

Page Replacement

First In First Out

```
#include<stdio.h>

int main()
{
    int incomingStream[] = {1,2,2,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
    int pageFaults = 0;
    int frames = 3;
    int m, n, s, pages;

    pages = sizeof(incomingStream)/sizeof(incomingStream[0]);

    printf("Incoming \t Frame 1 \t Frame 2 \t Frame 3");
    int temp[frames];
    for(m = 0; m < frames; m++)
    {
        temp[m] = -1;
    }

    for(m = 0; m < pages; m++)
    {
        s = 0;

        for(n = 0; n < frames; n++)
        {
            if(incomingStream[m] == temp[n])
            {
                s++;
                pageFaults--;
            }
        }
    }
}
```

```

    }
}
pageFaults++;

if((pageFaults <= frames) && (s == 0))
{
    temp[pageFaults-1] = incomingStream[m];
}
else if(s == 0)
{
    temp[(pageFaults - 1) % frames] = incomingStream[m];
}

printf("\n");
printf("%d\t\t",incomingStream[m]);
for(n = 0; n < frames; n++)
{
    if(temp[n] != -1)
        printf(" %d\t\t", temp[n]);
    else
        printf(" - \t\t");
}
}

printf("\nTotal Page Faults:\t%d\nTotal Hits:\t%d\n", pageFaults,(pages-pageFaults));
return 0;
}

```

Output

```

(kali1@kali)~/@1_DDrive/Code_Files/21bce1070
$ g++ OS.c
$ ./a.out

```

Incoming	Frame 1	Frame 2	Frame 3
1	1	-	-
2	1	2	-
3	1	2	3
4	4	2	3
2	4	2	3
1	4	1	3
5	4	1	5
6	6	1	5
2	6	2	5
1	6	2	1
2	6	2	1
3	3	2	1
7	3	7	1
6	3	7	6
3	3	7	6
2	2	7	6
1	2	1	6
2	2	1	6
3	2	1	3
6	6	1	3
Total Page Faults: 16			
Total Hits: 4			

```

(kali1@kali)~/@1_DDrive/Code_Files/21bce1070
$

```

Least Recently Used

```
#include <stdio.h>
```

```
#include <limits.h>
```

```
int checkHit(int incomingPage, int queue[], int occupied)
```

```
{
```

```
    for (int i = 0; i < occupied; i++)
```

```
    {
```

```
        if (incomingPage == queue[i])
```

```
            return 1;
```

```
    }
```

```

    return 0;
}

void printFrame(int queue[], int occupied)
{
    for (int i = 0; i < occupied; i++)
        printf("%d\t\t", queue[i]);
}

int main()
{
    int incomingStream[] = {1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
    int n = sizeof(incomingStream) / sizeof(incomingStream[0]);
    int frames = 3;
    int queue[frames];
    int distance[frames];
    int occupied = 0;
    int pagefault = 0;

    printf("Page\t Frame1 \t Frame2 \t Frame3\n");

    for (int i = 0; i < n; i++)
    {
        printf("%d \t\t", incomingStream[i]);

        if (checkHit(incomingStream[i], queue, occupied))
        {
            // Move the accessed page to the front
            int page = incomingStream[i];
            int j;
            for (j = 0; j < occupied; j++)

```

```

{
    if (queue[j] == page)
        break;
}

// Shift the remaining pages to the right
for (int k = j; k > 0; k--)
    queue[k] = queue[k - 1];

// Place the accessed page at the front
queue[0] = page;

printFrame(queue, occupied);
}
else if (occupied < frames)
{
    queue[occupied] = incomingStream[i];
    occupied++;

    printFrame(queue, occupied);
    pagefault++;
}
else
{
    int max = INT_MIN;
    int index;

    for (int j = 0; j < frames; j++)
    {
        distance[j] = 0;

```

```

    for (int k = i - 1; k >= 0; k--)
    {
        ++distance[j];

        if (queue[j] == incomingStream[k])
            break;
    }

    if (distance[j] > max)
    {
        max = distance[j];
        index = j;
    }
}

queue[index] = incomingStream[i];
printFrame(queue, occupied);
pagefault++;
}

printf("\n");
}

printf("Page Fault: %d\nHits: %d\n", pagefault, n-pagefault);

return 0;
}

```

Output

```

(kali1@kali)-[~/@1_DDrive/Code_Files/21bce1070]
$ g++ OS.c -std=c++11 -O2 -o a.out

(kali1@kali)-[~/@1_DDrive/Code_Files/21bce1070]
$ ./a.out
Page    Frame1    Frame2    Frame3
1        1
2        1 ("00", incomingStream[1]);
3        1        2        3
4        4 lockBit(incomingStream[2], queue, occupied);
2        2        4        3
1        2 Place the accessed page at the front
5        2 if page == incomingStream[i];
6        6 if i;
2        6 if i == 0; i < occupied; i++)
1        6        1        2
2        2 if (queue[i] == page)
3        2        3        1
7        2        3        7
6        6        3        7
3        3 shift the remaining pages to the right
2        3 for (int k = i; k < 0; k--)
1        3 queue[k] = queue[k-1];
2        2        3        1
3        3 Place the accessed page at the front
6        3 queue[0] = page;        2        6
Page Fault: 15
Hits: 5    printf("Stream ");

```

Optimized

```
#include <stdio.h>
```

```
int search(int key, int frame_items[], int frame_occupied)
```

```
{
    for (int i = 0; i < frame_occupied; i++)
        if (frame_items[i] == key)
            return 1;
    return 0;
}
```

```
void printOuterStructure(int max_frames)
```

```
{
    printf("Stream ");
}
```

```

    for (int i = 0; i < max_frames; i++)
        printf("Frame%d ", i + 1);
}

```

```

void printCurrFrames(int item, int frame_items[], int frame_occupied, int max_frames)
{
    printf("\n%d \t\t", item);
    for (int i = 0; i < max_frames; i++)
    {
        if (i < frame_occupied)
            printf("%d \t\t", frame_items[i]);
        else
            printf("- \t\t");
    }
}

```

```

int predict(int ref_str[], int frame_items[], int refStrLen, int index, int frame_occupied)
{
    int result = -1, farthest = index;
    for (int i = 0; i < frame_occupied; i++)
    {
        int j;
        for (j = index; j < refStrLen; j++)
        {
            if (frame_items[i] == ref_str[j])
            {
                if (j > farthest)
                {
                    farthest = j;
                    result = i;
                }
            }
        }
    }
}

```



```

        break;
    }
}
if (j == refStrLen)
    return i;
}
return (result == -1) ? 0 : result;
}

```

```

void optimalPage(int ref_str[], int refStrLen, int frame_items[], int max_frames)

```

```

{
    int frame_occupied = 0;
    printOuterStructure(max_frames);
    int hits = 0;
    for (int i = 0; i < refStrLen; i++)
    {
        if (search(ref_str[i], frame_items, frame_occupied))
        {
            hits++;
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
            continue;
        }
        if (frame_occupied < max_frames)
        {
            frame_items[frame_occupied] = ref_str[i];
            frame_occupied++;
            printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
        }
        else
        {
            int pos = predict(ref_str, frame_items, refStrLen, i + 1, frame_occupied);

```

```

        frame_items[pos] = ref_str[i];

        printCurrFrames(ref_str[i], frame_items, frame_occupied, max_frames);
    }
}

printf("\n\nHits: %d\n", hits);
printf("Misses: %d", refStrLen - hits);
}

int main()
{
    int ref_str[] = {1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6};
    int refStrLen = sizeof(ref_str) / sizeof(ref_str[0]);
    int max_frames = 3;
    int frame_items[max_frames];

    optimalPage(ref_str, refStrLen, frame_items, max_frames);
    return 0;
}

```

Output

```
(kali1@kali)-[~/@1_DDrive/Code_Files/21bce1070]
$ g++ OS.c

(kali1@kali)-[~/@1_DDrive/Code_Files/21bce1070]
$ ./a.out
Stream Frame1 Frame2 Frame3
1 1 1 1
2 1 1 2
3 1 1 2
4 1 1 2
5 1 1 2
6 1 1 2
7 1 1 2
8 1 1 2
9 1 1 2
10 1 1 2
11 1 1 2
12 1 1 2
13 1 1 2
14 1 1 2
15 1 1 2
16 1 1 2
17 1 1 2
18 1 1 2
19 1 1 2
20 1 1 2
21 1 1 2
22 1 1 2
23 1 1 2
24 1 1 2
25 1 1 2
26 1 1 2
27 1 1 2
28 1 1 2
29 1 1 2
30 1 1 2
31 1 1 2
32 1 1 2
33 1 1 2
34 1 1 2
35 1 1 2
36 1 1 2
37 1 1 2
38 1 1 2
39 1 1 2
40 1 1 2
41 1 1 2
42 1 1 2
43 1 1 2
44 1 1 2
45 1 1 2
46 1 1 2
47 1 1 2
48 1 1 2
49 1 1 2
50 1 1 2
51 1 1 2
52 1 1 2
53 1 1 2
54 1 1 2
55 1 1 2
56 1 1 2
57 1 1 2
58 1 1 2
59 1 1 2
60 1 1 2
61 1 1 2
62 1 1 2
63 1 1 2
64 1 1 2
65 1 1 2
66 1 1 2
67 1 1 2
68 1 1 2
69 1 1 2
70 1 1 2
71 1 1 2
72 1 1 2
73 1 1 2
74 1 1 2
75 1 1 2
76 1 1 2
77 1 1 2
78 1 1 2
79 1 1 2
80 1 1 2
81 1 1 2
82 1 1 2
83 1 1 2
84 1 1 2
85 1 1 2
86 1 1 2
87 1 1 2
88 1 1 2
89 1 1 2
90 1 1 2
91 1 1 2
92 1 1 2
93 1 1 2
94 1 1 2
95 1 1 2
96 1 1 2
97 1 1 2
98 1 1 2
99 1 1 2
100 1 1 2
101 1 1 2
102 1 1 2
103 1 1 2
104 1 1 2
105 1 1 2
106 1 1 2
107 1 1 2
108 1 1 2
109 1 1 2
110 1 1 2
111 1 1 2
112 1 1 2
113 1 1 2
114 1 1 2
115 1 1 2
116 1 1 2
117 1 1 2
118 1 1 2
119 1 1 2
120 1 1 2
121 1 1 2
122 1 1 2
123 1 1 2
124 1 1 2
125 1 1 2
126 1 1 2
127 1 1 2
128 1 1 2
129 1 1 2
130 1 1 2
131 1 1 2
132 1 1 2
133 1 1 2
134 1 1 2
135 1 1 2
136 1 1 2
137 1 1 2
138 1 1 2
139 1 1 2
140 1 1 2
141 1 1 2
142 1 1 2
143 1 1 2
144 1 1 2
145 1 1 2
146 1 1 2
147 1 1 2
148 1 1 2
149 1 1 2
150 1 1 2
151 1 1 2
152 1 1 2
153 1 1 2
154 1 1 2
155 1 1 2
156 1 1 2
157 1 1 2
158 1 1 2
159 1 1 2
160 1 1 2
161 1 1 2
162 1 1 2
163 1 1 2
164 1 1 2
165 1 1 2
166 1 1 2
167 1 1 2
168 1 1 2
169 1 1 2
170 1 1 2
171 1 1 2
172 1 1 2
173 1 1 2
174 1 1 2
175 1 1 2
176 1 1 2
177 1 1 2
178 1 1 2
179 1 1 2
180 1 1 2
181 1 1 2
182 1 1 2
183 1 1 2
184 1 1 2
185 1 1 2
186 1 1 2
187 1 1 2
188 1 1 2
189 1 1 2
190 1 1 2
191 1 1 2
192 1 1 2
193 1 1 2
194 1 1 2
195 1 1 2
196 1 1 2
197 1 1 2
198 1 1 2
199 1 1 2
200 1 1 2
201 1 1 2
202 1 1 2
203 1 1 2
204 1 1 2
205 1 1 2
206 1 1 2
207 1 1 2
208 1 1 2
209 1 1 2
210 1 1 2
211 1 1 2
212 1 1 2
213 1 1 2
214 1 1 2
215 1 1 2
216 1 1 2
217 1 1 2
218 1 1 2
219 1 1 2
220 1 1 2
221 1 1 2
222 1 1 2
223 1 1 2
224 1 1 2
225 1 1 2
226 1 1 2
227 1 1 2
228 1 1 2
229 1 1 2
230 1 1 2
231 1 1 2
232 1 1 2
233 1 1 2
234 1 1 2
235 1 1 2
236 1 1 2
237 1 1 2
238 1 1 2
239 1 1 2
240 1 1 2
241 1 1 2
242 1 1 2
243 1 1 2
244 1 1 2
245 1 1 2
246 1 1 2
247 1 1 2
248 1 1 2
249 1 1 2
250 1 1 2
251 1 1 2
252 1 1 2
253 1 1 2
254 1 1 2
255 1 1 2
256 1 1 2
257 1 1 2
258 1 1 2
259 1 1 2
260 1 1 2
261 1 1 2
262 1 1 2
263 1 1 2
264 1 1 2
265 1 1 2
266 1 1 2
267 1 1 2
268 1 1 2
269 1 1 2
270 1 1 2
271 1 1 2
272 1 1 2
273 1 1 2
274 1 1 2
275 1 1 2
276 1 1 2
277 1 1 2
278 1 1 2
279 1 1 2
280 1 1 2
281 1 1 2
282 1 1 2
283 1 1 2
284 1 1 2
285 1 1 2
286 1 1 2
287 1 1 2
288 1 1 2
289 1 1 2
290 1 1 2
291 1 1 2
292 1 1 2
293 1 1 2
294 1 1 2
295 1 1 2
296 1 1 2
297 1 1 2
298 1 1 2
299 1 1 2
300 1 1 2
301 1 1 2
302 1 1 2
303 1 1 2
304 1 1 2
305 1 1 2
306 1 1 2
307 1 1 2
308 1 1 2
309 1 1 2
310 1 1 2
311 1 1 2
312 1 1 2
313 1 1 2
314 1 1 2
315 1 1 2
316 1 1 2
317 1 1 2
318 1 1 2
319 1 1 2
320 1 1 2
321 1 1 2
322 1 1 2
323 1 1 2
324 1 1 2
325 1 1 2
326 1 1 2
327 1 1 2
328 1 1 2
329 1 1 2
330 1 1 2
331 1 1 2
332 1 1 2
333 1 1 2
334 1 1 2
335 1 1 2
336 1 1 2
337 1 1 2
338 1 1 2
339 1 1 2
340 1 1 2
341 1 1 2
342 1 1 2
343 1 1 2
344 1 1 2
345 1 1 2
346 1 1 2
347 1 1 2
348 1 1 2
349 1 1 2
350 1 1 2
351 1 1 2
352 1 1 2
353 1 1 2
354 1 1 2
355 1 1 2
356 1 1 2
357 1 1 2
358 1 1 2
359 1 1 2
360 1 1 2
361 1 1 2
362 1 1 2
363 1 1 2
364 1 1 2
365 1 1 2
366 1 1 2
367 1 1 2
368 1 1 2
369 1 1 2
370 1 1 2
371 1 1 2
372 1 1 2
373 1 1 2
374 1 1 2
375 1 1 2
376 1 1 2
377 1 1 2
378 1 1 2
379 1 1 2
380 1 1 2
381 1 1 2
382 1 1 2
383 1 1 2
384 1 1 2
385 1 1 2
386 1 1 2
387 1 1 2
388 1 1 2
389 1 1 2
390 1 1 2
391 1 1 2
392 1 1 2
393 1 1 2
394 1 1 2
395 1 1 2
396 1 1 2
397 1 1 2
398 1 1 2
399 1 1 2
400 1 1 2
401 1 1 2
402 1 1 2
403 1 1 2
404 1 1 2
405 1 1 2
406 1 1 2
407 1 1 2
408 1 1 2
409 1 1 2
410 1 1 2
411 1 1 2
412 1 1 2
413 1 1 2

```