```c
// Duyet cay do thi theo chieu "Sau" in ra parent (Stack)
#include <stdio.h>

int mark[100];
int parent[100];

// List
typedef struct {
    int data[100];
    int size;
} List;

void make_null_list(List* L) {
    L->size = 0;
}

void push_back(List* L, int x) {
    L->data[L->size] = x;
    ++L->size;
}

int element_at(List* L, int i) {
    return L->data[i - 1];
}

// Stack
typedef struct {
    int u;
    int v;
} Element_type;

Element_type make_Element_type(int u, int v) {
    Element_type ret;

    ret.u = u;
    ret.v = v;

    return ret;
}

typedef struct {
    Element_type data[100];
    int size;
```

```c
} Stack;

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

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

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

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

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

// Graph
typedef struct {
    int A[100][100];
    int n;
} Graph;

void init_graph(Graph* G, int n) {
    G->n = n;

    int i, j;

    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j) {
            G->A[i][j] = 0;
        }
    }
}

void add_egde(Graph* G, int x, int y) {
    G->A[x][y] = 1;
    G->A[y][x] = 1;
}

int adjacent(Graph* G, int x, int y) {
    return G->A[x][y];
}

List neighbors(Graph* G, int x) {
    int y;
    List list;

    make_null_list(&list);

    for (y = 1; y <= G->n; ++y) {
        if (adjacent(G, x, y)) {
            push_back(&list, y);
        }
    }

    return list;
}

void depth_first_search(Graph* G, int start) {
    Stack frontier;
    make_null_stack(&frontier);

    push(&frontier, make_Element_type(start, 0));

    mark[start] = 1;
    parent[start] = 0;

    while (!empty(&frontier)) {
        Element_type x = top(&frontier);
        pop(&frontier);

        mark[x.u] = 1;

        List list = neighbors(G, x.u);

        int j;
        for (j = 1; j <= list.size; ++j) {
            int y = element_at(&list, j);
```

```c
                if (!mark[y]) {
                    push(&frontier, make_Element_type(y, x.u));
                    parent[y] = x.u;
                }
            }
        }
}

int main() {
    //freopen("dt.txt", "r", stdin);

    Graph G;
    int n, m, i, x, y;

    scanf("%d%d", &n, &m);

    init_graph(&G, n);

    for (i = 1; i <= m; ++i) {
        scanf("%d%d", &x, &y);

        add_egde(&G, x, y);
    }

    for (i = 1; i <= n; ++i) {
        mark[i] = 0;
        parent[i] = 0;
    }

    for (i = 1; i <= n; ++i) {
        if (!mark[i]) {
            depth_first_search(&G, i);
        }
    }

    for (i = 1; i <= n; ++i) {
        printf("%d %d\n", i, parent[i]);

    }

    return 0;
}
```