# OS LAB: Page replacement Algorithm

Name: Disha Khater
Roll no. 64
12010067
CS-A

```
package OS_LABS;
//Page replacement algo with FIFO
import java.util.*;

public class page_replacement_algo {
    public static int FIFO(int pages[], int n, int capacity){
        HashSet<Integer> pageframes = new HashSet<>();
        Queue<Integer> queue = new LinkedList<>();
        int pagefaults = 0;
        for (int i=0;i<n;i++){
            if(pageframes.size()  < capacity){
                if(!pageframes.contains(pages[i])){
                    pageframes.add(pages[i]);
                    queue.add(pages[i]);
                    pagefaults++;
                }
            }
            else {
                if(!pageframes.contains(pages[i])){
                  int r = queue.peek();
                  queue.poll();
                  pageframes.remove(r);
                  pageframes.add(pages[i]);
                  queue.add(pages[i]);
                  pagefaults++;
                }
            }
        }
        return pagefaults;
    }

    public static int LRU(int pages[], int n, int capacity){
        HashSet<Integer> s = new HashSet<>();
        HashMap<Integer, Integer> indexes = new HashMap<>();
        int pagefaults = 0;
        for(int i=0;i<n;i++){
            if(s.size() < capacity){
                if(!s.contains(pages[i])){
                    s.add(pages[i]);
                    pagefaults++;
                }
                indexes.put(pages[i],i);
            }
            else {
                if(!s.contains(pages[i])){
                    int lru = Integer.MAX_VALUE, val=Integer.MIN_VALUE;
                    Iterator<Integer> itr = s.iterator();
```

```
                while (itr.hasNext()) {
                    int temp = itr.next();
                    if (indexes.get(temp) < lru) {
                        lru = indexes.get(temp);
                        val = temp;
                    }
                }
                s.remove(val);
                indexes.remove(val);
                s.add(pages[i]);
                pagefaults++;
            }
            indexes.put(pages[i], i);
        }
    }
    return pagefaults;
}

static boolean search(int key, int[] fr)
{
    for (int i = 0; i < fr.length; i++)
        if (fr[i] == key)
            return true;
    return false;
}

static int predict(int pg[], int[] fr, int pn, int index)
{
    int res = -1, farthest = index;
    for (int i = 0; i < fr.length; i++) {
        int j;
        for (j = index; j < pn; j++) {
            if (fr[i] == pg[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == pn)
            return i;
    }
    return (res == -1) ? 0 : res;
}

static void optimalPage(int pg[], int pn, int fn)
{
    int[] fr = new int[fn];
    int hit = 0;
    int index = 0;
    for (int i = 0; i < pn; i++) {

        if (search(pg[i], fr)) {
            hit++;
            continue;
        }

        if (index < fn)
            fr[index++] = pg[i];
```

```
            else {
                int j = predict(pg, fr, pn, i + 1);
                fr[j] = pg[i];
            }
        }
        System.out.println("No. of hits = " + hit);
        System.out.println("No. of misses = " + (pn - hit));
    }

    public static void main(String[] args) {
        int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
        int capacity = 4;
        System.out.println("FIFO page faults: "+FIFO(pages, pages.length,
capacity));
        System.out.println("LRU page faults: "+LRU(pages, pages.length,
capacity));
        int pg[]
                = { 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 };
        int pn = pg.length;
        int fn = 4;
        optimalPage(pg, pn, fn);
    }
}
```

OUTPUT:

```
FIFO page faults: 7
LRU page faults: 6
No. of hits = 7
No. of misses = 6

Process finished with exit code 0
```