

```
// Thuyen Truong Haddock  
// Kiem tra chu trinh cua do thi co huong  
#include <stdio.h>
```

```
#define white 0  
#define black 1  
#define gray 2
```

```
int color[100];  
int cycle;
```

```
// 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];  
}
```

```
// 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 dfs(Graph* G, int x) {
    color[x] = gray;

    List list = neighbors(G, x);

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

        if (color[y] == gray) {
            cycle = 1;
            return;
        }
    }
}

```

```

        if (color[y] == white) {
            dfs(G, y);
        }
    }

    color[x] = black;
}

int contains_cycle(Graph* G) {
    int i;
    for (i = 1; i <= G->n; ++i) {
        color[i] = white;
    }

    cycle = 0;

    for (i = 1; i <= G->n; ++i) {
        if (color[i] == white) {
            dfs(G, i);
        }
    }

    return cycle;
}

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);
    }

    if (!contains_cycle(&G)) {
        printf("YES");
    }
}

```

```
    } else {  
        printf("NO");  
    }  
  
    return 0;  
}
```