**Implementation of First - Fit and Second Chance algorithms**

**AKZHOL SOVET,** BDA, Bachelor

**MIROLIM SAIDAKHMATOV,** BDA, Bachelor

A picture containing text

Description automatically generated

EXPO business center, block C.1.

Nur-Sultan, Kazakhstan,010000

**Supervisor:** Ruslan Omirgaliyev

**Contents:**

[**Chapter 1: Introduction** 3](#_Toc66521351)

[**1.1 First-Fit Memory Allocation:** 3](#_Toc66521352)

[**1.2 Clock Page Replacement Policy** 4](#_Toc66521353)

[**Chapter 2: Methodology** 5](#_Toc66521354)

[**Chapter 3: Discussion** 5](#_Toc66521355)

[**3.1 First-fit algorithm** 5](#_Toc66521356)

[**3.2 Second chance(clock) algorithm** 6](#_Toc66521357)

[**Chapter 4: Conclusion** 9](#_Toc66521358)

[**References** 10](#_Toc66521359)

# **Chapter 1: Introduction**

For both fixed and dynamic memory allocation schemes, the operating system must keep list of each memory location noting which are free and which are busy. Then as new jobs come into the system, the free partitions must be allocated. These partitions may be allocated by 4 ways:

**1. First-Fit Memory Allocation**

**2. Best-Fit Memory Allocation**

3. Worst-Fit Memory Allocation

4. Next-Fit Memory Allocation

## **First-Fit Memory Allocation**

This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, first job claims the first available memory with space more than or equal to it’s size. The operating system does not search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

**Advantages of First-Fit Memory Allocation:**

It is fast in processing. As the processor allocates the nearest available memory partition to the job, it is very fast in execution.

**Disadvantages of First-Fit Memory Allocation :**

It wastes a lot of memory. The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not. It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.

## **Clock Page Replacement Policy**

In the Second Chance page replacement policy, the candidate pages for removal are considered in a round robin matter, and a page that has been accessed between consecutive considerations will not be replaced. The page replaced is the one that, when considered in a round robin matter, has not been accessed since its last consideration.

It can be implemented by adding a “second chance” bit to each memory frame-every time the frame is considered (due to a reference made to the page inside it), this bit is set to 1, which gives the page a second chance, as when we consider the candidate page for replacement, we replace the first one with this bit set to 0 (while zeroing out bits of the other pages we see in the process). Thus, a page with the “second chance” bit set to 1 is never replaced during the first consideration and will only be replaced if all the other pages deserve a second chance too!

# **Chapter 2: Methodology**

## **2.1 First-fit algorithm Java code**

class FirstFit {  
  
 static void firstFit(int blockSize[], int m,  
 int processSize[], int n) {  
  
 int allocation[] = new int[n];  
 for (int i = 0; i < allocation.length; i++)  
 allocation[i] = -1;  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < m; j++) {  
 if (blockSize[j] >= processSize[i]) {  
 allocation[i] = j;  
 blockSize[j] -= processSize[i];  
  
 break;  
 }  
 }  
 }  
  
 System.*out*.println("\nProcess No.\tProcess Size\tBlock no.");  
 for (int i = 0; i < n; i++) {  
 System.*out*.print(" " + (i + 1) + "\t\t" +  
 processSize[i] + "\t\t");  
 if (allocation[i] != -1)  
 System.*out*.print(allocation[i] + 1);  
 else  
 System.*out*.print("Not Allocated");  
 System.*out*.println();  
 }  
 }  
}

public class main {  
  
 public static void main(String[] args)  
 {  
 int blockSize[] = {100, 500, 200, 300, 600};  
 int processSize[] = {212, 417, 112, 426};  
 int m = blockSize.length;  
 int n = processSize.length;  
  
 FirstFit.*firstFit*(blockSize, m, processSize, n);  
 }  
}

## **2.2 Clock(Second Chance) algorithm Java code**

import java.util.\*;  
import java.io.\*;  
  
class secondChance  
{  
 public static void main(String args[])throws IOException  
 {  
 String reference\_string = "";  
 int frames = 0;  
  
 //Test 1:  
 reference\_string = "0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4";  
 frames = 3;  
  
 //Output is 9  
 *printHitsAndFaults*(reference\_string,frames);  
  
 //Test 2:  
 reference\_string = "2 5 10 1 2 2 6 9 1 2 10 2 6 1 2 1 6 9 5 1";  
 frames = 4;  
  
 //Output is 11  
 *printHitsAndFaults*(reference\_string,frames);  
  
 }  
  
 //If page found, updates the second chance bit to true  
 static boolean findAndUpdate(int x,int arr[],  
 boolean second\_chance[],int frames)  
  
 {  
 int i;  
  
 for(i = 0; i < frames; i++)  
 {  
  
 if(arr[i] == x)  
 {  
 //Mark that the page deserves a second chance  
 second\_chance[i] = true;  
  
 //Return 'true', that is there was a hit  
 //and so there's no need to replace any page  
 return true;  
 }  
 }  
  
 //Return 'false' so that a page for replacement is selected  
 //as he reuested page doesn't exist in memory  
 return false;  
  
 }  
  
  
 //Updates the page in memory and returns the pointer  
 static int replaceAndUpdate(int x,int arr[],  
 boolean second\_chance[],int frames,int pointer)  
 {  
 while(true)  
 {  
  
 //We found the page to replace  
 if(!second\_chance[pointer])  
 {  
 //Replace with new page  
 arr[pointer] = x;  
  
 //Return updated pointer  
 return (pointer+1)%frames;  
 }  
  
 //Mark it 'false' as it got one chance  
 // and will be replaced next time unless accessed again  
 second\_chance[pointer] = false;  
  
 //Pointer is updated in round robin manner  
 pointer = (pointer+1)%frames;  
 }  
 }  
  
 static void printHitsAndFaults(String reference\_string,  
 int frames)  
 {  
 int pointer,i,l,x,pf;  
  
 //initially we consider frame 0 is to be replaced  
 pointer = 0;  
  
 //number of page faults  
 pf = 0;  
  
 //Create a array to hold page numbers  
 int arr[] = new int[frames];  
  
 //No pages initially in frame,  
 //which is indicated by -1  
 Arrays.*fill*(arr,-1);  
  
 //Create second chance array.  
 //Can also be a byte array for optimizing memory  
 boolean second\_chance[] = new boolean[frames];  
  
 //Split the string into tokens,  
 //that is page numbers, based on space  
 String str[] = reference\_string.split(" ");  
  
 //get the length of array  
 l = str.length;  
  
 for(i = 0; i<l; i++)  
 {  
  
 x = Integer.*parseInt*(str[i]);  
  
 //Finds if there exists a need to replace  
 //any page at all  
 if(!*findAndUpdate*(x,arr,second\_chance,frames))  
 {  
 //Selects and updates a victim page  
 pointer = *replaceAndUpdate*(x,arr,  
 second\_chance,frames,pointer);  
  
 //Update page faults  
 pf++;  
 }  
 }  
  
 System.*out*.println("Total page faults were "+pf);  
 }  
}

# **Chapter 3: Discussion**

Choosing the algorithm for memory allocation is very significant process, because it allows the system work optimization and efficient allocation of resources. Therefore, this research and practical implementation of the first fit algorithm as one of the variants is a good explanatory method that will clarify the situations in which it is needed.

## **3.1 First-fit algorithm**

In the first fit, the partition is allocated which is first sufficient from the top of Main Memory.

**Example :**

Input : blockSize[] = {100, 500, 200, 300, 600};

processSize[] = {212, 417, 112, 426};

Output:

Process No. Process Size Block no.

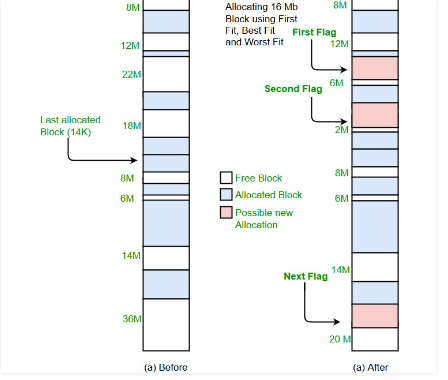
1 212 2

2 417 5

3 112 2

4 426 Not Allocated

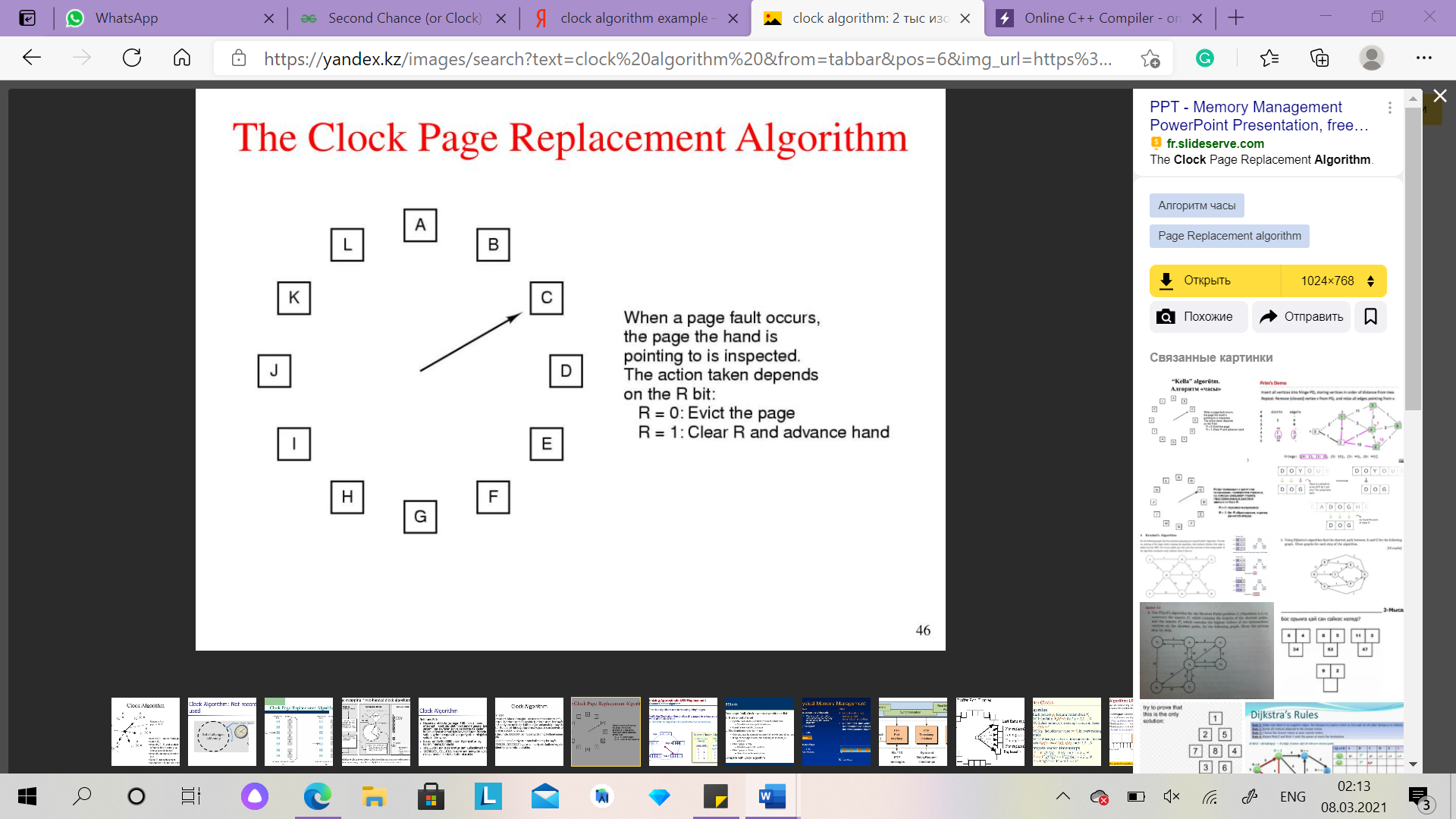
Its advantage is that it is the fastest search as it searches only the first block i.e. enough to assign a process.



It may have problems of not allowing processes to take space even if it was possible to allocate. Consider the above example, process number 4 (of size 426) does not get memory. However it was possible to allocate memory if we had allocated using [best fit allocation](http://quiz.geeksforgeeks.org/operating-system-memory-management-partition-allocation-method/) [block number 4 (of size 300) to process 1, block number 2 to process 2, block number 3 to process 3 and block number 5 to process 4].

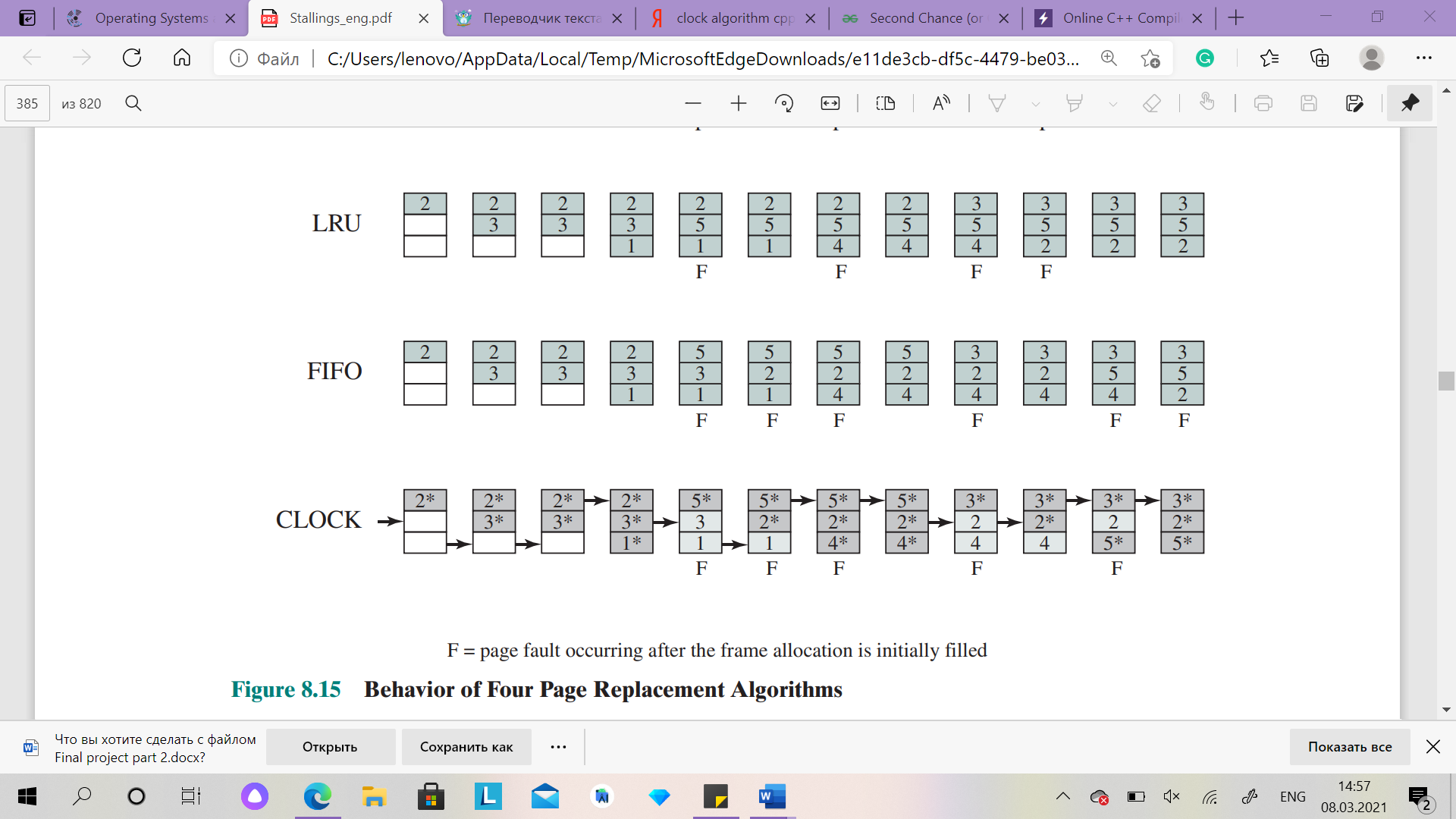
## **3.2 Second chance(clock) algorithm**

As stated above, the clock algorithm is an improved version of the second chance algorithm. For this reason, the second chance algorithm will be present in the clock algorithm, however, the system for moving pages in the clock algorithm disappears because the arrow, which points to the page, shuffles. Whereas in the second chance algorithm, the page itself is moved to the end of the queue.



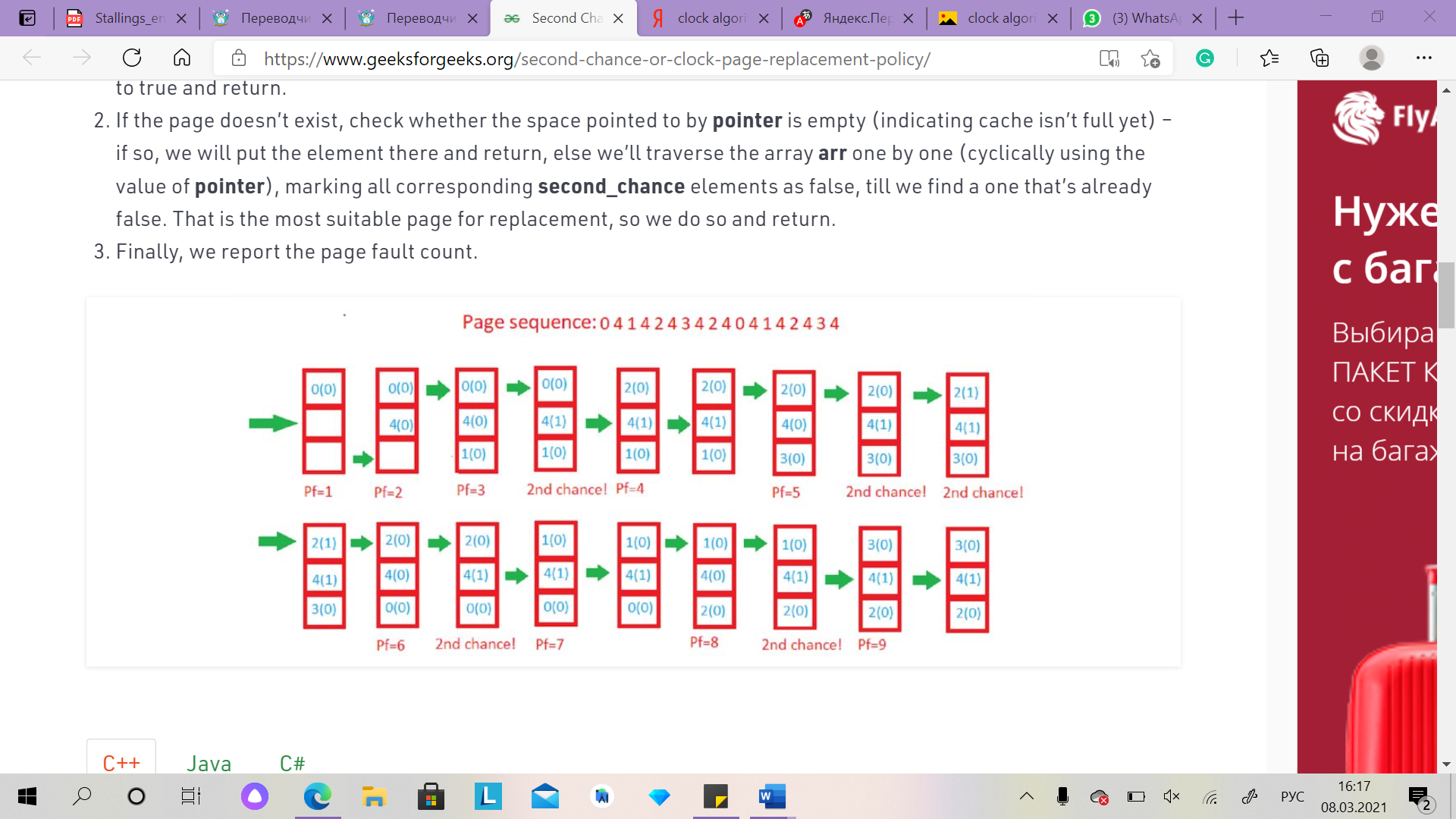
This is a light example of how the clock algorithm works.

As shown in figure, you can see an easy diagram of how the clock algorithm works, each time the page bit is equal to 1, the arrow will move forward, from page С to page D.



This is an example of how the clock algorithm works.

Figure above also shows a more detailed way of moving the arrow and resuming bits. Whereas Figure 3.3 shows an example, and further down in Chapter 4, you can see the parsing and code of this example. As shown in other 2 figures, there are 3 frames and they are empty, for this reason, when pages are requested, the frames will be filled with page digits, and each time a new page is requested, the first page whose bit is 0 is removed and a new page is loaded in its place and the arrow continues its way.



This is an example of how the clock algorithm works with the second chance algorithm.

In figure above, we are shown the reference string that is equals to 0 4 1 4 2 4 3 4 2 4 0 4 1 4 2 4 3 4, and we have 3 frames. In this case, the algorithm will track the second chance bit and the pointer, so the clock algorithm works for us. The first Pass-3s are loading reference strings, and since they have not been referenced yet, their second chance array will be 0, 0, 0. And the pointer will go back to 0. Pass-4: Frame = 0, 4, 1, second chance = 0, 1, 0 (4 will get a second chance), pointer = 0 (there is no need to refresh the page, so the candidate is still a page in the frame. 0), pf = 3 (without increasing the number of page errors). And this process continues until the algorithm reads the entire reference line. Ultimately, we get the number of page errors.

Almost all well-known modern memory managers try to use the clock algorithm. Such controllers store two bits in the page or segment descriptor - a clock bit and a modification flag. The modification flag is set at the first write to the page, in the descriptor of which this flag was cleared.

# **Chapter 4: Conclusion**

To sum up, after completing all the practical and theoretical tasks, it can be seen that First-fit algorithm gives opportunity to fast search, due to it searches only the first block and its enough to assign process. But also it has the drawbacks, that First-Fit may have problems of not allowing processes to take space even if it was possible to allocate.

In addition, Second Chance algorithm create an array frames to track the pages currently in memory and another Boolean array second\_chance to track whether that page has been accessed since it’s last replacement (that is if it deserves a second chance or not) and a variable pointer to track the target for replacement and finally its report the page fault count. These two algorithms are too flexible and in general easy to use. If you face with tasks like in example, you will know what you should use!

# **References**

1. Program for first fit algorithm in memory management. (2020, November 06). Retrieved March 09, 2021, from <https://www.geeksforgeeks.org/program-first-fit-algorithm-memory-management/>
2. The second Chance page replacement policy. (n.d.). Retrieved March 09, 2021, from <http://www.mathcs.emory.edu/~cheung/Courses/355/Syllabus/9-virtual-mem/SC-replace.html>
3. Second Chance (or Clock) Page Replacement Policy. [Second Chance (or Clock) Page Replacement Policy - GeeksforGeeks](https://www.geeksforgeeks.org/second-chance-or-clock-page-replacement-policy/)
4. Operating systems internals and design principles 7 edition. [Stallings\_eng.pdf](file:///C:\Users\lenovo\AppData\Local\Temp\MicrosoftEdgeDownloads\59eb2d60-c6da-41dc-b5e1-c1e7775ebc13\Stallings_eng.pdf)
5. Modern operating systems 2 edition by Andrew S. Tanenbaum. [Tanenbaum\_eng.pdf](file:///C:\Users\lenovo\AppData\Local\Temp\MicrosoftEdgeDownloads\3c5af8c9-bbbe-42c4-8aaf-1db5666e80df\Tanenbaum_eng.pdf)