Breadth-first Search

Depth-first Search เป็นการท่องกราฟแบบ recursive ซึ่งจะมีลักษณะการทำงานแบบ stack (Last In, First Out; LIFO) กล่าวคือ จะพิจารณาโหนดล่าสุดที่เพิ่งเพิ่มเข้ามาใหม่ทันที

Breadth-first Search จะมีลักษณะการทำงานแบบ queue (First In, First Out) คือจะต้องพิจารณาโหนดรอบตัวทุกโหนดที่ตัวตามลำดับเท่านั้น ห้ามพิจารณาโหนดอื่นที่ลึกไปกว่านั้นก่อน

หลักการทำงานมีดังนี้

- กำหนดคิว Q ขึ้นมา (แนะนำให้ใช้ `std::queue` ของ STL)
- เพิ่มโหนดเริ่มต้นเข้าไปใน Q
- while ยังมีโหนดอยู่ใน Q:
  - นำโหนดแรกออกมาจากคิว ตั้งชื่อว่า u
  - ถ้าเคยปรินท์โหนด u แล้ว ให้ข้ามไปพิจารณาโหนดถัดไปในคิวทันที
  - ถ้ายังไม่เคย ให้ปรินท์ u แล้วจดว่าเคยปรินท์ u ไปแล้ว
  - พิจารณาโหนดที่อยู่ติดกับ u แต่ละโหนด ถ้ายังไม่เคยปรินท์ ให้เพิ่มเข้าคิว (พิจารณาโหนดที่มีหมายเลขน้อยสุดก่อน)

ภาพตัวอย่างอยู่หน้าสุดท้าย

จงเขียนโปรแกรมหาลำดับโหนดที่ได้จากการทำ BFS เมื่อเริ่มต้นจากจุดเริ่มต้นที่กำหนดให้

Input

บรรทัดแรก ประกอบด้วยจำนวนเต็ม n และ m แทนจำนวนโหนด และจำนวนเส้นเชื่อมในกราฟ ($1 \leq n \leq 10^5, 0 \leq m \leq 10^5$)

บรรทัดที่ $1+i$ ($1 \leq i \leq m$) ประกอบด้วยจำนวนเต็ม u_i และ v_i แสดงว่า มีเส้นเชื่อมระหว่างโหนดที่ u_i และโหนดที่ v_i ($1 \leq u_i, v_i \leq n$)

บรรทัดที่ $m+2$ ประกอบด้วยจำนวนเต็ม s แทนจุดเริ่มต้นของ BFS

รับประกันว่ากราฟที่กำหนดให้จะเป็นกราฟอย่างง่าย (Simple Graph)

Output

ในหนึ่งบรรทัด ให้ปรินท์ตัวเลขลำดับโหนดที่ได้จากการ BFS

ระหว่างที่ทำ BFS เมื่อพิจารณาโหนดที่อยู่ติดกันมากกว่า 1 โหนด ให้พิจารณาโหนดที่มีหมายเลขน้อยที่สุดก่อน

Example #1

Input

```
6 6
6 1
6 2
1 3
2 3
2 4
3 5
6
```

Output

```
6 1 2 3 4 5
```

Example #2

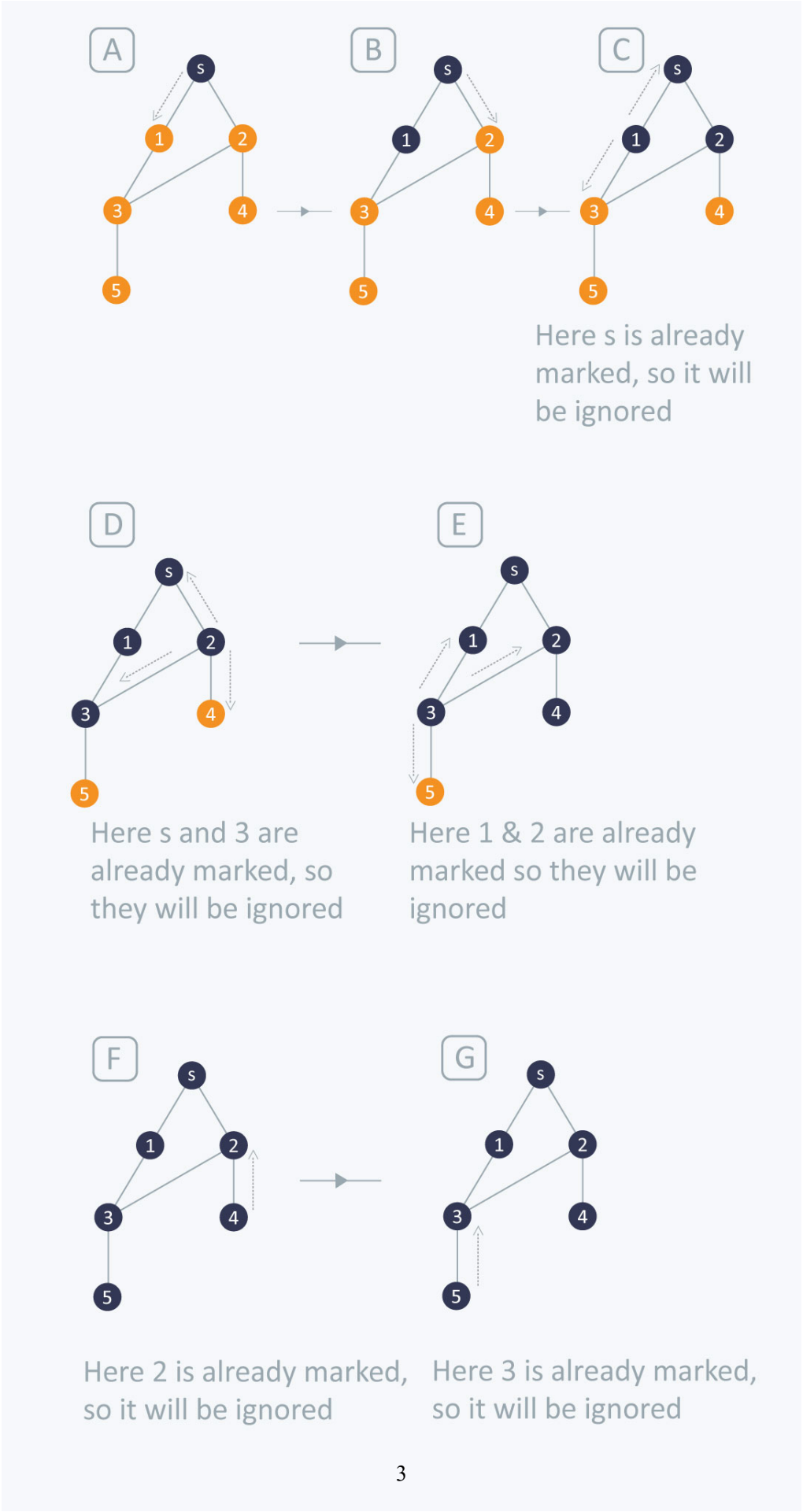
Input

```
5 6
1 2
1 3
1 4
1 5
2 4
4 5
2
```

Output

```
2 1 4 3 5
```

Note


A
B
C
Here s is already marked, so it will be ignored
D
E
Here s and 3 are already marked, so they will be ignored
Here 1 & 2 are already marked so they will be ignored
F
G
Here 2 is already marked, so it will be ignored
Here 3 is already marked, so it will be ignored