

## Разбор задачи «Дороги-2»

Так как при «секретной» нумерации получается, что дороги идут в направлении возрастания номеров, то «секретная» нумерация — одна из возможных топологических сортировок графа дорог. По условию гарантируется существование хотя бы одной топологической сортировки. Если мы рассмотрим любую вершину, то она обязательно должна идти после всех вершин, из которых она достижима и перед всеми, которые из неё достижимы. Если же позиций для неё несколько, то мы можем ставить её на любую из них.

Если мы посчитаем два числа  $B$  — количество вершин перед ней и  $A$  — после неё, то всего позиций для неё будет  $N - B - A + 1$ :  $B + 1, B + 2, \dots, N - A$ .

Решение будет следующим:

1. Создаём представление прямого и обратного графа списком рёбер из каждой вершины. Здесь `head[num]` - указатель на первое ребро из данной вершины, а `next[num]` - на следующее за ним.  
`inl[num]` - количество уже назначенных рёбер.

```
void insert(int num, int f, int s) {
    next[num][inl[num]] = head[num][f];
    head[num][f] = inl[num];
    end[num][inl[num]] = s;
    inl[num]++;
}
```

...

```
for( i = 0; i < m; i++ ) {
    scanf("%d%d", &a, &b);
    a--, b--;
    insert(0, a, b);
    insert(1, b, a);
}
```

2. Считаем количества вершин достижимых из нашей вершины и тех, из которых она достижима. Последние вершины - это те, которые достижимы из нашей в обратном графе. Делаем это поиском в глубину, возвращающим число вершин, в которых он побывал.

```
int dfs(int v, int num) {
    int r = 1, e;
    mark[num][v] = 1;
```

```

        for( e = head[num][v]; e != -1; e = next[num][e] )
            if( !mark[num][end[num][e]] )
                r += dfs(end[num][e], num);

    return r;
}

...

a = dfs(0, 0)-1;
b = dfs(0, 1)-1;

```

3. Выводим ответ:

```

printf("%d\n", n-a-b);
for( i = 0; i < n-a-b; i++ )
    printf("%d ", b+i+1);

```

Суммарно решение будет выглядеть так:

```

#include <stdio.h>
#include <memory.h>

#define maxn 100000

int n, m;
int head[2][maxn], next[2][maxn], end[2][maxn], inl[2];
int mark[2][maxn];

void insert(int num, int f, int s) {
    next[num][inl[num]] = head[num][f];
    head[num][f] = inl[num];
    end[num][inl[num]] = s;
    inl[num]++;
}

int dfs(int v, int num) {
    int r = 1, e;
    mark[num][v] = 1;

    for( e = head[num][v]; e != -1; e = next[num][e] )
        if( !mark[num][end[num][e]] )
            r += dfs(end[num][e], num);
}

```

```

    return r;
}

int main(void)
{
    int i, a, b;

    freopen("roads2.in", "r", stdin);
    freopen("roads2.out", "w", stdout);

    memset(head, -1, sizeof(head));

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

    for( i = 0; i < m; i++ ) {
        scanf("%d%d", &a, &b);
        a--, b--;
        insert(0, a, b);
        insert(1, b, a);
    }

    a = dfs(0, 0)-1;
    b = dfs(0, 1)-1;

    printf("%d\n", n-a-b);
    for( i = 0; i < n-a-b; i++ )
        printf("%d ", b+i+1);

    return 0;
}

```