

**Question 1 (10 Marks)**

The program to be implemented reads a character from a file. The program is required to count the frequency of the input character in three text files. In order to perform the above, two versions of the program need to be implemented, one a sequential version and the other a concurrent version. The sequential version implements a function to determine the frequency of occurrence of the input character in a text file. The main function consolidates frequency of the input character in the three text files. The concurrent version of the program spawns three threads, each one to determine the frequency of occurrence of the input character in one of the three files. The main thread computes the consolidated frequency of the input character based on the individual thread’s output.

Perform the following using file management system calls:

1. Design and implement the application using sequential approach with functions
2. Design and implement the application using multithreaded approach
3. Compare the execution time of the above two versions of the program and analyse their performance

**Question 2 (10 Marks)**

Consider a computer machine with a memory system containing three physical page frames and eight virtual pages. Suppose the reference string of physical page accesses is 0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2 1 2.

1. Determine the number of page faults that occur among the following page replacement algorithms
   * FIFO :

It associates with each page the time when that page was brought into memory.

When a page must be replaced, the oldest page is chosen.

Note: It is not necessary to record the time when a page is brought in.

Given;

Number of physical page frames = 3

Number of virtual pages = 8

Reference string of physical page access is = 0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2 1 2

\*= Page hit

P = Page fault

|  |
| --- |
| 1 |
| 4 |
| 3 |

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2

|  |
| --- |
| 0 |
|  |
|  |

|  |
| --- |
| 0 |
| 1 |
|  |

|  |
| --- |
| 0 |
| 1 |
| 2 |

|  |
| --- |
| 3 |
| 1 |
| 2 |

|  |
| --- |
| 3 |
| 0 |
| 2 |

|  |
| --- |
| 3 |
| 0 |
| 4 |

|  |
| --- |
| 5 |
| 0 |
| 4 |

|  |
| --- |
| 5 |
| 2 |
| 4 |

|  |
| --- |
| 5 |
| 2 |
| 3 |

|  |
| --- |
| 1 |
| 2 |
| 3 |

|  |
| --- |
| 1 |
| 4 |
| 3 |

|  |
| --- |
| 1 |
| 4 |
| 2 |

|  |
| --- |
| 3 |
| 1 |
| 2 |

|  |
| --- |
| 3 |
| 1 |
| 2 |

P P P P \* \* P P P P P P P P \*

|  |
| --- |
| 6 |
| 3 |
| 2 |

|  |
| --- |
| 6 |
| 4 |
| 2 |

|  |
| --- |
| 6 |
| 4 |
| 2 |

|  |
| --- |
| 6 |
| 3 |
| 1 |

|  |
| --- |
| 2 |
| 3 |
| 1 |

6 3 2 1 2

P P \* P P

* Initially all slots are empty, so when 0, 1, 2 came they are allocated to the empty slots.

So, there is page fault.

* Then 3 comes, it is not available in the memory so it replaces the oldest page slot i.e 0

So, there is page fault.

* Then 2 comes, it is already in memory. So, there is not page fault.
* Then 3 comes, it is already in the memory. So, there is not page fault.
* Then 0 comes, it is not in the memory so it replaces oldest page slot i.e 1. So, there is page fault.
* Then 4 comes, it is not in the memory so it replaces oldest page slot i.e 2. So, there is page fault.
* Then 5 comes, it is not in the memory so it replaces oldest page slot i.e 3. So, there is page fault.
* Then 2 comes, it is not in the memory so it replaces oldest page slot i.e 0. So, there is page fault.
* Then 3 comes, it is not in the memory so it replaces oldest page slot i.e 4. So, there is page fault.
* Then 1 comes, it is not in the memory so it replaces oldest page slot i.e 5. So, there is page fault.
* Then 4 comes, it is not in the memory so it replaces oldest page slot i.e 2. So, there is page fault.
* Then 3 comes, it is already in the memory. So, there is not page fault.
* Then 2 comes, it is not in the memory so it replaces oldest page slot i.e 3. So, there is page fault.
* Then 6 comes, it is not in the memory so it replaces oldest page slot i.e 1. So, there is page fault.
* Then 3 comes, it is not in the memory so it replaces oldest page slot i.e 4. So, there is page fault.
* Then 2 comes, it is already in the memory. So, there is not page fault.
* Then 1 comes, it is not in the memory so it replaces oldest page slot i.e 2. So, there is page fault.
* Then 2 comes, it is not in the memory so it replaces oldest page slot i.e 6. So, there is page fault.

Total number of page hit (\*) = 4

Hence, Total number of page faults = Total number of pages – Number of page hits

= 20 – 4 = 16

* + LRU :

It associates with each page the time of that page’s last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.

LRU can be done by two methods: (i) Counter algorithm

(ii) Stack algorithm

Here, I am using Stack algorithm.

|  |
| --- |
| 3 |
| 4 |
| 1 |

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2

|  |
| --- |
| 0 |
|  |
|  |

|  |
| --- |
| 1 |
| 0 |
|  |

|  |
| --- |
| 2 |
| 1 |
| 0 |

|  |
| --- |
| 3 |
| 2 |
| 1 |

|  |
| --- |
| 0 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 0 |
| 3 |

|  |
| --- |
| 5 |
| 4 |
| 0 |

|  |
| --- |
| 2 |
| 5 |
| 4 |

|  |
| --- |
| 3 |
| 2 |
| 5 |

|  |
| --- |
| 1 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 1 |
| 3 |

|  |
| --- |
| 2 |
| 3 |
| 4 |

|  |
| --- |
| 2 |
| 3 |
| 1 |

|  |
| --- |
| 3 |
| 2 |
| 1 |

P P P P \* \* P P P P P P P \* P

|  |
| --- |
| 6 |
| 2 |
| 3 |

|  |
| --- |
| 3 |
| 6 |
| 2 |

|  |
| --- |
| 2 |
| 3 |
| 6 |

|  |
| --- |
| 1 |
| 2 |
| 3 |

|  |
| --- |
| 2 |
| 1 |
| 3 |

6 3 2 1 2

P \* \* P \*

* Initially all slots are empty, so when 0, 1, 2 came they are allocated to the empty slots.

So, there is page fault.

* Then 3 comes, it is not already there so it pushes others down and take the top spot.

So, there is page fault.

* Then 2 comes, it is already available so it takes the top spot. So, there is not page fault.
* Then 3 comes, it is already available so it takes the top spot. So, there is not page fault.
* Then 0 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 4 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault
* Then 5 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 2 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 3 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 1 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 4 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 3 comes, it is already available so it takes the top spot. So, there is not page fault.
* Then 2 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 6 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 3 comes, it is already available so it takes the top spot. So, there is not page fault.
* Then 2 comes, it is already available so it takes the top spot. So, there is not page fault.
* Then 1 comes, it is not already there so it pushes others down and take the top spot. So, there is page fault.
* Then 2 comes, it is already available so it takes the top spot. So, there is not page fault.

Total number of page hit (\*) = 6

Hence, Total number of page faults = Total number of pages – Number of page hits

= 20 – 6 = 14

* Optimal :

It replaces the page that will not be used for the longest period of time.

|  |
| --- |
| 5 |
| 3 |
| 2 |

|  |
| --- |
| 5 |
| 3 |
| 2 |

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2

|  |
| --- |
| 0 |
|  |
|  |

|  |
| --- |
| 0 |
| 1 |
|  |

|  |
| --- |
| 0 |
| 1 |
| 2 |

|  |
| --- |
| 0 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 3 |
| 2 |

|  |
| --- |
| 5 |
| 3 |
| 2 |

|  |
| --- |
| 1 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 3 |
| 2 |

|  |
| --- |
| 0 |
| 3 |
| 2 |

|  |
| --- |
| 0 |
| 3 |
| 2 |

|  |
| --- |
| 0 |
| 3 |
| 2 |

|  |
| --- |
| 6 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 3 |
| 2 |

|  |
| --- |
| 4 |
| 3 |
| 2 |

|  |
| --- |
| 6 |
| 3 |
| 2 |

|  |
| --- |
| 6 |
| 3 |
| 2 |

P P P P \* \* \* P P \* \* P P \* \* P \* \*

1 2

|  |
| --- |
| 1 |
| 3 |
| 2 |

|  |
| --- |
| 1 |
| 3 |
| 2 |

P \*

* Initially all slots are empty, so when 0, 1, 2 came they are allocated to the empty slots. So, there is page fault.
* Then 3 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 1. So, there is page fault.
* Then 2 comes, it is already available. So, there is not page fault.
* Then 3 comes, it is already available. So, there is not page fault.
* Then 0 comes, it is already available. So, there is not page fault.
* Then 4 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 0. So, there is page fault.
* Then 5 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 4. So, there is page fault.
* Then 2 comes, it is already available. So, there is not page fault.
* Then 3 comes, it is already available. So, there is not page fault.
* Then 1 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 5. So, there is page fault.
* Then 4 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 1. So, there is page fault.
* Then 3 comes, it is already available. So, there is not page fault.
* Then 2 comes, it is already available. So, there is not page fault.
* Then 6 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 4. So, there is page fault.
* Then 3 comes, it is already available. So, there is not page fault.
* Then 2 comes, it is already available. So, there is not page fault.
* Then 1 comes, it is not available already so it replaces the page that will not be used for the longest period of time i.e 3. So, there is page fault.
* Then 2 comes, it is already available. So, there is not page fault.

Total number of page hit = 10

Hence, Total number of page faults = Total number of pages – Number of page hits

= 20 – 10 = 10

1. Draw the diagram of the probability density function of distance strings based on LRU algorithm.

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2 1 2

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2 1 2

0 1 2 3 2 3 0 4 5 2 3 1 4 3 2 6 3 2 1

0 1 1 1 2 3 0 4 5 2 3 1 4 3 2 6 3 3

0 0 0 1 2 3 0 4 5 2 2 1 4 4 4 6 6

1 2 3 0 4 5 5 5 1 1 1 4 4

1 1 1 0 0 0 0 5 5 5 5 5

0 0 0 0 0

String distance: x x x x 2 2 4 x x 5 5 6 5 3 4 x 3 3 5 2

Probability density function:

P(1) = 0

P(2) = 3/20 = 0.15

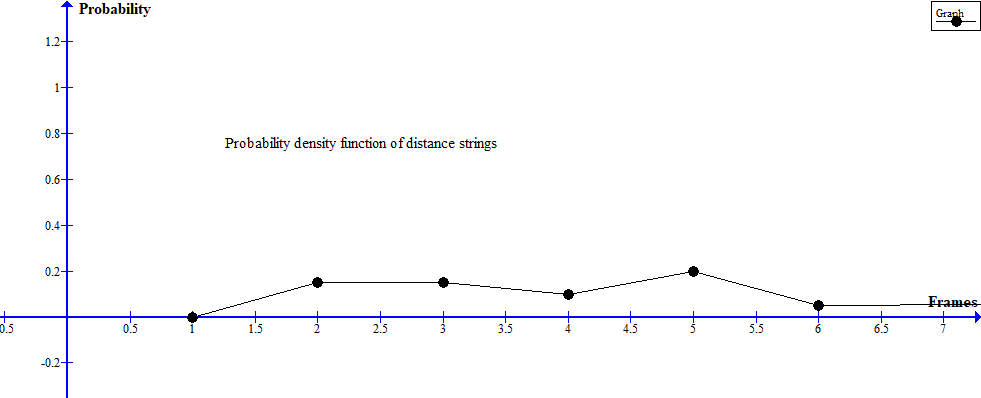
P(3) = 3/20 = 0.15

P(4) = 2/20 = 0.1

P(5) = 4/20 = 0.2

P(6) = 1/20 = 0.05

P(infinity) = 7/20 = 0.35



1. Recommend an optimal number of physical page frames appropriate for the given string of accesses:

Since, probability is highest at frame 5 the optimal number of physical page frames appropriate for the give string of accesses, according to me is 5.