REPLACEMENT ALGORITHMS EXAMPLES:
OPT, FIFO (QUEUE, CIRCULAR), LRU
Same Page Input Sequence For All Three
REPLACEMENT ALGORITHMS-SECOND CHANCE, AGING
RELATION BETWEEN THE FIFO QUEUE AND THE FIFO CIRCULAR DATA STRUCTURES
On Each Replacement

1-2 Handed Clock Algorithm
Based On Second Chance-FIFO
LRU MATRIX ALGORITHM

The LRU Matrix has a distinct row and column for each page frame. Page frame $i$ is assigned row $i$ and column $i$. When page $i$ is referenced all entries in row $i$ are made 1 and then all entries in column $i$ are made 0. At any time the page(s) whose row has the most 0s is the least recently used.

In the final matrix there is exactly one 0 in row 2. There are exactly two 0s in row 1, because the last two pages referenced are 1 and 2. There are exactly three 0s in row 4 because the last three pages referenced were 1, 2 and 4. We can ignore the fact that another use of page 2 occurred earlier. There are exactly four 0s in row 3 because the last four pages referenced were 1, 2, 3, and 4.

---

**GENERAL**

**REPLACEMENT ALGORITHMS LRU MATRIX IMPLEMENTATION**
Working Set of Process P = The set of P’s pages in current use

Formally \( w(k,t) = \) set of k pages most recently referenced at time t

An approximation: \( w(I,t) = \) set of pages referenced within the last interval of time I. I is determined by simulations or other experimental methods.

A working set replacement algorithm replaces any page in MM which is not in the working set.

To obtain a useful working set algorithm: keep the last virtual time at which a process has been used. If it is within an interval, I, before the current virtual time it is in the working set and should not be replaced. If it was not used within that interval I it may be replaced.

**WORKING SET**
Given that MM1 has fewer page frames than MM2:

If whenever ALG is used to allocate the same page input sequence to both MM1 and MM2, every page in MM1 is always also in MM2, ALG is a STACK REPLACEMENT ALGORITHM.

So for FIFO it is necessary to keep tables having 1, 2, ..., n rows to record the result of a simulation of all MMs having any number of pages from 1 to n. So $1 + 2 + ... + n = O(n^2)$ rows are required.

For a Stack Algorithm only n rows are required to record the result of a simulation of all MMs having any number of pages from 1 to n.

STACK ALGORITHM
FIFO IS NOT ONE
To represent the result of LRU run on a given page input sequence:

1) put new page arrivals in row 1 and push all others down if that page is not already in any row of the table.

2) If the incoming page, \( P \), is already in a row of the table put \( P \) in row 1, and now remove \( P \)'s other occurrence from the row it was in, and finally push down all other entries filling in the blank left.

This is the same procedure one would use if there are a fixed number of pages, (therefore of rows,) except when a new page came in and the MM was already filled the page at the bottom of table would be removed.

At any stage in the reception of the input page sequence the page that would be removed if the next page input required a replacement is the dispensable one.

The star (*) in this representation has significance:

\( x \) in row \( k \) signifies that \( x \) is dispensable in a \( k \) page memory and, if the next row with a starred (*) entry is in row \( k+i \), then \( x \) is dispensable also in row \( k+1 \) to row \( k+i+1 \). For LRU all entries are starred, so row \( n \) in a table with \( n \) rows (MM with \( n \) pages) is the dispensable page. This is not the case for OPT considered next.

### LRU Stack Replacement Algorithm

**Table:**

<table>
<thead>
<tr>
<th>Page Frames</th>
<th>Page Input Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[a^* b^* c^* a^* d^* e^* c^* e^*]</td>
</tr>
<tr>
<td>2</td>
<td>[-a b^* c^* a^* d^* e^* c^*]</td>
</tr>
<tr>
<td>3</td>
<td>[- - a^* b^* c^* a^* d^* d^*]</td>
</tr>
<tr>
<td>4</td>
<td>[- - - - b^* c^* a^* a^*]</td>
</tr>
<tr>
<td>5</td>
<td>[- - - - - b^* b^* b^*]</td>
</tr>
<tr>
<td>6</td>
<td>[- - - - - - - - - - -]</td>
</tr>
</tbody>
</table>

**Distance String:** \[\infty \infty \infty 3 \infty \infty 4 \infty 2\]
The algorithm for filling in the table for OPT is not as simple as with LRU.

Unlike when using LRU, using OPT all pages in the table are starred. The dispensable pages in each column are determined by the order in which those pages in MM will again arrive in the future.

<table>
<thead>
<tr>
<th>Page Frames</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>a</th>
<th>d</th>
<th>e</th>
<th>c</th>
<th>e</th>
<th>d</th>
<th>b</th>
<th>a</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a*</td>
<td>b*</td>
<td>c*</td>
<td>a*</td>
<td>d*</td>
<td>e*</td>
<td>c*</td>
<td>e*</td>
<td>d*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>e</td>
<td>c*</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>–</td>
<td>–</td>
<td>b*</td>
<td>b</td>
<td>b*</td>
<td>d*</td>
<td>d</td>
<td>d</td>
<td>c*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>a*</td>
<td>b*</td>
<td>b</td>
<td>b</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>a*</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[
C_\infty = 5 \quad C_3 = 1 \quad C_2 = 3
\]

\(C_k\) is the number of hits in an MM with \(k\) page frames, but not in an MM with fewer page frames. With \(k\) page frames there are \(C_1 + \ldots + C_k\) hits.

**OPT - A STACK REPLACEMENT ALGORITHM**