**Exp No.** 11 **Name:** PONNURI ANIRUDDHA

**Date**: 18/10/2022 **Reg No:** RA2112704010015

**LRU and LFU Page Replacement Algorithm**

**Aim: --**

To implement LRU and LFU Page Replacement Algorithm

**Procedure: --**

**LRU (Least Recently Used) Page Replacement:**

The LRU stands for the Least Recently Used. It keeps track of page usage in the memory over a short period of time. It works on the concept that pages that have been highly used in the past are likely to be significantly used again in the future. It removes the page that has not been utilized in the memory for the longest time. LRU is the most widely used algorithm because it provides fewer page faults than the other methods.

**LFU (Least Frequently Used) Page Replacement:**

In this, it is using the concept of paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when the new page comes in. Whenever a new page is referred to and is not present in memory, the page fault occurs and the Operating System replaces one of the existing pages with a newly needed page. LFU is one such page replacement policy in which the least frequently used pages are replaced. If the frequency of pages is the same, then the page that has arrived first is replaced first.

**Code (LRU):** --

*capacity = int(input(‘Enter number of page frames :-- ‘)*

*processList = [ 7, 0, 1, 2, 0, 3, 0,4, 2, 3, 0, 3, 2]*

*# List of current pages in Main Memory*

*s = []*

*pageFaults = 0*

*for i in processList:*

*# If i is not present in currentPages list*

*if i not in s:*

*# Check if the list can hold equal pages*

*if(len(s) == capacity):*

*s.remove(s[0])*

*s.append(i)*

*else:*

*s.append(i)*

*# Increment Page faults*

*pageFaults +=1*

*# If page is already there in*

*# currentPages i.e in Main*

*else:*

*# Remove previous index of current page*

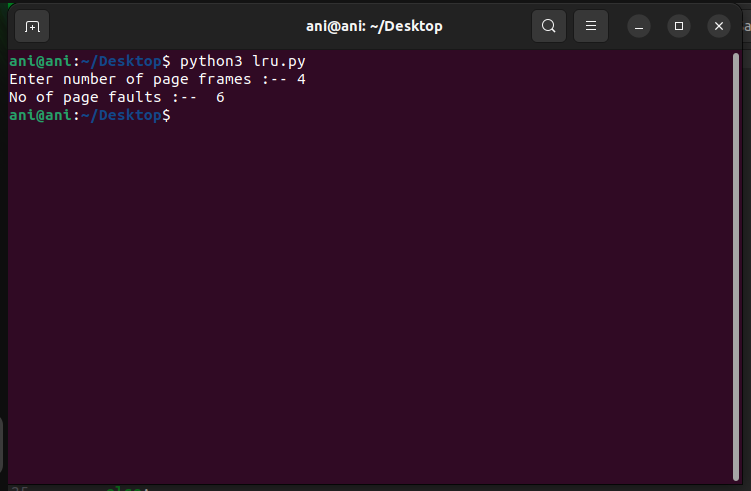
*s.remove(i)*

*# Now append it, at last index*

*s.append(i)*

*print("No of page faults :-- ",pageFaults)*

**OUTPUT (LRU): --**



**Code (LFU): --**

*#include <bits/stdc++.h>*

*using namespace std;*

*int pageFaults(int n, int c, int pages[])*

*{*

*// Initialise count to 0*

*int count = 0;*

*vector<int> v;*

*// To store frequency of pages*

*unordered\_map<int, int> mp;*

*int i;*

*for (i = 0; i <= n - 1; i++) {*

*// Find if element is present in memory or not*

*auto it = find(v.begin(), v.end(), pages[i]);*

*// If element is not present*

*if (it == v.end()) {*

*// If memory is full*

*if (v.size() == c) {*

*// Decrease the frequency*

*mp[v[0]]--;*

*// Remove the first element as*

*// It is least frequently used*

*v.erase(v.begin());*

*}*

*// Add the element at the end of memory*

*v.push\_back(pages[i]);*

*// Increase its frequency*

*mp[pages[i]]++;*

*// Increment the count*

*count++;*

*}*

*else {*

*// If element is present*

*// Remove the element*

*// And add it at the end*

*// Increase its frequency*

*mp[pages[i]]++;*

*v.erase(it);*

*v.push\_back(pages[i]);*

*}*

*// Compare frequency with other pages*

*// starting from the 2nd last page*

*int k = v.size() - 2;*

*// Sort the pages based on their frequency*

*// And time at which they arrive*

*// if frequency is same*

*// then, the page arriving first must be placed first*

*while (mp[v[k]] > mp[v[k + 1]] && k > -1) {*

*swap(v[k + 1], v[k]);*

*k--;*

*}*

*}*

*return count;*

*}*

*int main()*

*{*

*int pages[] = { 1, 2, 3, 4, 2, 1, 5 };*

*int n = 7, c = 3;*

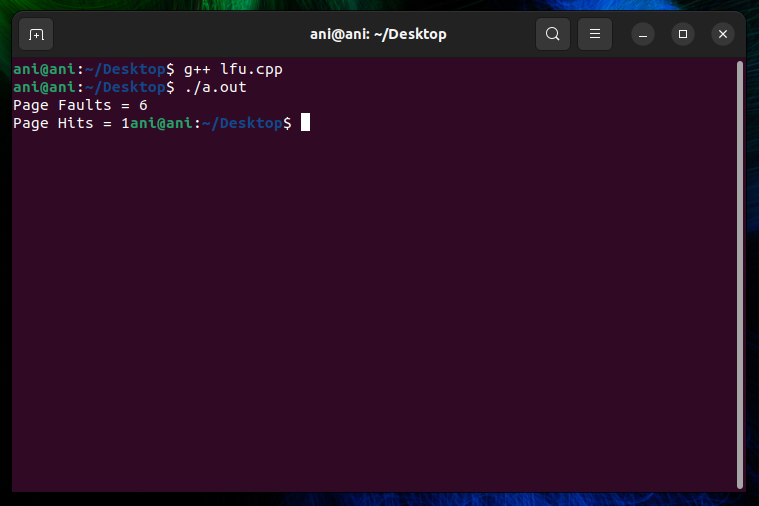
*cout << "Page Faults = " << pageFaults(n, c, pages)<< endl;*

*cout << "Page Hits = " << n - pageFaults(n, c, pages);*

*return 0;*

*}*

**OUTPUT (LFU): --**



**RESULT: --**

Successfully implemented LRU and LFU page replacement algorithm in python