```c
//khoi bao cau truc do thi
typedef struct {
    int A[MAX_VERTICES][MAX_VERTICES];
    int n,m;
}Graph;
void make_null_list(List* L) {
    L->size = 0;
}
/* Them mot phan tu vao cuoi danh sach */
void push_back(List* L, ElementType x) {
    L->data[L->size] = x;
    L->size++;
}
/* Lay phan tu tai vi tri i, phan tu bat dau o vi tri 1 */
ElementType element_at(List* L, int i) {
    return L->data[i-1];
}
/* Tra ve so phan tu cua danh sach */
int count_list(List* L) {
    return L->size;
}
/* phan do thi */
void init_graph(Graph *G,int n){
    int i,j;
    G->n = n;
    for(i=1;i<=n;i++)
        for(j=1;j<=n;j++)
            G->A[i][j]=0;
}
void add_edge(Graph *G, int x,int y){
    G->A[x][y]=1;

}
```

```c
int adjacent(Graph *G, int x, int y){
    return G->A[x][y] != 0;
}
int degree(Graph *G,int x){
    int y,deg=0;
    for(y=1; y<= G->n; y++)
        deg+= G->A[x][y];
    return deg;
}
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 copy_list(List *S1, List *S2){
    int i, x;
    make_null_list(S1);
    for(i=1;i<=S2->size;i++){
        x=element_at(S2,i);
        push_back(S1,x);
    }
}
int k=0;
List S1, S2;
```

[*] rank.c

```c
int k=0;
List S1, S2;
void ranking(Graph *G){
    int x, u;
    for(u = 1; u <= G->n; u++){
        d[u] = 0;
    }
    for(x = 1; x <= G->n; x++)
        for(u = 1; u <= G->n; u++)
        if(G->A[x][u] != 0)
            d[u]++;
    // List S1, S2;
    make_null_list(&S1);
    for(u = 1; u <= G->n; u++)
        if(d[u] == 0)
            push_back(&S1, u);
    // int k = 1, i;
    int i;
    while(S1.size > 0){
        make_null_list(&S2);
        for(i = 1; i <= S1.size; i++){
            int u = element_at(&S1, i);
            rank[u] = k;
            int v;
            for (v = 1; v <= G->n; v++)
                if(G->A[u][v] != 0){
                    d[v]--;
                if(d[v] == 0)
                    push_back(&S2, v);
                }
        }
        copy_list(&S1, &S2);
        k++;
    }
}
int main (){
    // freopen("dt.txt", "r", stdin);
    Graph G;
    int n, m, u, v, e;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
    for (e = 0; e < m; e++) {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }
    ranking(&G);
    for(u=1;u<=n;u++)
        printf("%d \n",rank[u]);
        return 0;
}
```