cài đặt nhanh hơn phương pháp sử dụng stack và thường được sử dụng. Biến hỗ trợ mark phải được khai báo bên ngoài (toàn cục hoặc là tham số của hàm traversal)