Page Replacement Algorithms:

1. **Optimal Policy:**
   - Selects the page for replacement that will not be used for the longest time in the future.
   - Results in the fewest number of page faults.
   - Impossible to implement in real-world scenarios since future memory references are unknown.
   - Used as a benchmark for evaluating other algorithms.
2. **Least Recently Used (LRU) Policy:**
   - Replaces the page that has not been referenced for the longest time.
   - Works well due to the principle of locality.
   - Hard to implement efficiently due to tracking each memory reference.
   - Can be implemented using a stack or a counter mechanism.
3. **First-In-First-Out (FIFO) Policy:**
   - Uses a circular queue to replace the oldest page in memory.
   - Simple to implement but may result in excessive page faults.
   - Doesn't account for how frequently or recently a page has been used.
   - Can lead to the "Belady's Anomaly," where increasing frames increases faults.

Algorithms:

FIFO (First In First Out) Algorithm:

1. Accept frame size, number of pages, and the page sequence from the user.
2. Create a circular linked list with size equal to the frame size.
3. Initialize each node’s data to `-99`.
4. Insert page if it is not already present:
   - If there is an empty slot, place it in an available node.
   - If all slots are occupied, overwrite the oldest page.
5. Stop when all pages are processed.

LRU (Least Recently Used) Algorithm:

1. Start
2. Declare the size of the frame.

3. Get the number of pages to be inserted.
4. Get the sequence of pages.
5. Maintain a counter and a stack to track usage.
6. Replace the least recently used page when needed.
7. Display the page fault count.
8. Stop

Optimal Algorithm:

1. Start
2. Declare required variables and initialize them.
3. Get the frame size and reference string from the user.
4. When inserting a new element, find the page that will be used the farthest in the future and replace it.
5. Count the number of page faults and display the result.
6. Stop

Key Takeaways:

- **Optimal Algorithm** gives the best performance but is impractical.
- **LRU** is a good approximation but requires tracking usage history.
- **FIFO** is simple but can perform poorly in some cases.
- **Choosing the right algorithm depends on the workload and system constraints.**