edgeSet.insert(((long long)u << 32) | v);

**🔎 Explanation**

We want to store an **undirected edge** (u, v) in a hash set, so that we can quickly check if it exists.

1. First, I normalize the edge so that u < v (swap if needed).
   * This ensures (u, v) and (v, u) are treated the same.
2. Then I combine them into a **single 64-bit integer key**:
   * (long long)u << 32 shifts u to the high 32 bits.
   * | v puts v into the low 32 bits.
   * So (u, v) → key = u \* 2^32 + v.

Example:

* Edge (2, 5) → key = (2 << 32) | 5.
* Edge (5, 2) → normalize → (2, 5) → same key.

1. This long long key goes into an unordered\_set<long long>, giving **O(1)** edge lookup.

✅ Why it works:

* u, v <= n where n ≤ 10^5 in constraints, so both fit safely in 32 bits.
* Using (u << 32) | v avoids collisions.

⚡ Basically, it’s a **custom perfect hash function** for storing pairs in a single number.

Do you want me to also show you an **alternative version using unordered\_set<pair<int,int>> with a custom hash function struct** (instead of bit tricks)?

* Insertion with Lazy deletion