Lab 09

Started: Oct 23 at 12:48pm

Quiz Instructions

Question 1 **100 pts**

Task Description

In this lab, you will implement the Least Recently Used (LRU) page replacement algorithm using a page table based on a stack implemented as a double-linked list. The program will be fed with a sequence of page references and its output will be the corresponding page fault rate for a given maximum capacity of the table.

Input

The input will consist of the following components:

- list maximum capacity of the table in the first line, and
- a sequence of page references in the second.

The list maximum capacity determines the number of frames in the reference table; i.e., the number of page references that can be kept at any given moment.

The sequence of page references is a list of numbers that correspond to referenced pages. Each page number is a non-negative integer.

Example Input

```
3
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```

Output

For every page reference request in the list of page references the following components must be displayed:

- the content of the reference table before the request
  - the table is displayed in the order of the stack

- the current page reference request
- an indicator whether there was a hit or a fault
  - hit - the reference was already in the table
  - fault - the reference was not in the table and needed to be inserted, and finally
- the reference table after the request

Once the entire list of references has been processed, the program should display the number of page faults.

Example Output

(corresponding to the example input)

```
*7
*0 7
*1 0 7
*2 1 0
>0 2 1
*3 0 2
>0 3 2
*4 0 3
*2 4 0
*3 2 4
*0 3 2
>3 0 2
>2 3 0
*1 2 3
>2 1 3
*0 2 1
>1 0 2
*7 1 0
>0 7 1
>1 0 7

Number of faults: 12
```

Notes: *The top of the stack is on the left, and the bottom is on the right. The * indicates a page fault, and > indicates referencing a page that was already in the table and that was moved to the top.*

Implementation

You are required to implement the table as a quasi-stack made up of double-linked nodes (as discussed in the lecture). This means that each node has a pointers to its next and its previous neighbors, as well the corresponding number of the referenced page.

Initially, the table does not have any pages (it is an empty list). As pages are being

referenced, new nodes representing the referenced pages are being pushed on the stack until the list reaches its maximum capacity (given as an input parameter). When the list is at its maximum capacity, then there are two cases to consider:

1. When the referenced page is not in the table, it is placed at the top of the stack and the node from the bottom of the stack (representing the victim page) is removed.
2. When the referenced page is already in the table, the node corresponding to that page must be moved to the top of the stack.

Since removal of an element other than the top of a stack is not a standard stack behavior, we refer to the structure used in this task as a quasi-stack.

Submission:

You will need to submit the following:

- `replace.c` and `replace.h` that contain your code
- typescript of the sessions showing multiple runs with different page reference sequences and table sizes.

Upload Choose a File

Saving... Submit Quiz